Budapest University of Technology and Economics

Faculty of Electrical Engineering and Informatics

Department of Control Engineering and Information Technology

Institute for Computer Science and Control

Systems and Control Laboratory

Visual-inertial navigation with intermittent or spoofed GPS information

MASTER'S THESIS

Author

Zsombor Novozánszki

Advisor

Dr. Emese Szádeczky-Kardoss Gincsainé

Dr. Péter Bauer

December 10, 2023

TANSZÉKVEZETŐ

DIPLOMATERV FELADAT

Novozánszki Zsombor
szigorló villamosmérnök hallgató részére

Vizuális inerciális navigáció időszakos vagy hamisított GPS információval

A drónok elterjedésével és tervezett városi bevezetésével egyre inkább előtérbe kerül a GPS nélküli navigáció kutatása akár a GPS zavarására vagy eltérítésére készülve, akár figyelembe véve az épületek közti jelvesztés problémáját.

A feladat célja pilóta nélküli légieszköz IMU (Inertial Measurement Unit, inerciális mérőegység) és mono kamera alapú navigációja szakaszosan nem elérhető, vagy megzavart globális pozíció információk mellett.

A hallgató feladatának a következőkre kell kiterjednie:

- Tekintse át a terület irodalmát, válassza ki a releváns műveket.
- Ismerje meg a Számítógépes Optikai Érzékelés és Feldolgozás Kutatólaboratórium ezen a területen elért eredményeit.
- Implementáljon egy IMU és mono kamera alapú navigációs algoritmust és tesztelje városi és természeti környezet felett készült szintetikus vagy valós felvételeken különböző repülési sebességekkel és manőverekkel. Mérje fel a rendszer korlátait!
- Vizsgálja meg, hogy GPS-en kívül milyen globális pozíció információk érkezhetnek a rendszerbe! Végezze el a fenti megoldás globális pozíció pontosítását elérhető adatok esetén.
- Vizsgálja meg, hogyan lehetséges GPS spoofing detektálása a fenti megoldások alkalmazásával!

Külső konzulens: Dr. Bauer Péter, tudományos főmunkatárs
ELKH SZTAKI Rendszer és Irányításelméleti Kutatólaboratórium
Tanszéki konzulens: Gincsainé Szádeczky-Kardoss Emese, docens
Diplomaterv nyelve: angol

Budapest, 2023. március 30.

/ Dr. Kiss Bálint /
docens
tanszékvezető

Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai kar
Irányítástechnika és Informatika Tanszék

1117 Budapest, Magyar tudósok körútja 2. I. ép. III.em.
Postacím: 1521 Budapest, Pf.: 91.
Tel: 463-2699, Fax: 463-2204, http://www.iit.bme.hu

CONTENTS

Hallgatói nyilatkozat ii

Kivonat iii

Abstract v

1 Introduction 1

1.1 Structure of the paper 3

2 Mathematical foundations 5

2.1 Navigation frames 5

2.1.1 North-East-Down reference frame 6

2.1.2 Body frame 7

2.2 Camera frames 9

2.2.1 Camera system 10

2.3 Pinhole camera projection 11

2.4 Quaternions 13

3 Introduction of the applied filter technique 15

3.1 In general about the Kalman filter 16

3.2 Error-state kinematics 16

3.2.1 Kalman filter states for IMU driven systems 17

3.2.2 ESKF states 18

3.2.3 Error-state kinematics in discrete time 22

3.3 Measurement equation 23
3.4 ESKF framework 24
3.4.1 Prediction step 24
3.4.2 Update step 26
3.4.3 Error injection into the nominal-state 28

4 Overview of the triangulation method **30**
4.1 Problem statement 30
4.1.1 Line of sight (LOS) measurements 32
4.2 LOST method 32
4.2.1 Covariance calculations 33
4.2.2 The optimal solution 35
4.2.3 Practical formalization of the estimator 36
4.2.4 Covariance of the estimator 37
4.3 The recursive version 37

5 Synergizing LOST and ESKF for robust navigation **39**
5.1 The modified Kalman gain 39
5.1.1 The error propagation from measurement to measurement . 39
5.1.2 The measurement model 40
5.1.3 The optimal solution for Kalman gain 41
5.2 Cross-covariance estimation 42

6 Development **44**
6.1 Simulation environment 44
6.2 Noise calibration 45
6.3 Camera configuration 45

6.3.1 Feature point selection 45
6.4 Simulation results 47
6.4.1 Straight and level flight path 47
6.4.2 Straight and descending flight path 49
6.4.3 Zig-zag flight path 49

7 Conclusion **50**

A Quaternion operations **51**
A.1 Rotation vector to quaternion formula 52
A.2 The time derivative of quaternion 52
A.3 Jacobian of quaternion rotation with respect to the vector 54
A.4 Jacobian of quaternion rotation with respect to the quaternion . . . 54

B ESKF-related equations **55**
B.1 True rotation matrix 55
B.2 Error-state equations 56
B.2.1 Position error 56
B.2.2 Angular error 57
B.2.3 Velocity error 58
B.3 Jacobian of true quaternion with respect to the error rotation vector 59
B.4 Jacobian of reset function (g) with respect to the error rotation vector 59

Abbreviations **61**

List of Figures **62**

List of Tables **62**

Bibliography **63**

HALLGATÓI NYILATKOZAT

Alulírott *Novozánszki Zsombor*, szigorló hallgató kijelentem, hogy ezt a diplomatervet meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy autentikált felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Budapest, 2023. december 10.

Novozánszki Zsombor
hallgató

KIVONAT

Az elmúlt évtizedekben a pilóta nélküli légijárművek (Unmanned Aerial Vehicle, UAV) egyre népszerűbbé váltak civil és akadémiai alkalmazásokban is, mint például felderítés, áruszállítás, geofizikai adatgyűjtés, mentési műveletek vagy éppen mezőgazdasági célú felhasználás. Az előrelépés nemcsak a technikai modernizációnak köszönhető, hanem a szélesebb körű felhasználhatóságnak, ami nagyban a hagyományos navigációs módszerek innovációjának köszönhető. A UAV-k jövőre gyakorolt hatása azon is múlik mennyire tudnak jól navigálni GPS (Global Positioning System) nélküli környezetben például kimaradó, zavart (jamming) vagy hamisított (spoofing) jelek esetén.

Dolgozatom középpontjában egy vizuális-inerciális navigációs algoritmus áll. A tanulmány során megvalósított rendszer célja meghatározni egy légijármű pozícióját, orientációját, sebességét és a gyorsulás és szögsebesség szenzor bias értékeit inerciális szenzorrendszer (Inertial Measurement Unit, IMU) mérések és mono kameraképek alapján. A javasolt rendszer egy hiba állapot Kálmán-szűrő (Error-State Kalman Filter, ESKF) alapú keretrendszerben hajtja végre az IMU és a kamera adatok integrációját. Az IMU mérésekből a repülőgép pozícióját, sebességét és orientációját lehet becsülni, amelynek hibáját a kamera képekből nyert információval korrigálom. A korrekció során felhasznált jellegpontok pozícióját háromszögelés útján számítom, amelyhez egy ún. LOST (Linear Optimal Sine Triangulation) módszert alkalmazok. Az algoritmus tesztelése úgy történik, hogy a kezdeti állapotban ismert GPS koordinátákat feltételezek, és ez idő alatt már megkezdődik a jellegpontok pozíciójának optimalizációja, amelyeket az ESKF frissítési fázisában használok fel. Később a GPS koordináták már nem ismertek, ezért az ESKF becslésekből történik az újabb jellegpontok pozíciójának optimalizációja, azok alapján pedig a frissítés. Munkámban megvizsgálom a módszer pontosságának korlátait, és azt is, hogy a GPS jel elvesztése után meddig ad kielégítő pontosságot csak a vizuális-inerciális navigáció.

Összefoglalva, a dolgozat bemutatja az algoritmus fejlesztését, amely fuzionálja az inerciális és vizuális adatokat, valamint az ehhez szükséges elméleti alapismereteket és matematikai módszertanokat. Az alap algoritmusokat (ESKF és LOST) szakirodalomból vettem, amelyeket saját rendszerbe integráltam. A kutatás során először egy szimulációt hoztam létre, amelyet fokozatosan közelítettem a valóságos körülményeket leginkább modellező környezethez. Az ígéretes szimulációs eredményeket követően fő célom az elkészített navigációs rendszer tesztelése lesz szintetikus vagy valós felvételeken.

ABSTRACT

Over the past decades, Unmanned Aerial Vehicles (UAVs) have become increasingly popular in both civilian and academic applications, such as exploration, cargo delivery, geophysical data collection, rescue operations, and agricultural purposes. This expansion required not only technical modernization but also advancement in traditional navigation methods. The future impact of UAVs also depends on their ability to navigate effectively in GPS (Global Positioning System)-denied environments, for example, scenarios involving signal dropout, jamming, or spoofing.

My work focuses on a visual-inertial navigation algorithm. In this study, the implemented system's goal is to determine an aircraft's position, orientation, velocity, and bias values of the accelerometer and gyroscope based on measurements from an Inertial Measurement Unit (IMU) and monocular camera images. The proposed system performs the integration of IMU and camera data within an Error-State Kalman Filter (ESKF) framework. The aircraft's position, velocity, and orientation are estimated from IMU measurements, and these estimates are corrected with the information obtained from camera images. During the correction, the positions of feature points are computed through triangulation using a method called Linear Optimal Sine Triangulation (LOST). The algorithm is tested as follows: in the initial stage the GPS coordinates are assumed to be known, and the optimization of feature point positions begins this time, and they are utilized in the ESKF update. Later, when GPS coordinates are no longer known, the optimization of feature point positions is based on the ESKF estimates, and these optimized values are used in the update. In my work, I examine the limitations of the method's accuracy and investigate how long it can provide satisfactory accuracy after losing the GPS signal, relying only on visual-inertial navigation.

To summarize, the thesis presents the development of an algorithm that fuses inertial and visual data besides the required theoretical background and mathematical foundations. The utilized algorithms (ESKF and LOST) were chosen from the literature and integrated into the newly developed system. I applied a simulation environment and a gradual approach during the development process, starting with ideal conditions and incrementally introducing more realistic elements to the simulation. After achieving promising simulation results, my primary objective in the future is to test the developed navigation system on synthetic or real-world footage.

Chapter 1

Introduction

Until the early 2000s Unmanned Aerial Vehicles (UAV) have been limited to the defense and military industries, because of the high costs and the complexity of constructing these vehicles. Nevertheless, they have become cheaper and more available in several civil and academic applications over the past decades. They have not just turned more common, but their capabilities have dramatically increased, therefore they are more popular and widely used in applications such as rescue operations [1], data collection and geophysics exploration [2], inspections [3], and agricultural purposes [4].

This expansion required not only technical developments but advancement in traditional navigation methods, where Global Positioning System (GPS) is combined with Inertial Navigation System (INS), creating GPS-INS system [5]. The small unmanned aircraft's future impact depends on how well they can navigate in GPS-denied environments such as narrow city corridors or circumstances with GPS disturbance or spoofing. Inertial measurements by themselves can be used to estimate the position of the aircraft respected to a known initial position, but they will accumulate errors over time, especially with low-cost Inertial Measurement Unit (IMU) sensors. This phenomenon is usually called drift because estimated values drift away from the true values with time.

To give a better estimation of the position in most GPS-free applications exteroceptive sensors are used such as cameras, laser scanners, distance sensors, etc. The type of used sensor mostly depends on the kind of vehicle. For example only considering UAVs, a multirotor drone has completely different aircraft dynamics, mission profile, and sensing requirements compared to fixed-wing aircraft. Since fixed-wing aircraft usually fly at high altitudes above the environment with rel-

atively high speeds, distance sensors are ineffective. In this case, the most common approach is using cameras for instance, both [6] and [7] leverage visual information captured by cameras and integrate this data with measurements from an IMU to estimate the motion of the aircraft. When an algorithm fails to close the navigation loop, particularly in the absence of external absolute positioning references like GPS, it results in the estimates drift, which emerges due to the algorithm's inability to establish consistent and accurate reference points or constraints, thereby leading to unbounded cumulative errors over time.

The measurement fusion and sensor noise filtering can take place in an Extended Kalman Filter (EKF) framework because typically these are nonlinear systems. EKFs can be used on any kind of vehicle including robots [8], Unmanned Aerial Systems(UAS) [9, 10], etc. They can account for both sensor errors and process uncertainty, but it is important to note that these methods only work well when errors remain small, e.g. when the availability of GPS measurements makes it possible to regularly remove drift errors. When GPS or any other global measurements are unavailable for a longer period, the global position and yaw angle of the aircraft is unobservable, which eventually leads to the divergence of estimates [11, 12]. In addition, if an EKF receives a global measurement after significant drift errors have accumulated, nonlinearities can complicate the utilization of the measurement, and it could result in large jumps in the estimate, in some cases it can even lead to filter divergence. This is because the local linearization of the measurement equations around the drifted states is applied, which can present incorrect dynamics.

In recent years a new method was proposed called relative navigation [13, 14], which can handle these observability and consistency problems. With exteroceptive sensors navigation can be performed concerning the surroundings, hence this method can be divided into a relative front-end and a global back-end, the complete architecture is shown in Figure 1.1. The front end provides an estimation of the state relative to the local environment, while the back end is an optimization algorithm, which uses the calculations of the front end to produce global estimates. Several important observability and computational benefits are

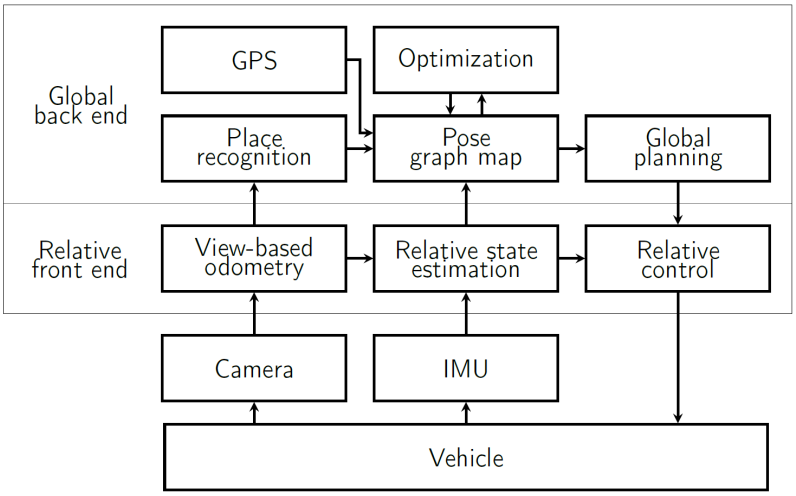


Figure 1.1: Block diagram of relative navigation ([7]: page 2, Figure 2)

obtained by dividing the architecture into a relative front end and a global back end:

- The front end calculates the estimate relative to a local frame, where the states can remain observable and the uncertainty can be accurately represented by a Gaussian distribution. This enables the utilization of the computational advantage of an Error-State Kalman Filter (ESKF), which is a better solution, than an ordinary EKF because it calculates the nominal state according to the original non-linear dynamics, therefore the linearization around this state is a better approximation.

- On the other hand, the back end uses a graph that can effectively represent nonlinearities in heading and can be robustly optimized with additional constraints, such as opportunistic global measurements or place recognition.

In this paper, a visual-inertial navigation algorithm will be presented, which is based on [7]. The target UAV is a fixed-wing aircraft called Sindy, shown in Figure 1.2.

1.1 Structure of the paper

To sum up the involved steps during the project, the primary objective was to integrate a visual-inertial ESKF into a simulation that was already available to

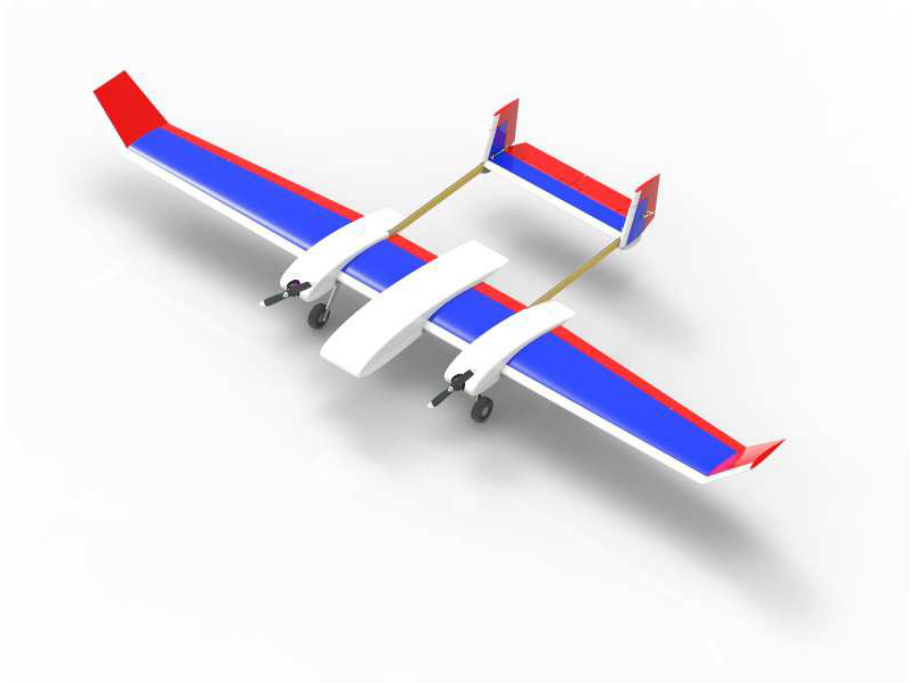


Figure 1.2: Realistic drawing of Sindy ([15]: title page)

me. The simulation consisted of a grid of feature points, and the goal was to use an ESKF-based algorithm to estimate the aircraft's state while flying along a predefined trajectory. I followed a gradual approach in the development process, starting with ideal conditions and incrementally introducing more realistic elements to the simulation. These included IMU- and measurement- noises, sensor biases, and pixelization. Firstly, I implemented the previously described filter with known 3-D coordinates of the measured feature points, but later a triangulation method was implemented called Linear Optimal Sine Triangulation (LOST). The next step involved the insertion of the LOST estimates into the ESKF framework.

The paper is structured as follows: Chapter 2 provides an introduction to several fundamental mathematical concepts that are crucial for comprehending our approach. In Chapter 3, the mathematical foundations of the filter are explained, and the steps of the filter are detailed. Chapter 4 focuses on the introduction of the applied triangulation method called LOST. Chapter 5 is devoted to introducing the integration issues of ESKF and LOST. Chapter 6 presents the estimation results in Matlab/Simulink environment. Finally, Chapter 7 offers a summary of the work, and highlights certain future development steps.

CHAPTER 2

MATHEMATICAL FOUNDATIONS

Before I delve into the details of the visual-inertial relative navigation algorithm, it is important to establish theoretical foundations. This chapter summarizes the mathematical preliminaries that are essential for understanding how the algorithm works. It covers key concepts about coordinate systems and camera projection and introduces an alternative method of rotation representation.

In this paper, filter-related mathematics uses 3+1 frames: Earth (E), node (N)-, body (B), and camera (C) frame. The purpose of the node frame is mentioned later which is connected to relative navigation applications, but currently, there is not applied such a frame in my approach. The Earth alias localization, body, and camera frames are utilized in the calculations of the algorithm. The frames are shown in Figure 2.1.

The Earth frame is an Earth fixed frame and in this application, the North-East-Down (NED) coordinate system was chosen for this purpose because it can be used as an inertial frame in such UAV applications where the vehicle covers only a few kms[1]. In the following, the filter-related mathematical notation uses n, b, and c for inertial, body, and camera frames.

2.1 Navigation frames

The purpose of using different kinds of frames is to facilitate the kinematic modeling of UAVs. To effectively study UASs, it is crucial to comprehend the relative orientation and translation of different coordinate systems. Multiple frames are required for several reasons:

[1] If flight trajectories remain short the curvature of the Earth can be neglected.

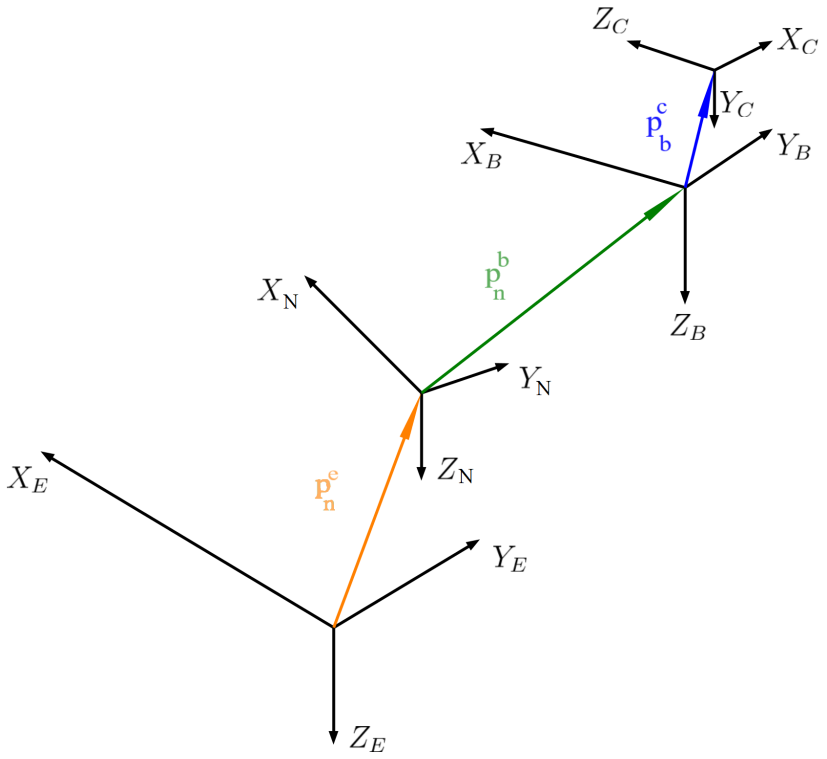


Figure 2.1: Applied coordinate systems

- The motion of the aircraft is most easily described in a body-fixed frame, however, Newton's equations of motion are derived relative to an inertial reference frame.
- The body-fixed frame is also used to express the aerodynamic forces and moments that affect the aircraft.
- On-board sensors such as accelerometers and gyroscopes provide measurements concerning the body frame in the case of strap-down IMUs.
- Finally, the mission requirements of the aircraft, e.g. flight path require a global (absolute) frame.

2.1.1 North-East-Down reference frame

The NED coordinate system has emerged as a standard in UAV applications over limited distances, typically spanning a few kilometers. Notably, this reference frame is conventionally anchored to a stationary point on Earth, affording its use as an inertial reference frame in analytical computations. It can be seen in Figure 2.2 compared to the globe and Earth-centered Earth-fixed (ECEF) coordinate

system. Consistent with its name, the NED system aligns its X-axis with the geographic north, its Y-axis with the east, and its Z-axis points inwards to the Earth. Its X-Y plane is the local tangent plane of the Earth's ellipsoid.

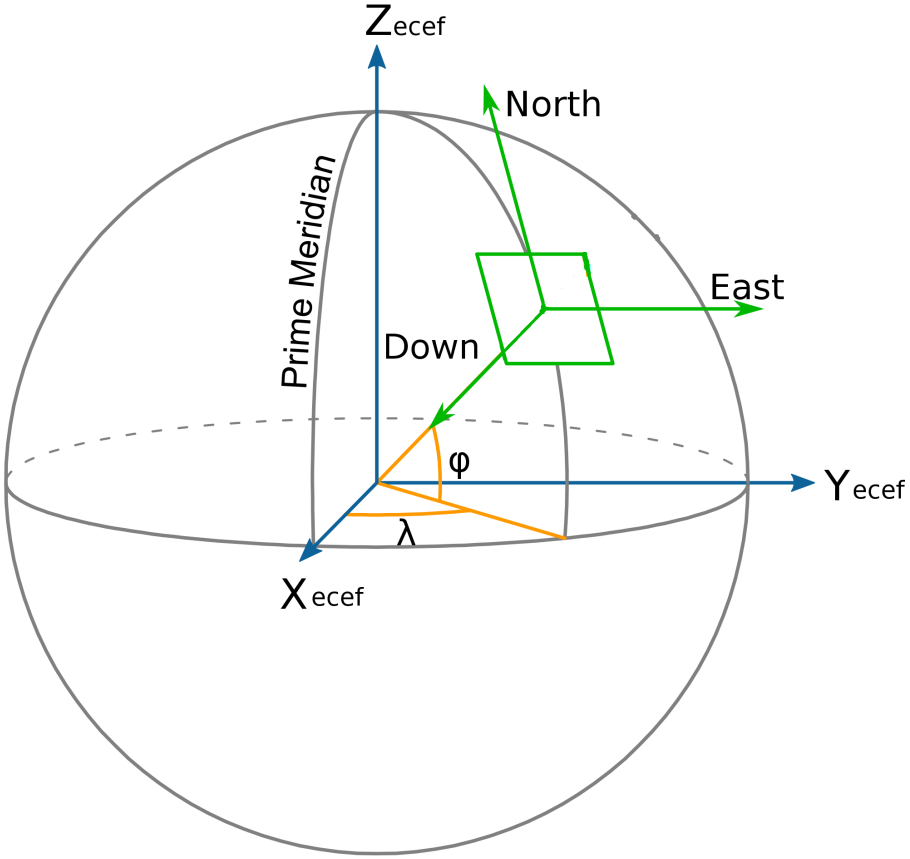


Figure 2.2: NED coordinate system [16]

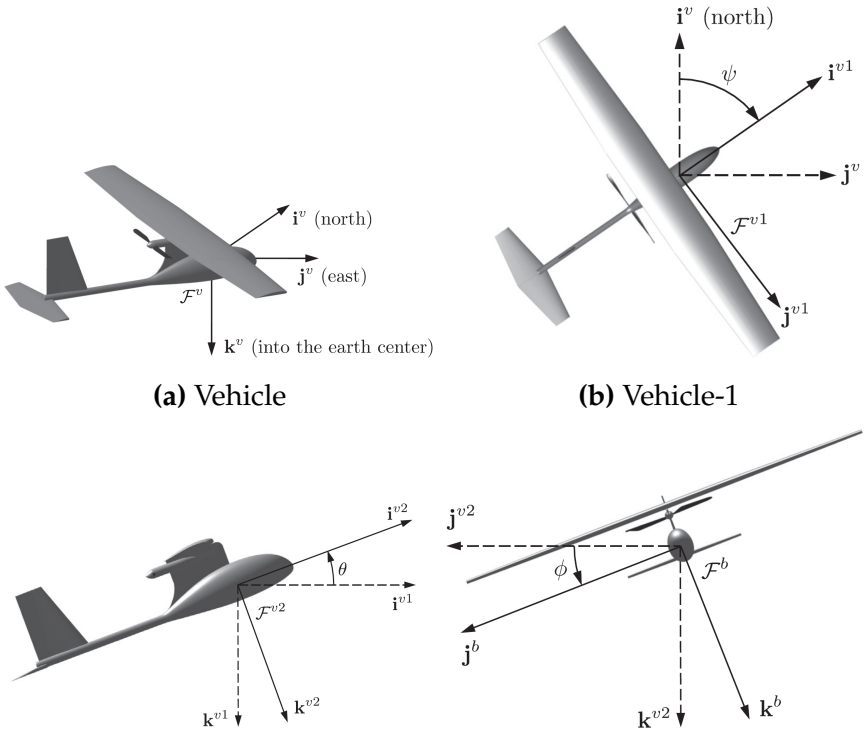
The front end of relative navigation methods adopts the node coordinate system as an inertial frame, a crucial precondition for the functioning of the filter. For example in [7] the node frame remains locally fixed and undergoes redeclaration whenever they perform a measurement update. The back end is responsible for producing global estimates, such as NED coordinates, utilizing the outcomes yielded by the filter.

2.1.2 Body frame

In the kinematic equations, quantities are usually described in inertial or body frame, therefore it is crucial to understand properly how the body frame is defined and how can the relation can be accounted to the inertial frame. The origin is aligned with the center of the aircraft's mass, and worth mentioning the fact

in realizations the IMU is placed here. The axes of the body frame are defined as follows: the X-axis points out the nose of the aircraft, the Y-axis points out the right wing and the Z-axis points downwards.

Once the coordinate system has been defined, it is necessary to mention the relation to the inertial frame. The most commonly applied approach to obtain the orientation of the aircraft in the NED system is to express it with Euler angles which means three rotations one after another. Defining the Euler angles I use the same terminology as in [10], their approach is to introduce three additional coordinate systems: vehicle, vehicle-1, and vehicle-2. They are shown in Figure 2.3a-2.3d.



(a) Vehicle **(b)** Vehicle-1

(c) Vehicle-2 **(d)** Body frame

Figure 2.3: Frames to define Euler angles ([10]: pages 13-15, Figures 2.4-2.7)

The vehicle frame is the NED coordinate system with shifted origin into the center of mass, vehicle-1 frame is rotated with the yaw angle (ψ) around the Z-axis of the vehicle frame, vehicle-2 frame is rotated with pitch angle (θ) around the Y-axis of vehicle-1 frame, and body frame is rotated with roll angle (ϕ) around the X-axis of vehicle-2 frame. To summarize, the resulting rotation can be defined as:

$$\begin{aligned}\mathbf{R}_{BV}(\phi,\theta,\psi) &= \mathbf{R}_{BV_2}(x,\phi)\mathbf{R}_{V_2V_1}(y,\theta)\mathbf{R}_{V_1V}(z,\psi) \\ &= \begin{bmatrix} C_\theta C_\psi & C_\theta S_\psi & -S_\theta \\ S_\phi S_\theta C_\psi - C_\phi S_\psi & S_\phi S_\theta S_\psi + C_\phi C_\psi & S_\phi C_\theta \\ C_\phi S_\theta C_\psi + S_\phi S_\psi & C_\phi S_\theta S_\psi - S_\phi C_\psi & C_\phi C_\theta \end{bmatrix},\end{aligned} \tag{2.1}$$

where C_α is shorthand for $\cos(\alpha)$ and S_α for $\sin(\alpha)$.

Finally, I want to mention the usage of homogeneous transformation which is widespread regarding vision-based applications, thanks to the fact it allows the calculation of three-dimensional (3-D) coordinates from one frame to another with a single matrix multiplication. The homogeneous transformation between two frames requires the usage of both orientation and position relative to each other. For example, considering body-to-earth transformation $\mathbf{R}_{NB} \equiv \mathbf{R}_{VB}$ accounts for rotation, and $\mathbf{p}_n^b$ is the translational vector describing the position of the body frame's origin in NED coordinates. Then the homogeneous transformation can be expressed as:

$$\mathbf{H}_{NB} = \begin{bmatrix} \mathbf{R}_{NB} & \mathbf{p}_n^b \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{2.2}$$

The transformation can be applied as:

$$\mathbf{H}_{NB} \begin{bmatrix} \mathbf{v}_b \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{NB} & \mathbf{p}_n^b \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{v}_b \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{NB}\mathbf{v}_b + \mathbf{p}_n^b \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{v}_n \\ 1 \end{bmatrix} \tag{2.3}$$

2.2 Camera frames

It is important to say a few words about camera-related frames and transformation since camera pictures are used to improve the navigation algorithm. A camera- and an image coordinate system are distinguished, the first one is interpreted in 3-D, and its origin is aligned with the camera center. The image coordinate system spans two dimensions and it is usually one of the XY-planes of the camera frame.

To mathematically model the projection, the camera's intrinsic and extrinsic parameters have to be used. For the pinhole model, the intrinsic parameters of the camera are focal length (f), principal point (p_x, p_y), and a constraint for the visible points is the field of view (FOV) or image size. Extrinsic parameters describe the position and orientation of the camera in the inertial frame.

2.2.1 Camera system

It is very important to determine the transformation from body-to-camera system with sufficient accuracy. Generally, the camera can not be placed in the body center, therefore the transformation requires a position that is described compared to the body frame and detonated as $\mathbf{p}_b^c$. As regards the orientation, since navigation happens concerning the surroundings an at least partially downward-facing camera is mandatory, thus introducing a rotation around the Y-axis of the body frame. Furthermore, the axes of the camera system are defined differently, than the axes of the body frame, therefore an additional operation is essential to change the axes. Figure 2.4 shows the camera frame and the image plane.

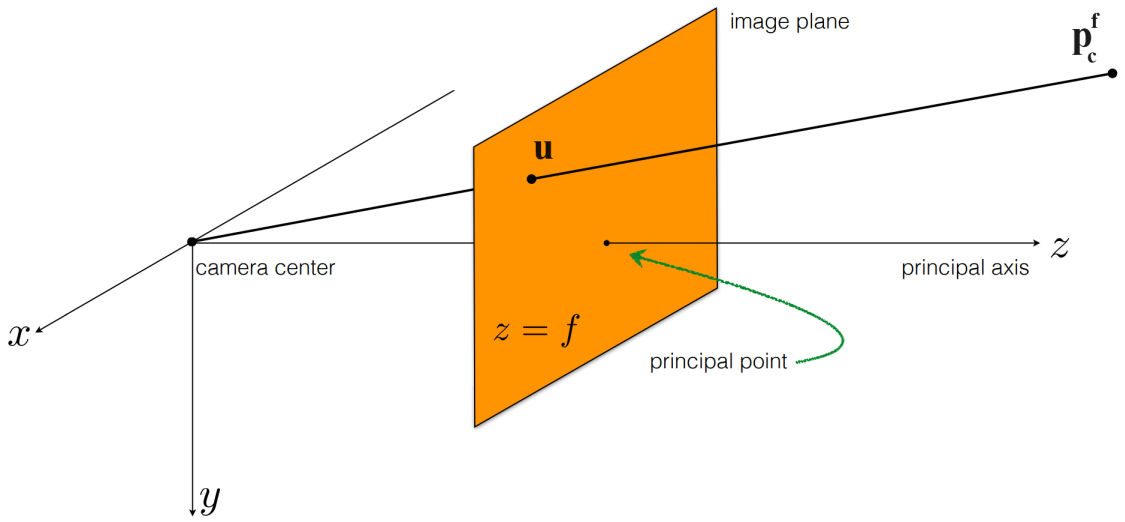


Figure 2.4: Camera and image system ([17]: page 7)

The axes of the image system are aligned with the camera system's X and Y axes but shifted with the principal point. The principal point is usually near to the center of the image and shifting with it results in a top left corner origin of the pixel coordinate system.

The axes swap operation of the rotated body frame should be transformed as outlined below: $\mathbf{i}'_b = \mathbf{k}_c$, $\mathbf{j}'_b = \mathbf{i}_c$ and $\mathbf{k}'_b = \mathbf{j}_c$, where $\mathbf{i}$, $\mathbf{j}$ and $\mathbf{k}$ are unit vectors along X-, Y- and Z-axis respectively. Utilizing the linear transformations the whole transformation from body-to-camera frame:

$$\begin{aligned}\mathbf{T}_{CB} = \mathbf{S}\mathbf{R}_{CB}(y,-\beta) &= \begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ 1 & 0 & 0\end{bmatrix}\begin{bmatrix}\cos(\beta) & 0 & -\sin(\beta)\\ 0 & 1 & 0\\ \sin(\beta) & 0 & \cos(\beta)\end{bmatrix}\\ &= \begin{bmatrix}0 & 1 & 0\\ \sin(\beta) & 0 & \cos(\beta)\\ \cos(\beta) & 0 & -\sin(\beta)\end{bmatrix},\end{aligned} \tag{2.4}$$

where $\mathbf{S}$ stands for the axes swapping transformation, and $\mathbf{R}_{CB}$ represents the camera rotation with a β angle around the Y-axis (for downward facing camera it's negative). It is still unitary matrix, therefore $\mathbf{T}_{BC} = \mathbf{T}_{CB}^T$.

Summarising both translational and rotational effects the whole camera-to-NED transformation is:

$$\begin{aligned}\mathbf{H}_{NC} = \mathbf{H}_{NB}\mathbf{H}_{BC} &= \begin{bmatrix}\mathbf{R}_{NB} & \mathbf{p}_n^b\\ \mathbf{0}^T & 1\end{bmatrix}\begin{bmatrix}\mathbf{T}_{BC} & \mathbf{p}_b^c\\ \mathbf{0}^T & 1\end{bmatrix}\\ &= \begin{bmatrix}\mathbf{R}_{NC} & \mathbf{p}_n^b + \mathbf{R}_{NB}\mathbf{p}_b^c\\ \mathbf{0}^T & 1\end{bmatrix} = \begin{bmatrix}\mathbf{R}_{NC} & \mathbf{p}_n^c\\ \mathbf{0}^T & 1\end{bmatrix}\end{aligned} \tag{2.5}$$

2.3 Pinhole camera projection

The pinhole camera model serves as a mathematical representation for the projection $h(\mathbf{p}_c^f)$, which results in the pixel measurement. This model is articulated within the camera frame and necessitates intrinsic parameters of the camera and the feature point $\mathbf{p}_c^f$ in the camera frame.

The pinhole model framework describes the aperture of the camera as a point, therefore the lens distortion errors are neglected [18, 19]. In reality, the transformation yields an inverted image behind the $z = 0$ plane, but it is more illustrative

to depict it in front of the $z = 0$ plane as can be seen in Figure 2.4. Mathematically the two interpretations will produce equivalent solutions. The parameters of the model:

1. The focal length (f) which shows the distance between the camera center and the image plane.

2. The principal point (p_x, p_y) which is typically near the center of the image.

The projection first requires the normalization of $\mathbf{p}_c^f = \begin{bmatrix} x_c & y_c & z_c \end{bmatrix}^T$ by its z coordinate, and then it can be projected onto the image plane with the camera matrix, which can be formalized with the usage of intrinsic parameters:

$$\mathbf{K} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & f \end{bmatrix} \tag{2.6}$$

First applying the normalization, then using (2.6) the whole projection is defined as:

$$h(\mathbf{p}_c^f) = \mathbf{K}\frac{\mathbf{p}_c^f}{z_c} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & f \end{bmatrix} \begin{bmatrix} \frac{x_c}{z_c} \\ \frac{y_c}{z_c} \\ 1 \end{bmatrix} = \begin{bmatrix} f\frac{x_c}{z_c} + p_x \\ f\frac{y_c}{z_c} + p_y \\ f \end{bmatrix} = \begin{bmatrix} u \\ v \\ f \end{bmatrix} = \overline{\mathbf{u}}, \tag{2.7}$$

where $\mathbf{u}$ is the resulting measurement which pixel coordinates are u and v and its z coordinate equals the focal length since it points onto the image. The interpretation range of the pinhole model contains every $\mathbf{p}_c^f$ which is not in the z=0 plane, but as a final point, I want to emphasize that the constraint finite image size (W, H) or FOV angles have to be applied for real cameras. The image size and FOV angles result in the same constraint in fact with known focal length and constraint they can be calculated from each other.

2.4 Quaternions

Before embarking on the ESKF, it is worth mentioning a few words about quaternions. Originally formulated in [20] by Sir William Rowan in the 19th century. Nowadays, quaternions have found extensive applications in various domains, including computer graphics, robotics, and aerospace engineering.

In this paper, I adopt the same mathematical representation for quaternions as described in [21]. Quaternions belong to the extended complex number space, which includes two additional imaginary units, namely j and k, in addition to the real and imaginary unit i. As a result, quaternions can be represented by four-component vectors:

$$\mathbf{q} = a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k} = \begin{bmatrix} a & b & c & d \end{bmatrix}^T = \begin{bmatrix} q_w \\ \mathbf{q}_v \end{bmatrix}, \tag{2.8}$$

where $\mathbf{q}$ is a quaternion with a, b, c, and d parameters. The quaternion can be expressed as a column vector or as a combination of the scalar component q_w and the vector component $\mathbf{q}_v$.

It is noticeable that, while regular complex numbers of unit length ($\mathbf{z} = e^{i\theta}$) can encode rotations in the 2D space, quaternions of unit length ($\mathbf{q} = e^{(u_x i + u_y j + u_z k)\theta/2}$) can encode rotations in the 3-D space. These quaternions can always be written in the form:

$$\mathbf{q} = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right)\mathbf{u} \end{bmatrix}, \tag{2.9}$$

where $\mathbf{u}$ is the rotation axis and θ is the angle of the rotation. The rotation can be performed on the 3-D vector $\mathbf{v}$ by the following operation:

$$\begin{bmatrix} 0 \\ \mathbf{v}' \end{bmatrix} = \mathbf{q} \otimes \begin{bmatrix} 0 \\ \mathbf{v} \end{bmatrix} \otimes \mathbf{q}^*, \tag{2.10}$$

where $\otimes$ is the Hamiltonian-quaternion multiplication, and $\mathbf{q}^*$ is the conjugate of $\mathbf{q}$.

The composition of two rotation, described by $\mathbf{q}_{AB}$ and $\mathbf{q}_{BC}$ quaternions can also be evaluated using quaternion product:

$$\mathbf{q}_{AC} = \mathbf{q}_{AB} \otimes \mathbf{q}_{BC} \tag{2.11}$$

Further definitions and important formulas relevant to this paper are provided in Appendix A.

Finally, I aim to justify the significance of quaternions in the field of computer calculations. The first striking benefit is that quaternions offer a compact representation of rotations, requiring only 4 parameters. In contrast, rotation matrices necessitate 9 parameters. Another advantage of quaternions is their ability to ensure more stable and accurate computations by avoiding singularities and mitigating the issue of gimbal lock, which can occur both for rotation matrices and Euler angles.

CHAPTER 3

INTRODUCTION OF THE APPLIED FILTER TECHNIQUE

Upon laying down essential mathematical foundations, the subsequent focus shifts toward introducing the chosen filter technique. In visual-inertial projects, various approaches are available to filter measurements and estimate the system's state. Many of these approaches are a modified Kalman filter. The basics of the implemented filter are based on the ESKF, which stands as a remarkable method for filtering and estimating nonlinear systems. The basic algorithm is taken from [21], but with three major differences:

1. I didn't insert the gravitational constant ($\mathbf{g}$) in the state vectors $\mathbf{x}$, $\delta\mathbf{x}$ to estimate because UAVs usually fly small distances and the gravitational acceleration can be assumed constant.

2. The velocity vector $\mathbf{v}$ is body-fixed in my approach, making it $\mathbf{v}_b$. On the contrary, the mentioned literature uses an inertial frame fixed velocity vector.

3. In their method, the rotation described by quaternion $\mathbf{q}$ and rotation matrix $\mathbf{R}\{\mathbf{q}\} \equiv \mathbf{R}$ [1] stands for the body-to-earth rotation, while in this paper rotation stands for the inverse transformation, earth-to-body transformation to match the conventional Euler angle representation of rotation commonly applied in aerospace. This impacts the formulation of the mathematical equations, but in most cases, the equations have symmetrical properties.

[1] In this paper, $\mathbf{R}\{\mathbf{q}\}$ detones the rotation matrix obtained from quaternion.

3.1 In general about the Kalman filter

The Kalman filter is a recursive algorithm used for state estimation in a time-varying linear system. It combines information from past measurements and predictions of the system's state to produce an optimal estimate of the current state. It does this by incorporating a weighted average of the predicted state and the new measurement, where the weights are calculated optimally considering the uncertainty in the state and measurements.

The different types of Kalman filters have the same property in that they involve a prediction and an update step. The prediction step propagates the state through time according to the system's dynamics and it's also called a priori information. The update step involves measurements which is a posteriori information obtained from the real-world system. The Kalman filter gives an optimal solution in recursive form, therefore the new estimation is calculated as a composition of the propagated state and residual. The residual carries the new information defined as the difference between the measurement and the propagated state. To sum up, Kalman filters provide an optimal solution to a linear system by combining the a priori and a posteriori information. If Kalman filters are applied to non-linear systems, then the system dynamics and measurement models have to be linearized [22].

3.2 Error-state kinematics

In IMU-driven systems the goal is to create a filter framework, that integrates the accelerometer and gyrometer readings considering their bias and measurement noise. As I previously detailed the IMU measurements only themselves cause drift in their estimate, therefore they should be fused with absolute position readings such as GPS or vision.

The ESKF defines an ideal state called a nominal state and applies an error-state relative to that. During the state propagation, the system is propagated based on the ideal non-linear system dynamics and accounts for the evolution of the error

state. The assumption is that the error state always operates close to the actual system state, thus it has several advantages:

- Since the error state remains small the linear Kalman filter approach is a better estimation for the error state.
- All second-order products are negligible because the error state remains small. This makes the computation of Jacobians very easy and fast.
- The dynamics of the error state are small and slow, due to all large-signal dynamics being integrated into the nominal dynamics. This results in a highly beneficial property: the KF corrections can be applied at a lower rate, than the predictions.

3.2.1 Kalman filter states for IMU driven systems

The objective is to estimate the position of the aircraft, which requires to use of the kinematic equations of the aircraft. The kinematics expresses the relationships among position, orientation, velocity, acceleration, and angular velocity. Whereas acceleration ($\mathbf{a}_m$) and angular velocity ($\boldsymbol{\omega}_m$) are utilized as IMU measurements, the position ($\mathbf{p}_n^b$), velocity ($\mathbf{v}_b$), and orientation ($\mathbf{q}_{NB}$) are included in the state. Moreover, the state is supplemented by estimating the biases of the sensors ($\boldsymbol{\beta}_a$, $\boldsymbol{\beta}_\omega$).

The mentioned quantities are expressed respectively to the body or inertial frame. I did not explicitly mark that measurements are also captured concerning the body frame because it is unambiguous in the case of strap-down IMUs. In the context of ESKF, I use $\mathbf{p}$ and $\mathbf{q}$ lighter notations for accounting for the relation between the body and the inertial frame. This results in the state vector:

$$\mathbf{x} = \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \\ \mathbf{v}_b \\ \boldsymbol{\beta}_a \\ \boldsymbol{\beta}_\omega \end{bmatrix} \tag{3.1}$$

3.2.2 ESKF states

In the ESKF framework, three different states are defined: true ($\mathbf{x}_t$), nominal ($\mathbf{x}$), and error state ($\delta\mathbf{x}$). All of the ESKF variables are summarized in Table 3.1.

True	Nominal	Error	Composition	Noise	Measured
$\mathbf{p}_t$	$\mathbf{p}$	$\delta\mathbf{p}$	$\mathbf{p}_t = \mathbf{p} + \delta\mathbf{p}$		
$\mathbf{q}_t$	$\mathbf{q}$	$\delta\mathbf{q} \approx \begin{bmatrix} 1 \\ \frac{1}{2}\delta\boldsymbol{\theta} \end{bmatrix}$	$\mathbf{q}_t = \delta\mathbf{q} \otimes \mathbf{q}$		
$\mathbf{v}_{b,t}$	$\mathbf{v}_b$	$\delta\mathbf{v}_b$	$\mathbf{v}_{b,t} = \mathbf{v}_b + \delta\mathbf{v}_b$		
$\boldsymbol{\beta}_{a,t}$	$\boldsymbol{\beta}_a$	$\delta\boldsymbol{\beta}_a$	$\boldsymbol{\beta}_{a,t} = \boldsymbol{\beta}_a + \delta\boldsymbol{\beta}_a$	$\boldsymbol{\eta}_{\beta_a}$	
$\boldsymbol{\beta}_{\omega,t}$	$\boldsymbol{\beta}_\omega$	$\delta\boldsymbol{\beta}_\omega$	$\boldsymbol{\beta}_{\omega,t} = \boldsymbol{\beta}_\omega + \delta\boldsymbol{\beta}_\omega$	$\boldsymbol{\eta}_{\beta_\omega}$	
$\mathbf{R}_t$	$\mathbf{R}$	$\delta\mathbf{R} = e^{\left[\delta\boldsymbol{\theta}\right]_\times}$	$\mathbf{R}_t = \delta\mathbf{R}\mathbf{R}$		
$\mathbf{a}_t$	$\mathbf{a}$	$\delta\mathbf{a}$	$\mathbf{a} + \delta\mathbf{a}$	$\boldsymbol{\eta}_a$	$\mathbf{a}_m$
$\boldsymbol{\omega}_t$	$\boldsymbol{\omega}$	$\delta\boldsymbol{\omega}$	$\boldsymbol{\omega} + \delta\boldsymbol{\omega}$	$\boldsymbol{\eta}_\omega$	$\boldsymbol{\omega}_m$

Table 3.1: ESKF states and inputs

The states are divided in such a way that the large-signal dynamics are integrated into the nominal state for example state and biases, while the small and slowly varying effects are assigned to the error state for instance measurement noise or bias varying. A few comments on the table:

- All quantity yields a simple sum operation on nominal and error state to get the true state except the orientation.

- In the nominal state, orientation is described using quaternion ($\mathbf{q}$). However, the error in the rotation is described with a rotation vector ($\delta\boldsymbol{\theta}$) that represents a small deviation or error from a reference rotation. The relation was presented between the quaternion and rotation vector in Appendix A.1. Utilizing the assumption that $\delta\boldsymbol{\theta}$ is always small its cosine and sine can be approximated as $\cos\left(\frac{\theta}{2}\right) = 1$ and $\sin\left(\frac{\theta}{2}\right) = \frac{\theta}{2}$ and this yields the formula in Table 3.1.

- The body-referenced measurements are $\mathbf{a}_m$ and $\boldsymbol{\omega}_m$ and they are captured as noisy IMU measurements. The noise impulses $\boldsymbol{\eta}_q$ and $\boldsymbol{\eta}_\omega$ are modeled with white Gaussian distribution. It is worth mentioning that non-g-compensated accelerometers measure gravitational acceleration therefore it has to be considered in the equations, thus:

$$\begin{aligned} \mathbf{a}_m &= \mathbf{a}_t - \mathbf{R}_t\mathbf{g} + \boldsymbol{\beta}_{a,t} + \boldsymbol{\eta}_a \\ \boldsymbol{\omega}_m &= \boldsymbol{\omega}_t + \boldsymbol{\beta}_{\omega,t} + \boldsymbol{\eta}_\omega \end{aligned} \tag{3.2}$$

  Substituting the true state models from Table 3.1 in (3.2), the true value of the acceleration and angular rate can be expressed:

$$\begin{aligned} \mathbf{a}_t &= \overbrace{\mathbf{a}_m - \boldsymbol{\beta}_a}^{\mathbf{a}} + \mathbf{R}\mathbf{g}\overbrace{-\delta\boldsymbol{\beta}_a - \boldsymbol{\eta}_a}^{\delta\mathbf{a}} \\ \boldsymbol{\omega}_t &= \overbrace{\boldsymbol{\omega}_m - \boldsymbol{\beta}_\omega}^{\boldsymbol{\omega}}\overbrace{-\delta\boldsymbol{\beta}_\omega - \boldsymbol{\eta}_\omega}^{\delta\boldsymbol{\omega}} \end{aligned} \tag{3.3}$$

  In (3.3) the error part of the biases $\delta\boldsymbol{\beta}_a$ and $\delta\boldsymbol{\beta}_\omega$ account for slowly varying biases over time since these parameters are typically temperature-dependent. It is also represented with Gaussian white noises.

- In the context of 3-D rotations, the unit quaternions and rotation matrices belong to the SO(3)[2] group. The skew-symmetric matrix of rotation vector $\left[\delta\boldsymbol{\theta}\right]_\times$ is part of the $\mathfrak{so}(3)$ Lie Algebra and in [21] they propose that the exponential map is a powerful mathematical tool to map it into $\mathrm{SO}(3)$ space, therefore the error rotation matrix $\delta\mathbf{R}$ can be expressed with the rotation vector as in Table 3.1. The value of $\mathbf{R}_t$ can be determined by expanding $\delta\mathbf{R}$ into Taylor series, neglecting the second- and higher-order terms, it yields:

$$\mathbf{R}_t = \delta\mathbf{R}\mathbf{R} = \left(\mathbf{I} + \left[\delta\boldsymbol{\theta}\right]_\times\right)\mathbf{R} + \mathcal{O}(||\delta\boldsymbol{\theta}||^2), \tag{3.4}$$

  where $\mathcal{O}(||\delta\boldsymbol{\theta}||^2)$ stands for the negligable terms. The calculations related to (3.4) are detailed in Appendix B.1.

[2]Special Orthogonal

True-state

The dynamics of the true state can be described by the following equations:

$$\dot{\mathbf{p}}_t = \mathbf{R}_t^T \mathbf{v}_{b,t} \tag{3.5a}$$

$$\dot{\mathbf{q}}_t = \frac{1}{2} \begin{bmatrix} 0 \\ -(\boldsymbol{\omega}_m - \boldsymbol{\beta}_{\omega,t} - \boldsymbol{\eta}_\omega) \end{bmatrix} \otimes \mathbf{q}_t \tag{3.5b}$$

$$\dot{\mathbf{v}}_{b,t} = \mathbf{a}_m - \boldsymbol{\beta}_{a,t} - \boldsymbol{\eta}_a + \mathbf{R}_t \mathbf{g} + \left[\mathbf{v}_{b,t}\right]_\times (\boldsymbol{\omega}_m - \boldsymbol{\beta}_{\omega,t} - \boldsymbol{\eta}_\omega) \tag{3.5c}$$

$$\dot{\boldsymbol{\beta}}_{a,t} = \boldsymbol{\eta}_{\beta_a} \tag{3.5d}$$

$$\dot{\boldsymbol{\beta}}_{\omega,t} = \boldsymbol{\eta}_{\beta_\omega} \tag{3.5e}$$

(3.5b) is the time derivative formula of quaternion, and detailed in Appendix A.2. (3.5c) also requires some explanation because this equation applies the rule of vector derivative in rotating frames compared to an inertial frame:

$$\frac{d}{dt_i}\mathbf{v} = \frac{d}{dt_b}\mathbf{v} + \boldsymbol{\omega}_{b/i} \times \mathbf{v} \Rightarrow \frac{d}{dt_b}\mathbf{v} = \frac{d}{dt_i}\mathbf{v} + \mathbf{v} \times \boldsymbol{\omega}_{b/i}, \tag{3.6}$$

where the anti-commutative property of the cross product was used ($\boldsymbol{\omega}_{b/i} \times \mathbf{v} = -\mathbf{v} \times \boldsymbol{\omega}_{b/i}$). In (3.5c), the cross product is expressed in matrix form with the skew operator $\left[.\right]_\times$, which can be constructed as:

$$\left[\mathbf{a}\right]_\times \triangleq \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \tag{3.7}$$

Nominal-state

The nominal state corresponds to the modeled system without noises and perturbations:

$$\dot{\mathbf{p}} = \mathbf{R}^T \mathbf{v}_b \tag{3.8a}$$

$$\dot{\mathbf{q}} = \frac{1}{2} \begin{bmatrix} 0 \\ -\boldsymbol{\omega} \end{bmatrix} \otimes \mathbf{q} \tag{3.8b}$$

$$\dot{\mathbf{v}}_b = \mathbf{a} + \mathbf{R}\mathbf{g} + \left[\mathbf{v}_b\right]_\times \boldsymbol{\omega} \tag{3.8c}$$

$$\dot{\boldsymbol{\beta}}_{a,t} = 0 \tag{3.8d}$$

$$\dot{\boldsymbol{\beta}}_{\omega,t} = 0 \tag{3.8e}$$

It results in constant biases in the nominal state.

Error-state

The error-state equations are derived by using the previously defined true- and nominal-state equations. The equations governing the error state are as follows:

$$\dot{\delta\mathbf{p}} = \mathbf{R}^T \left[\mathbf{v}_b\right]_\times \delta\boldsymbol{\theta} + \mathbf{R}^T \delta\mathbf{v}_b \tag{3.9a}$$

$$\dot{\delta\boldsymbol{\theta}} = -\left[\boldsymbol{\omega}_m + \boldsymbol{\beta}_\omega\right]_\times \delta\boldsymbol{\theta} + \delta\boldsymbol{\beta}_\omega + \boldsymbol{\eta}_\omega \tag{3.9b}$$

$$\begin{aligned} \dot{\delta\mathbf{v}_b} = &-\left[\mathbf{R}\mathbf{g}\right]_\times \delta\boldsymbol{\theta} - \left[\boldsymbol{\omega}_m - \boldsymbol{\beta}_\omega\right]_\times \delta\mathbf{v}_b - \delta\boldsymbol{\beta}_a - \left[\mathbf{v}_b\right]_\times \delta\boldsymbol{\beta}_\omega \\ &- \boldsymbol{\eta}_a - \left[\mathbf{v}_b\right]_\times \boldsymbol{\eta}_\omega \end{aligned} \tag{3.9c}$$

$$\dot{\delta\boldsymbol{\beta}}_a = \boldsymbol{\eta}_{\beta_a} \tag{3.9d}$$

$$\dot{\delta\boldsymbol{\beta}}_\omega = \boldsymbol{\eta}_{\beta_\omega} \tag{3.9e}$$

The calculations are detailed in Appendix B.2.

3.2.3 Error-state kinematics in discrete time

The previous equations are defined in continuous-time, but computer implementations use a discrete-time model. To incorporate discrete time intervals $\Delta t > 0$, the differential equations (3.9) must be transformed into difference equations through integration.

The integration method may vary, since in certain cases, exact closed-form solutions can be utilized, while in other cases, numerical integration techniques with varying degrees of accuracy may be employed. Generally, we could say the equations have a deterministic part related to state dynamics and control, on the other hand, there is a stochastic part related to perturbations and noises. In this case, the same method will be presented as in [21]: the deterministic part integrated normally, and the stochastic part modeled as random impulses. This results in the nominal state equations:

$$\mathbf{p}_{k+1} = \mathbf{p}_k + \mathbf{R}^T \mathbf{v}_{b,k} \Delta t + \frac{1}{2} \mathbf{R}^T \left(\mathbf{a}_k + \mathbf{R}\mathbf{g} + \left[\mathbf{v}_{b,k} \right]_\times \boldsymbol{\omega}_k \right) \Delta t^2 \tag{3.10a}$$

$$\mathbf{q}_{k+1} = \mathbf{q}\{-(\boldsymbol{\omega}_k - \boldsymbol{\beta}_\omega)\Delta t\} \otimes \mathbf{q}_k \tag{3.10b}$$

$$\mathbf{v}_{b,k+1} = \mathbf{v}_{b,k} + \left(\mathbf{a}_k + \mathbf{R}\mathbf{g} + \left[\mathbf{v}_{b,k} \right]_\times \boldsymbol{\omega}_k \right) \Delta t \tag{3.10c}$$

$$\boldsymbol{\beta}_{a,k+1} = \boldsymbol{\beta}_{a,k} \tag{3.10d}$$

$$\boldsymbol{\beta}_{\omega,k+1} = \boldsymbol{\beta}_{\omega,k} \tag{3.10e}$$

The position is integrated both from velocity and acceleration, and the rotation from angular velocity. Using the same integration approach complemented by the integration of the stochastic part, the error state equations result in:

$$\delta\mathbf{p}_{k+1} = \delta\mathbf{p}_k + \mathbf{R}^T \left(\delta\mathbf{v}_{b,k} + \left[\mathbf{v}_{b,k} \right]_\times \delta\boldsymbol{\theta}_k \right) \Delta t \tag{3.11a}$$

$$\delta\boldsymbol{\theta}_{k+1} = \mathbf{R}\{-(\boldsymbol{\omega}_{m,k} - \boldsymbol{\beta}_{\omega,k})\Delta t\}\delta\boldsymbol{\theta}_k + \delta\boldsymbol{\beta}_{\omega,k}\Delta t + \boldsymbol{\theta}_{i,k} \tag{3.11b}$$

$$\begin{aligned}\delta\mathbf{v}_{b,k+1} = \delta\mathbf{v}_{b,k} + \Big(&-\left[\mathbf{Rg}\right]_\times \delta\boldsymbol{\theta}_k - \left[\boldsymbol{\omega}_{m,k} - \boldsymbol{\beta}_{\omega,k}\right]_\times \delta\mathbf{v}_{b,k} \\ &-\delta\boldsymbol{\beta}_{a,k} - \left[\mathbf{v}_{b,k}\right]_\times \delta\boldsymbol{\beta}_{\omega,k}\Big)\,\Delta t - \left[\mathbf{v}_{b,k}\right]_\times \boldsymbol{\theta}_{i,k} - \mathbf{v}_{b,i,k}\end{aligned} \tag{3.11c}$$

$$\delta\boldsymbol{\beta}_{a,k+1} = \delta\boldsymbol{\beta}_{a,k} + \mathbf{a}_{i,k} \tag{3.11d}$$

$$\delta\boldsymbol{\beta}_{\omega,k+1} = \delta\boldsymbol{\beta}_{\omega,k} + \boldsymbol{\omega}_{i,k} \tag{3.11e}$$

$\boldsymbol{\theta}_i, \mathbf{v}_{b,i}, \mathbf{a}_i$ and $\boldsymbol{\omega}_i$ are the random impulses which form the stochastic part of the equation. Those noises which appear only negatively in the equations can be changed to positive freely. Their covariance matrices are integrated as (see [21] Appendix E for details):

$$\begin{aligned}\boldsymbol{\Theta}_i &= \sigma_{\eta_\omega}^2 \Delta t^2 \mathbf{I} \quad [rad^2] \\ \mathbf{V}_i &= \sigma_{\eta_a}^2 \Delta t^2 \mathbf{I} \quad [m^2/s^2] \\ \mathbf{A}_i &= \sigma_{\beta_a}^2 \Delta t \mathbf{I} \quad [m^2/s^4] \\ \boldsymbol{\Omega}_i &= \sigma_{\beta_\omega}^2 \Delta t \mathbf{I} \quad [rad/s],\end{aligned} \tag{3.12}$$

where $\sigma_{\eta_a}, \sigma_{\eta_\omega}, \sigma_{\beta_a}$ and σ_{β_ω} are the deviations of acceleration, gyroscope measurement noise and the slowly varying bias impacts.

3.3 Measurement equation

The measurement equation is based on the fundamental concept that the system can acquire pixel coordinates of feature points as measurements. To form the residual, the predicted pixel coordinates must be determined for each feature point. This can be accomplished through the application of the pinhole equation (2.7), which leads to:

$$h(\mathbf{T}_{CB}(\mathbf{R}_{BN}(\mathbf{p}_n^f - \mathbf{p}_n^b) - \mathbf{p}_b^c)) = h(\mathbf{p}_c^f) = \begin{bmatrix} u \\ v \\ f \end{bmatrix}, \tag{3.13}$$

where $\mathbf{p}_n^f$ denotes the feature position in NED frame. The transformation above requires the usage of both the state ($\mathbf{p}_n^b$, $\mathbf{q}_n^b$) and feature position $\mathbf{p}_n^f$ and it is worth highlighting that the feature positions are not known prior unless a visual map of the surroundings is available onboard.

3.4 ESKF framework

The ESKF is a special version of the Kalman filter, that contains two state vectors: one for the nominal- ($\mathbf{x}$) and the other for the error-state ($\delta\mathbf{x}$) vector. The system is governed by the IMU measurements $\mathbf{c}$ and the noise perturbations $\mathbf{i}$:

$$\mathbf{x} = \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \\ \mathbf{v}_b \\ \boldsymbol{\beta}_a \\ \boldsymbol{\beta}_\omega \end{bmatrix}, \qquad \delta\mathbf{x} = \begin{bmatrix} \delta\mathbf{p} \\ \delta\boldsymbol{\theta} \\ \delta\mathbf{v}_b \\ \delta\boldsymbol{\beta}_a \\ \delta\boldsymbol{\beta}_\omega \end{bmatrix}, \qquad \mathbf{u} = \begin{bmatrix} \mathbf{a}_m \\ \boldsymbol{\omega}_m \end{bmatrix}, \qquad \mathbf{i} = \begin{bmatrix} \boldsymbol{\theta}_i \\ \mathbf{v}_i \\ \mathbf{a}_i \\ \boldsymbol{\omega}_i \end{bmatrix} \tag{3.14}$$

3.4.1 Prediction step

One of the best properties of the ESKF framework is that during the prediction steps the nominal state can be calculated by the original nonlinear equations (3.10), but the error state has to be linearized. The transition- ($\mathbf{F}_x$) and noise matrix ($\mathbf{F}_i$) can be derived from (3.11), where $\delta\mathbf{x}_{k+1} = f(\delta\mathbf{x}_k)$:

$$\mathbf{F}_x = \frac{\partial f}{\partial \delta\mathbf{x}} = \begin{bmatrix} \mathbf{I} & \mathbf{R}^T \left[\mathbf{v}_b\right]_\times \Delta t & \mathbf{R}^T \Delta t & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}\{-(\boldsymbol{\omega}_m - \boldsymbol{\beta}_\omega)\Delta t\} & \mathbf{0} & \mathbf{0} & \mathbf{I}\Delta t \\ \mathbf{0} & -\left[\mathbf{R}\mathbf{g}\right]_\times \Delta t & \mathbf{I} - \left[\boldsymbol{\omega}_m - \boldsymbol{\beta}_\omega\right]_\times \Delta t & -\mathbf{I}\Delta t & -\left[\mathbf{v}_b\right]_\times \Delta t \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{3.15}$$

$$\mathbf{F}_i = \frac{\partial f}{\partial \mathbf{i}} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ -\left[\mathbf{v}_b\right]_\times & -\mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \quad \mathbf{0},\ \mathbf{I} \in \mathbb{R}^{3\times 3} \tag{3.16}$$

Now the error state dynamic is:

$$\delta \mathbf{x}_{k+1} = \mathbf{F}_x \delta \mathbf{x}_k + \mathbf{F}_i \mathbf{i}_k \tag{3.17}$$

The process noise matrix is defined as:

$$\mathbf{Q} = E\{\mathbf{i}\mathbf{i}^T\} = \begin{bmatrix} \mathbf{\Theta}_i & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{V}_i & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_i & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{\Omega}_i \end{bmatrix} \quad \mathbf{Q} \in \mathbb{R}^{12\times 12}, \quad \mathbf{0} \in \mathbb{R}^{3\times 3} \tag{3.18}$$

Using formulas above and denote equations in (3.10) with $f(.)$ the prediction step involves forecasting the nominal state and error state too:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) \tag{3.19a}$$

$$(\delta \hat{\mathbf{x}}_{k+1} = \mathbf{F}_x \delta \hat{\mathbf{x}}_k) \tag{3.19b}$$

$$\mathbf{P}_{k+1} = \mathbf{F}_x \mathbf{P}_k \mathbf{F}_x^T + \mathbf{F}_i \mathbf{Q} \mathbf{F}_i^T, \tag{3.19c}$$

where $\delta \hat{\mathbf{x}}$ is the mean of the error state, therefore $\delta \mathbf{x} \sim \mathcal{N}(\delta \hat{\mathbf{x}}, \mathbf{P})$, but when a measurement update happens and the error was injected into the nominal state it gets reset to 0, therefore accounting for the error only involves the propagation of the covariance matrix.

3.4.2 Update step

In this section, I focus on how the measurement Jacobian of the error state is calculated. The most straightforward approach to do that for a feature point involves the utilization of the chain rule. First, the Jacobian of the pixel measurement with respect to the feature point ($\mathbf{J} \in \mathbb{R}^{2\times3}$) has to be computed, then the Jacobian of the feature point with respect to the true state ($\mathbf{P}_{x_t} \in \mathbb{R}^{3\times16}$) and finally the Jacobian of the true state with respect to the error state ($\mathbf{X}_{\delta x} \in \mathbb{R}^{16\times15}$). So the jacobian of the measurement with respect to the error state:

$$\mathbf{H}_x = \frac{\partial h(\mathbf{p}_c^f)}{\partial \delta \mathbf{x}} = \frac{\partial h(\mathbf{p}_c^f)}{\partial \mathbf{p}_c^f} \frac{\partial \mathbf{p}_c^f}{\partial \mathbf{x}_t} \frac{\partial \mathbf{x}_t}{\partial \delta \mathbf{x}} = \mathbf{J}\mathbf{P}_{x_t}\mathbf{X}_{\delta x}, \tag{3.20}$$

Jacobian of the pixel measurements with respect to feature point

The Jacobian of the projection is calculated by linearizing (2.7), whereas the transformation results in a 3-D vector, but the derivative of the 3^{rd} coordinate leads to zeros, therefore it is neglected. It shows the property that we lost a bit of information during the projection.

$$\mathbf{J} = \begin{bmatrix} \frac{\partial h_1(\mathbf{p}_c^f)}{\partial x} & \frac{\partial h_1(\mathbf{p}_c^f)}{\partial y} & \frac{\partial h_1(\mathbf{p}_c^f)}{\partial z} \\ \frac{\partial h_2(\mathbf{p}_c^f)}{\partial x} & \frac{\partial h_2(\mathbf{p}_c^f)}{\partial y} & \frac{\partial h_2(\mathbf{p}_c^f)}{\partial z} \end{bmatrix} = \begin{bmatrix} \frac{f}{z} & 0 & -\frac{fx}{z^2} \\ 0 & \frac{f}{z} & -\frac{fy}{z^2} \end{bmatrix} \Bigg|_{\mathbf{p}_c^f} \tag{3.21}$$

Jacobian of the feature point with respect to the true-state

The next step is to determine the derivative of $\mathbf{p}_c^f$ with respect to the true state. Expressing the feature vector in the camera frame with the state:

$$\frac{\partial \mathbf{p}_c^f}{\partial \mathbf{x}_t} = \frac{\partial \mathbf{T}_{CB}\mathbf{R}\{\mathbf{q}_{BN}\}(\mathbf{p}_n^f - \mathbf{p}_n^b)}{\partial \mathbf{x}_t}, \tag{3.22}$$

$\mathbf{p}_n^b$ and $\mathbf{q}_{NB}$ are elements of the state-vector, just the ease notation is omitted. It can be seen that there is no other state parameter in the expression, therefore

further derivatives result in $\mathbf{0}_{3\times3}$ matrices. The derivative with respect to $\mathbf{p}_n^b$ is:

$$\frac{\partial \mathbf{p}_c^f}{\partial \mathbf{p}_n^b} = -\mathbf{T}_{CB}\mathbf{R}\{\mathbf{q}_{BN}\} \tag{3.23}$$

Calculating the derivative by the quaternion is tricky. Initially, the quaternion rotation formula needs to be employed on (3.22), then the derivative should be decomposed using the chain rule into separate components: one with respect to the vector and the other with respect to the quaternion:

$$\begin{aligned}\frac{\partial \mathbf{p}_c^f}{\partial \mathbf{q}_{NB}} &= \left.\frac{\partial \left(\mathbf{q}_{CB}\otimes\mathbf{q}_{NB}\otimes(\mathbf{p}_n^f-\mathbf{p}_n^b)\otimes\mathbf{q}_{NB}^*\otimes\mathbf{q}_{CB}^*\right)}{\partial \mathbf{q}_n}\right|_{\mathbf{a}=\mathbf{p}_n^f-\mathbf{p}_n^b} \\ &= \frac{\partial \left(\mathbf{q}_{CB}\otimes\mathbf{q}_{NB}\otimes\mathbf{a}\otimes\mathbf{q}_{NB}^*\otimes\mathbf{q}_{CB}^*\right)}{\partial \left(\mathbf{q}_{NB}\otimes\mathbf{a}\otimes\mathbf{q}_{NB}^*\right)}\frac{\partial \left(\mathbf{q}_{NB}\otimes\mathbf{a}\otimes\mathbf{q}_{NB}^*\right)}{\partial \mathbf{q}_{NB}}\end{aligned} \tag{3.24}$$

Using partial results for the two Jacobian above from Appendices A.3 and A.4, the outcome is:

$$\frac{\partial \mathbf{p}_c^f}{\partial \mathbf{q}_{NB}} = 2\mathbf{T}_{CB}\left[q_w\mathbf{a}+\mathbf{q}_v\times\mathbf{a} \mid \mathbf{q}_v^T\mathbf{a}\mathbf{I}_{3\times3}+\mathbf{q}_v\mathbf{a}^T-\mathbf{a}\mathbf{q}_v^T-q_w\left[\mathbf{a}\right]_\times\right] \tag{3.25}$$

The whole derivative by the true state results in a 3×16 matrix:

$$\mathbf{P}_{\mathbf{x}_t} = \begin{bmatrix}\frac{\partial \mathbf{p}_c^f}{\partial \mathbf{p}_n^b} & \frac{\partial \mathbf{p}_c^f}{\partial \mathbf{q}_n} & \mathbf{0}_{3\times9}\end{bmatrix} \tag{3.26}$$

Jacobian of the true-state with respect to the error-state

Finally, the Jacobian of the true state with respect to the error state should be determined. All derivatives yield the identity block $\mathbf{I}_3$, except for the quaternion, this can be understood by examining the composition of individual states in Table 3.1. Looking to the Jacobian of quaternion $\mathbf{Q}_{\delta\theta}\in\mathbb{R}^{4\times3}$:

$$\frac{\partial(\delta\mathbf{q}\otimes\mathbf{q})}{\partial\delta\boldsymbol{\theta}} = \mathbf{Q}_{\delta\theta} = \frac{1}{2}\left[\mathbf{q}\right]_R\begin{bmatrix}\mathbf{0}^T\\ \mathbf{I}_3\end{bmatrix} \tag{3.27}$$

Detailed calculations can be found in Appendix B.3, which leads to the Jacobian:

$$\frac{\partial \mathbf{x}_t}{\partial \delta \mathbf{x}} = \mathbf{X}_{\delta x} = \begin{bmatrix} \frac{\partial \mathbf{p}_{n,t}}{\partial \delta \mathbf{p}_n} & \dots & \mathbf{0}_{3\times 3} \\ \vdots & \ddots & \vdots \\ \mathbf{0}_{3\times 3} & \dots & \frac{\partial \boldsymbol{\beta}_{\omega,t}}{\partial \delta \boldsymbol{\beta}_\omega} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{4\times 3} & \mathbf{Q}_{\delta\theta} & \mathbf{0}_{4\times 9} \\ \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{I}_9 \end{bmatrix} \tag{3.28}$$

3.4.3 Error injection into the nominal-state

After calculating the measurement matrix, the Kalman gain ($\mathbf{K}$) should be determined based on errors in the state, the measurement, and the feature estimation. The last one should be considered if it's not known prior, therefore I will detail the Kalman gain calculation later because it's a complex step. The Kalman gain and the residual ($\mathbf{r}$) can be utilized to calculate the error state:

$$\delta \hat{\mathbf{x}} = \mathbf{K}\mathbf{r} \quad \mathbf{K} \in \mathbb{R}^{15\times 2k}, \quad \mathbf{r} \in \mathbb{R}^{2k\times 1}, \tag{3.29}$$

where $\delta\hat{\mathbf{x}}$ is the mean of the error state, and k is the number of features involved in the update.

The mean has to be injected into the nominal state, this operation requires composition from Table 3.1. All quantities require a simple sum of the nominal and error state, except for the rotation which can be performed as a left-side quaternion multiplication, therefore the injection procedure:

$$\mathbf{p} = \mathbf{p} + \delta\hat{\mathbf{p}} \tag{3.30a}$$

$$\mathbf{q} = \mathbf{q}\{\delta\hat{\boldsymbol{\theta}}\} \otimes \mathbf{q} \tag{3.30b}$$

$$\mathbf{v}_b = \mathbf{v}_b + \delta\hat{\mathbf{v}}_b \tag{3.30c}$$

$$\boldsymbol{\beta}_a = \boldsymbol{\beta}_a + \delta\hat{\boldsymbol{\beta}}_a \tag{3.30d}$$

$$\boldsymbol{\beta}_\omega = \boldsymbol{\beta}_\omega + \delta\hat{\boldsymbol{\beta}}_\omega \tag{3.30e}$$

Next, the error state gets reset, therefore its mean has to be removed, which is done by the inverse operations above. I'm denoting this operation with $\delta\mathbf{x}^+ =$

$g(\delta\mathbf{x})$, which can be expressed as:

$$\delta\mathbf{p}^{+} = \delta\mathbf{p} - \hat{\delta\mathbf{p}} \tag{3.31a}$$

$$\delta\boldsymbol{\theta}^{+} = 2\begin{bmatrix} 0 & \mathbf{0}^{T} \\ \mathbf{0} & \mathbf{I}_3 \end{bmatrix}\left(\mathbf{q}\{\delta\boldsymbol{\theta}\} \otimes \mathbf{q}\{-\hat{\delta\boldsymbol{\theta}}\}\right) \tag{3.31b}$$

$$\delta\mathbf{v}_b^{+} = \delta\mathbf{v}_b - \hat{\delta\mathbf{v}}_b \tag{3.31c}$$

$$\delta\boldsymbol{\beta}_a^{+} = \delta\boldsymbol{\beta}_a - \hat{\delta\boldsymbol{\beta}}_a \tag{3.31d}$$

$$\delta\boldsymbol{\beta}_\omega^{+} = \delta\boldsymbol{\beta}_\omega - \hat{\delta\boldsymbol{\beta}}_\omega \tag{3.31e}$$

To update the error state, the mean has to be reset to zeros, but the covariance matrix update depends on $g(.)$, thus the full error reset:

$$\hat{\delta\mathbf{x}} = \mathbf{0} \tag{3.32a}$$

$$\mathbf{P}^{+} = \mathbf{G}\mathbf{P}\mathbf{G}^{T} \tag{3.32b}$$

Here, $\mathbf{G}$ is the Jacobian matrix of $g(.)$, defined as:

$$\mathbf{G} = \left.\frac{\partial g}{\partial \delta\mathbf{x}}\right|_{\hat{\delta\mathbf{x}}} = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{I}_3 + \left[\frac{1}{2}\hat{\delta\boldsymbol{\theta}}\right]_\times & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_9 \end{bmatrix} \tag{3.33}$$

Similarly to what happened with the update Jacobian before, all quantities result in identity blocks, except the orientation part. The calculations related to the formula above are detailed in Appendix B.4.

CHAPTER 4

OVERVIEW OF THE TRIANGULATION METHOD

In the previous chapter, the ESKF was introduced as a method capable of updating the state using camera measurements, provided by the precise localization coordinates of visible feature points are known. However, in practical applications, assuming prior knowledge of the positions of these visible feature points is often untenable. Consequently, there arises the need to estimate these vectors, a process commonly referred to as triangulation.

This chapter provides an in-depth look at the triangulation method applied in this study, with a focus on its integration into the visual-inertial navigation system. The foundation of this method draws from [23], utilizing the core equations as described therein. However, their approach only takes into account uncertainties in the camera measurements, but the current system requires consideration of uncertainties in the states too.

The central objective of this chapter is to explain the key components and modifications in the custom-made triangulation approach. In particular, answers will be given to questions such as how position and orientation are involved in the LOST equations, and how can be formed the recursive version of LOST.

4.1 Problem statement

The modern triangulation problem takes on one of two forms: intersection or resection [24], shown in the figure 4.1.

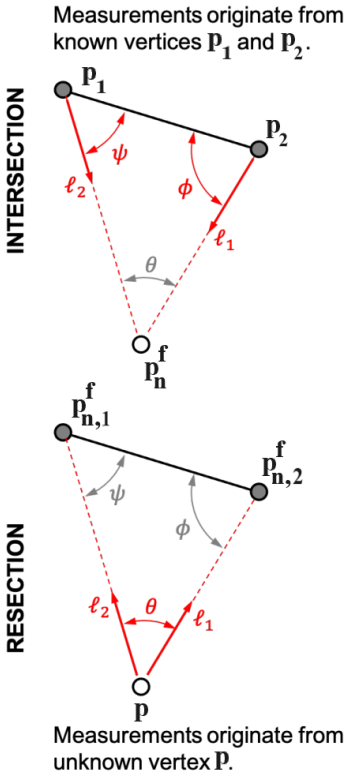


Figure 4.1: Illustration of intersection (top) and resection (bottom) forms of the triangulation problem ([23]: page 3)

The main difference between the two formalizations is that the intersection assumes measurements taken from known vertices ($\mathbf{p}_1$, $\mathbf{p}_2$). These vertices express the camera positions at each measurement in the NED frame ($\mathbf{p}_n^c$) and the goal is to determine the position of a visible feature point ($\mathbf{p}_n^f$). On the other hand, the resection problem supposes known visible vertices ($\mathbf{p}_{n,1}^f$, $\mathbf{p}_{n,2}^f$) and aims to estimate the position where the measurement was taken ($\mathbf{p}$).

The intersection problem has many practical applications such as satellite orbit determination [25, 26], and 3-D scene reconstruction usually called Structure from Motion (SfM) [27, 28, 29]. The resection problem describes the vehicle localization problem that is generally included in navigation applications [13, 14, 30]. In this project, both forms are applied, because LOST is used to optimize 3-D coordinates of visible feature points which means a 3-D reconstruction of the environment. On the contrary, the Kalman filter update is the form of a resection problem in the sense that the optimized LOST estimates are used to improve the state estimates.

4.1.1 Line of sight (LOS) measurements

The triangulation problem typically involves angles acquired through various optical instruments. In vehicle navigation, angles are often derived from images captured by cameras or telescopes. Regardless of the specific application, the angle measurements obtained through these optical instruments describe the direction from the sensor to the observed point. In other words, they express the path from one vertex of a triangle to another, and this direction represents a straight line connecting these two points, commonly known as the "line of sight" (LOS).

Before I delve into the details, it's important to establish a few mathematical notation:

- The actual measurement is $\begin{bmatrix} u_i & v_i & f \end{bmatrix}^T = \overline{\mathbf{u}}_i = \mathbf{K}\mathbf{u}_i$, but the LOST algorithm uses LOS measurement ($\mathbf{u}$, Figure 2.4) that can be derived from actual measurement as $\mathbf{u}_i = \mathbf{K}^{-1}\overline{\mathbf{u}}_i$. However, it is important to emphasize that this transformation accounts for both the adjustment of the principal point and multiplying the vector with the focal length. Later merely introduces a scaling factor, which is inherently resolved in LOST equations, therefore it is enough to compensate the principal point.

- The feature's position ($\mathbf{p}_c^f$) and the LOS measurement differ only with a scaling factor: $\mathbf{u}_i \propto \mathbf{a}_i \propto \mathbf{p}_{c,i}^f$, where $\mathbf{a}_i = \frac{\mathbf{u}_i}{||\mathbf{u}_i||} = \frac{\mathbf{p}_{\mathbf{c,i}}^{\mathbf{f}}}{||\mathbf{p}_{c,i}^f||}$

- $\mathbf{p}_{c,i}^f = \rho\mathbf{a}_i$, where ρ denotes the range from the camera center to the observed point

4.2 LOST method

When more than 2 LOS measurements are available the polynomial methods do not scale well [23], therefore the LOST suggests a Maximum Likelihood Estimation (MLE) solution for an unknown vertex ($\mathbf{p}_n^f$). The linear system can be created by double applications of the Law of Sines. Briefly, this trigonometric law relates the angles and side lengths of a triangle.

The initial equation is the Direct Linear Transform (DLT) form of the Law of Sines. This mathematical prescription removes the unknown scale ambiguity along the LOS trajectory, achieved by utilizing the collinearity of the LOS measurement ($\mathbf{u}$) and feature vector ($\mathbf{p}_c^f$). The vector of the feature can be expressed as:

$$\mathbf{p}_c^f = \mathbf{R}_{CN}(\mathbf{p}_n^f - \mathbf{p}_n^c) = \mathbf{R}_{CB}(\mathbf{R}_{BN}(\mathbf{p}_n^f - \mathbf{p}_n^b) - \mathbf{p}_b^c) \tag{4.1}$$

Then the initial equation is:

$$\left[\mathbf{u}\right]_\times \mathbf{p}_c^f = \left[\mathbf{u}\right]_\times \mathbf{R}_{CB}(\mathbf{R}_{BN}(\mathbf{p}_n^f - \mathbf{p}_n^b) - \mathbf{p}_b^c) = \mathbf{0} \tag{4.2}$$

4.2.1 Covariance calculations

(4.2) is only true if every value is ideal. When only the noisy measurements and nominal states are available, the right-hand side is no longer exactly zero:

$$\left[\mathbf{u}_i\right]_\times \mathbf{p}_c^f = \boldsymbol{\epsilon}_i \tag{4.3}$$

ϵ_i can be calculated by applying the previously introduced error models in Table 3.1 which are just an additional error in the case of body position and measurement noise, but not for the rotation. Using the notation $\mathbf{u}_t = \mathbf{u} + \delta\mathbf{u}$ for the measurement model, the cross-product results in:

$$\boldsymbol{\epsilon}_i = \left[\mathbf{u}_i\right]_\times \mathbf{R}_{CN}\delta\mathbf{p}_n^b + \left[\mathbf{u}_i\right]_\times \left[\rho\mathbf{a}_i + \mathbf{T}_{CB}\mathbf{p}_b^c\right]_\times \mathbf{T}_{CB}\delta\boldsymbol{\theta} + \rho\left[\mathbf{a}_i\right]_\times \delta\mathbf{u}_i \tag{4.4}$$

In (4.4) there was used the property $\rho\mathbf{a}_i = \mathbf{p}_c^f$, because $\mathbf{p}_c^f$ is not known apriori, but the scaling factor can be calculated with applying the Law of Sines. Let's consider the intersection problem in Figure 4.1, the traveled distance can be determined as $\mathbf{d}_{ij} = \mathbf{p}_{n,j}^c - \mathbf{p}_{n,i}^c = \mathbf{p}_{n,j}^b - \mathbf{p}_{n,i}^b$, and applying law of sines on vertex

$\mathbf{p}^b_{n,j}$ and $\mathbf{p}^f_n$:

$$\frac{||\mathbf{p}^f_{c,j}||}{\sin\phi} = \frac{||\mathbf{d}_{ij}||}{\sin\theta} \Rightarrow \rho_j = ||\mathbf{p}^f_{c,j}|| = \frac{||\mathbf{d}_{ij}||\sin\phi}{\sin\theta} = \frac{||\mathbf{d}_{ij} \times \mathbf{a}_i||}{||\mathbf{a}_i \times \mathbf{a}_j||}, \tag{4.5}$$

where the DLT form of the law of sines was given. Note that the formula above is only valid in consistent frames.

The $\boldsymbol{\epsilon_i}$ can be expressed with the $\delta\mathbf{x}$ also:

$$\boldsymbol{\epsilon}_i = \overbrace{\begin{bmatrix} \begin{bmatrix}\mathbf{u}_i\end{bmatrix}_\times \mathbf{R}_{CN} & \begin{bmatrix}\mathbf{u}_i\end{bmatrix}_\times \begin{bmatrix}\rho\mathbf{a}_i + \mathbf{T}_{CB}\mathbf{p}^c_b\end{bmatrix}_\times \mathbf{T}_{CB} & \mathbf{0}_{3\times 9} \end{bmatrix}}^{\mathbf{L}_x} \delta\mathbf{x} + \rho \begin{bmatrix}\mathbf{a}_i\end{bmatrix}_\times \delta\mathbf{u}_i \tag{4.6}$$

In this case, let's note the filter state covariance as $E\{\delta\mathbf{x}\delta\mathbf{x}^T\} = \mathbf{P}_x$. The measurement covariance is modeled with independent white Gaussian noise $E\{\delta\mathbf{u}\delta\mathbf{u}^T\} = \mathbf{R} \in \mathbb{R}^{3\times 3}$. It is crucial to highlight that the LOS and actual measurements covariance is the same if only the principal point was compensated, and it is rank-deficient because the camera provides only 2-D information, therefore, there is no uncertainty along the Z-axis. Their cross-covariance is $E\{\delta\mathbf{x}\delta\mathbf{u}^T\} = \mathbf{P}_{xr}$. First, it could seem a bit strange why a correlation between the state and measurement is defined and it's connected to the schedule of the algorithm. When a visual measurement is received the algorithm performs first an update on the state, and then uses the same measurement to optimize LOST estimations, therefore when it comes to LOST updates the measurement noise is already calculated in the state.

The a posteriori estimate of the state is formed as a linear combination of three error sources: the uncertainty in the state, the feature estimation ($\delta\mathbf{p}^f_n$), and the measurement. I will detail it later, but now just look at the result:

$$\delta\mathbf{x}_{k+1} = (\mathbf{I}_{15} - \mathbf{K}\mathbf{H}_x)\,\delta\mathbf{x}_k - \mathbf{K}\mathbf{H}_f\delta\mathbf{p}^f_n - \mathbf{K}\delta\mathbf{u}, \tag{4.7}$$

where $\mathbf{H}_f$ is the measurement Jacobian with respect to the feature. The correlation between the a posteriori state and measurement is:

$$\begin{aligned}\mathbf{P}_{xr} &= E\{\delta\mathbf{x}_{k+1}\delta\mathbf{u}^T\} \\ &= (\mathbf{I}_{15} - \mathbf{K}\cancelto{\mathbf{0}}{\mathbf{H}_x})E\{\delta\mathbf{x}\delta\mathbf{u}^T\} - \mathbf{K}\mathbf{H}_f\cancelto{\mathbf{0}}{E\{\delta\mathbf{p}_n^f\delta\mathbf{u}^T\}} - \mathbf{K}\mathbf{R}, \quad \mathbf{P}_{xr} \in \mathbb{R}^{15\times 3}\end{aligned} \tag{4.8}$$

Before the update, the measurement noise was not correlated with either the state or the feature, and it is important to emphasize if there was no update performed on the selected feature, then the measurement noise wouldn't be correlated with the state. Utilizing the results the covariance of $\boldsymbol{\epsilon}_i$ is:

$$\mathbf{P}_{\epsilon_i} = E\{\boldsymbol{\epsilon}_i\boldsymbol{\epsilon}_i^T\} = \begin{bmatrix}\mathbf{L}_x & \rho\begin{bmatrix}\mathbf{a}_i\end{bmatrix}_\times\end{bmatrix}\begin{bmatrix}\mathbf{P}_x & \mathbf{P}_{xr}\\ \mathbf{P}_{rx} & \mathbf{R}\end{bmatrix}\begin{bmatrix}\mathbf{L}_x^T \\ \rho\begin{bmatrix}\mathbf{a}_i\end{bmatrix}_\times^T\end{bmatrix} \tag{4.9}$$

4.2.2 The optimal solution

In [23], they propose the formalization of the MLE optimization problem that minimizes $\boldsymbol{\epsilon}$ by variable $\mathbf{p}_n^f$ as:

$$\min J(\mathbf{p}_n^f) = \sum_{i=1}^{n}\boldsymbol{\epsilon}_i^T\mathbf{P}_{\epsilon_i}^{-1}\boldsymbol{\epsilon}_i \tag{4.10}$$

Since $\mathbf{P}_{\epsilon_i}$ is always rank deficient due to skew-matrices involved in its calculation and also the pixel error is only 2-D, henceforth the approximation $\mathbf{P}_{\epsilon_i}^{-1} \to \mathbf{P}_{\epsilon_i}^{+}$ will be used, where $^+$ denotes the Moore-Penrose inverse [31]. Returning to (4.10), the next step is to substitute (4.2) in, then rearranging terms consisting $\mathbf{p}_n^f$ and neglecting those are not consist $\mathbf{p}_n^f$ the cost function to minimize:

$$
\begin{aligned}
\min J(\mathbf{p}_n^f) =& \mathbf{p}_n^{f^T} \sum_{i=1}^{n} \mathbf{R}_{NC,i} \left[\mathbf{u}_i\right]_\times \mathbf{P}_{\epsilon_i}^{+} \left[\mathbf{u}_i\right]_\times \mathbf{R}_{CB}(\mathbf{R}_{BN,i}\mathbf{p}_{n,i}^b + \mathbf{p}_b^c) \\
&+ \left(\sum_{i=1}^{n} (\mathbf{R}_{BN,i}\mathbf{p}_{n,i}^b + \mathbf{p}_b^c)^T \mathbf{R}_{BC} \left[\mathbf{u}_i\right]_\times \mathbf{P}_{\epsilon_i}^{+} \left[\mathbf{u}_i\right]_\times \mathbf{R}_{CN,i} \right) \mathbf{p}_n^f \\
&- \mathbf{p}_n^{f^T} \left(\sum_{i=1}^{n} \mathbf{R}_{NC,i} \mathbf{P}_{\epsilon_i}^{+} \mathbf{R}_{CN,i} \right) \mathbf{p}_n^f
\end{aligned}
\tag{4.11}
$$

Minimisation can be performed applying the 1^{st} differential condition on (4.11) that yields:

$$
\begin{aligned}
&\frac{\partial \min J(\mathbf{p}_n^f)}{\partial \mathbf{p}_n^f} = \\
&-2 \sum_{i=1}^{n} \mathbf{R}_{NC,i} \left[\mathbf{u}_i\right]_\times \mathbf{P}_{\epsilon_i}^{+} \left[\mathbf{u}_i\right]_\times \mathbf{R}_{CB}(\mathbf{R}_{BN,i}(\mathbf{p}_n^f - \mathbf{p}_{n,i}^b) - \mathbf{p}_b^c) = 0
\end{aligned}
\tag{4.12}
$$

Rearranging the terms, the following linear equation system has to be solved:

$$
\begin{aligned}
&\left(\sum_{i=1}^{n} \mathbf{R}_{NC,i} \left[\mathbf{u}_i\right]_\times \mathbf{P}_{\epsilon_i}^{+} \left[\mathbf{u}_i\right]_\times \mathbf{R}_{CN,i} \right) \mathbf{p}_n^f = \\
&\sum_{i=1}^{n} \mathbf{R}_{NC,i} \left[\mathbf{u}_i\right]_\times \mathbf{P}_{\epsilon_i}^{+} \left[\mathbf{u}_i\right]_\times \mathbf{R}_{CB}(\mathbf{R}_{BN,i}\mathbf{p}_{n,i}^b + \mathbf{p}_b^c)
\end{aligned}
\tag{4.13}
$$

4.2.3 Practical formalization of the estimator

The usual approach for solving least squares systems is to avoid the explicit formalization of the normal equations, and instead solve (4.13) with an alternative technique such as solving it through factorization [32].

Since the weighting matrix $\mathbf{P}_{\epsilon_i}^{+}$ is calculated from a covariance matrix which is positive semi-definite, therefore it has real non-negative eigenvalues and real eigenvectors. With the help of eigendecomposition [33], $\mathbf{P}_{\epsilon_i}^{+}$ can be factorized as $\mathbf{P}_{\epsilon_i}^{+} = \mathbf{V}_i \mathbf{D}_i \mathbf{V}_i^T = (\mathbf{V}_i \sqrt{\mathbf{D}_i})(\mathbf{V}_i \sqrt{\mathbf{D}_i})^T = \mathbf{B}_i \mathbf{B}_i^T$.

Decomposing (4.13) with notation $\mathbf{A}_i = \mathbf{B}_i^T \left[\mathbf{u}_i\right]_\times \mathbf{R}_{CN,i}$ it yields:

$$\overbrace{\begin{bmatrix}\mathbf{A}_1\\ \vdots \\ \mathbf{A}_n\end{bmatrix}}^{\mathbf{A}} \mathbf{p}_n^f = \overbrace{\begin{bmatrix}\mathbf{A}_1 \overbrace{(\mathbf{p}_{n,1}^b + \mathbf{R}_{NB,1}\mathbf{p}_b^c)}^{\mathbf{y}_1}\\ \vdots \\ \mathbf{A}_n \overbrace{(\mathbf{p}_{n,n}^b + \mathbf{R}_{NB,n}\mathbf{p}_b^c)}^{\mathbf{y}_n}\end{bmatrix}}^{\mathbf{b}} \tag{4.14}$$

The above equation must be solved in least squares terms, which means the estimator is calculated as:

$$\hat{\mathbf{p}}_n^f = \mathbf{A}^+\mathbf{b} \tag{4.15}$$

4.2.4 Covariance of the estimator

As a result of statistically optimal weighting the estimator covariance can be computed as:

$$\mathbf{P}_f = \left(\sum_{i=1}^{n} \mathbf{A}_i^T\mathbf{A}_i\right)^{-1} \tag{4.16}$$

It is worth mentioning that $\mathbf{A}_i^T\mathbf{A}_i$ matrices are rank deficient, therefore the inverse calculation accuracy can degrade especially when the number of samples is low, thus the application of pseudoinverse is recommended.

4.3 The recursive version

An additional advantage of decomposing the linear system is that the estimator takes the form of a general LS solution, for which the recursive version is easy to find. Initially, let's formalize the estimation until the i^{th} estimation based on (4.14) and (4.15):

$$\hat{\mathbf{p}}_{n,i}^f = \left(\sum_i \mathbf{A}_i^T\mathbf{A}_i\right)^{-1}\left(\sum_i \mathbf{A}_i^T\mathbf{y}_i\right) = \mathbf{P}_{f,i}\mathbf{b}_i, \tag{4.17}$$

note that $\mathbf{b}_i = \mathbf{P}_{f,i}^{-1}\hat{\mathbf{p}}_{n,i}^{f}$. Then the $(i+1)^{th}$ estimation is:

$$\begin{aligned}
\hat{\mathbf{p}}_{n,i+1}^{f} &= \mathbf{P}_{f,i+1}\left(\mathbf{b}_i + \mathbf{A}_{i+1}^{T}\mathbf{y}_{i+1}\right) \\
&= \mathbf{P}_{f,i+1}\left(\mathbf{P}_{f,i}^{-1}\hat{\mathbf{p}}_{n,i}^{f} + \mathbf{A}_{i+1}^{T}\mathbf{y}_{i+1}\right) \\
&= \mathbf{P}_{f,i+1}\left(\left(\mathbf{P}_{f,i+1}^{-1} - \mathbf{A}_{i+1}^{T}\mathbf{A}_{i+1}\right)\hat{\mathbf{p}}_{n,i}^{f} + \mathbf{A}_{i+1}^{T}\mathbf{y}_{i+1}\right) \\
&= \hat{\mathbf{p}}_{n,i}^{f} + \mathbf{P}_{f,i+1}\mathbf{A}_{i+1}^{T}\left(\mathbf{y}_{i+1} - \mathbf{A}_{i+1}\hat{\mathbf{p}}_{n,i}^{f}\right),
\end{aligned} \tag{4.18}$$

where $\mathbf{A}_{i+1}$ and $\mathbf{y}_{i+1}$ can be calculated from new data, and the covariance matrix can be propagated based on (4.16):

$$\mathbf{P}_{f,i+1} = \left((\mathbf{P}_{f,i})^{-1} + \mathbf{A}_{i+1}^{T}\mathbf{A}_{i+1}\right)^{-1} \tag{4.19}$$

CHAPTER 5

SYNERGIZING LOST AND ESKF FOR ROBUST NAVIGATION

In the previous two chapters, the applied filter technique and triangulation approach were introduced, and now the focus shifts toward the integration of the two methods. This integration process will begin by examining the impact on the Kalman gain, a crucial component in the navigation system. Furthermore, the discussion extends to the role of cross-covariances in the process of integration.

5.1 The modified Kalman gain

The derivation of the Kalman gain is based on [22], but adopted to the implemented system. Firstly, the integration affects the Kalman gain because forming the residual involves both the triangulated feature position and the filter state. Previously, only the uncertainty of the state was considered, but as soon as the true values of the 3-D feature position are not known, then its uncertainty should be taken into account too.

5.1.1 The error propagation from measurement to measurement

Taking a look at the error model on (3.17) and the propagation equations on (3.19b) and (3.19c), the error state is propagated from a posterior state to the next update as:

$$\begin{aligned}\delta\mathbf{x}_{k+N}^{-} &= \left(\prod_{j=0}^{N-1}\mathbf{F}_{x,k+j}\right)\delta\mathbf{x}_{k}^{-} + \left[\sum_{j=0}^{N-1}\left(\prod_{l=j+1}^{N-1}\mathbf{F}_{x,k+l}\right)\mathbf{F}_{i,k+j}\mathbf{i}_{k+j}\right] \\ &= \mathbf{P}\mathbf{F}_{x}^{N}\delta\mathbf{x}_{k}^{-} + \mathbf{S}\mathbf{F}_{i}^{N}\mathbf{i}\end{aligned} \tag{5.1}$$

The significant insight gleaned from the aforementioned equations is that error propagation can be divided into two components. The first pertains to the propagation of the mean error state, denoted as $\mathbf{PF}_x^N \delta\mathbf{x}_k$, while the second is associated with accounting for the integration of Additive White Gaussian Noise (AWGN) within the system, denoted as $\mathbf{SF}_i^N \mathbf{i}$.

5.1.2 The measurement model

The actual measurement is produced by h(.) non-linear function and the true value of the state and feature that leads to:

$$\mathbf{z}_{k+N} = h(\mathbf{x}_{t,k+N}, \mathbf{p}_{n,t}^f) \tag{5.2}$$

To form the residual, the measurements have to be predicted and it's done based on the following mathematical model:

$$\hat{\mathbf{z}}_{k+N} = h(\mathbf{x}_{k+N}, \mathbf{p}_n^f) + \mathbf{H}_{x,k+N}\delta\mathbf{x}_{k+N} + \mathbf{H}_{f,k+N}\delta\mathbf{p}_n^f + \delta\mathbf{u}_{k+N}, \tag{5.3}$$

where one part relates to the non-linear projection h(.) onto the nominal state and another is connected to the linearised error models. $\mathbf{H}_{f,k+N}$ is the pixel measurement Jacobian by the feature that can be calculated by taking the derivative of (4.1), that yields:

$$\frac{\partial \mathbf{p}_c^f}{\partial \mathbf{p}_n^f} = \mathbf{R}_{CN} \tag{5.4}$$

In this case, the Jacobian is the same for the error state and true state due to the true feature being modeled as $\mathbf{p}_{n,t}^f = \mathbf{p}_n^f + \delta\mathbf{p}_n^f$.

The Kalman filter formalizes the new estimation as a composition of the a posteriori estimate $\delta\mathbf{x}_{k+N}^-$ and the residual $\mathbf{r}_{k+N} = \mathbf{z}_{k+N} - \hat{\mathbf{z}}_{k+N}$:

$$\begin{aligned}
\delta\mathbf{x}_{k+N} &= \delta\mathbf{x}_{k+N}^- + \mathbf{K}_{k+N}\mathbf{r}_{k+N} \\
&= \delta\mathbf{x}_{k+N}^- - \mathbf{K}_{k+N}(\mathbf{H}_{x,k+N}\delta\mathbf{x}_{k+N}^- + \mathbf{H}_{f,k+N}\delta\mathbf{p}_n^f + \delta\mathbf{u}_{k+N}) \\
&= \overbrace{(\mathbf{I}_{15} - \mathbf{K}_{k+N}\mathbf{H}_{x,k+N})}^{\mathbf{T}_x}\delta\mathbf{x}_{k+N}^- - \overbrace{\mathbf{K}_{k+N}\mathbf{H}_{f,k+N}}^{\mathbf{T}_f}\delta\mathbf{p}_n^f - \mathbf{K}_{k+N}\delta\mathbf{u}_{k+N}
\end{aligned} \tag{5.5}$$

5.1.3 The optimal solution for Kalman gain

The propagated state covariance will denoted as $E\{\delta\mathbf{x}^-_{k+N}\delta\mathbf{x}^{-^T}_{k+N}\} = \mathbf{P}^-_{x,k+N}$, the feature covariance is $E\{\delta\mathbf{p}^f_n\delta\mathbf{p}^{f^T}_n\} = \mathbf{P}_f$, the measurement covariance is still $\mathbf{R}$. To give an optimal solution for the Kalman gain the a posteriori covariance should be expressed, but first, let's take some considerations. When a visual measurement was taken, the current approach updates the state first, then optimizes the feature with the same measurement which leads to a correlation between $\delta\mathbf{x}$ and $\delta\mathbf{p}^f_n$, that can be expressed with cross-covariance matrix $E\{\delta\mathbf{x}_{k+N}\delta\mathbf{p}^{f^T}_n\} = \mathbf{P}_{xf}$. At this point, I leave indexing because it is clear to which measurement it refers. The a posteriori covariance:

$$\begin{aligned}\mathbf{P}_x &= E\{\delta\mathbf{x}\delta\mathbf{x}^T\} \\ &= \mathbf{T}_x\mathbf{P}^-_x\mathbf{T}^T_x + \mathbf{T}_f\mathbf{P}_f\mathbf{T}^T_f + \mathbf{K}\mathbf{R}\mathbf{K}^T - \mathbf{T}_x\mathbf{P}_{xf}\mathbf{T}^T_f - \mathbf{T}_f\mathbf{P}_{fx}\mathbf{T}^T_x\end{aligned} \tag{5.6}$$

Now, the optimal Kalman gain should be calculated, in the sense that it minimizes the trace of the posterior covariance. For starters, let's formalize the a posteriori covariance by extracting the terms containing $\mathbf{K}$:

$$\begin{aligned}\mathbf{P}_x =& \mathbf{P}^-_x - \mathbf{K}\left(\mathbf{H}_x\mathbf{P}^-_x + \mathbf{H}_f\mathbf{P}_{fx}\right) - \overbrace{\left(\mathbf{P}^-_x\mathbf{H}^T_x + \mathbf{P}_{xf}\mathbf{H}^T_f\right)}^{E}\mathbf{K}^T \\ &+ \mathbf{K}\underbrace{\left(\mathbf{H}_x\mathbf{P}^-_x\mathbf{H}^T_x + \mathbf{H}_f\mathbf{P}_f\mathbf{H}^T_f + \mathbf{R} + \mathbf{H}_x\mathbf{P}_{xf}\mathbf{H}^T_f + \mathbf{H}_f\mathbf{P}_{fx}\mathbf{H}^T_x\right)}_{\mathbf{D}}\mathbf{K}^T\end{aligned} \tag{5.7}$$

Important to highlight the property $\mathbf{E} = \mathbf{E}^T$. Finally, the optimal Kalman gain can be computed from the above equation by taking the derivative by $\mathbf{K}$ and solving it for 0.

$$\frac{\partial \mathrm{tr}(\mathbf{P}_x)}{\partial \mathbf{K}} = -\mathbf{E}^T - \mathbf{E} + 2\mathbf{K}\mathbf{D} = 0 \tag{5.8}$$

Rearranging the terms for $\mathbf{K}$:

$$\mathbf{K} = \mathbf{E}\mathbf{D}^{-1} \tag{5.9}$$

Details about the result of the derivative can be found in [22].

5.2 Cross-covariance estimation

The method for cross-covariance modeling is utilized from [34], called Gaussian cosimulation. This method foundation is that a p-dimensional vector $\mathbf{y}$ is multivariate normal with rank m, mean $\boldsymbol{\mu}$ and covariance $\mathbf{P}$, then:

$$\mathbf{y} = \boldsymbol{\mu} + \mathbf{L}\mathbf{z}, \quad \mathbf{P} = \mathbf{L}\mathbf{L}^T, \tag{5.10}$$

where $\mathbf{L} \in \mathbb{R}^{p \times m}$ matrix of rank m and $\mathbf{z}$ is a vector m independent identically distributed standard normal variable [35]. Both [36, 37] propose the usage of the LU decomposition for conditional simulations, which results in simulating Gaussian random fields. In my work, I employ a similar approach by calculating the square root decomposition of covariance matrices to efficiently create cross-covariance models, maintaining simplicity, flexibility, and computational efficiency in the process.

The method introduces two correlated random Gaussian vectors $\mathbf{y}_1$ and $\mathbf{y}_2$ with ρ correlation:

$$\begin{aligned} \mathbf{y}_1 &= \boldsymbol{\mu}_1 + \mathbf{L}_1\mathbf{z}_1 \\ \mathbf{y}_2 &= \boldsymbol{\mu}_2 + \mathbf{L}_2\left(\rho\mathbf{z}_1 + \sqrt{1-\rho^2}\mathbf{z}_2\right) \end{aligned} \tag{5.11}$$

The covariance matrices of the vectors are $E\{(\mathbf{y}_1 - \boldsymbol{\mu}_1)(\mathbf{y}_1 - \boldsymbol{\mu}_1)^T\} = \mathbf{L}_1\mathbf{L}_1^T$ and $E\{(\mathbf{y}_2 - \boldsymbol{\mu}_2)(\mathbf{y}_2 - \boldsymbol{\mu}_2)^T\} = \mathbf{L}_2\mathbf{L}_2^T$, that utilizes $\mathbf{z}_1$ and $\mathbf{z}_2$ are zero-mean standard Gaussian distributed vectors, therefore $E\{\mathbf{z}_1\mathbf{z}_1^T\}, E\{\mathbf{z}_2\mathbf{z}_2^T\} = \mathbf{I}$ and $E\{\mathbf{z}_1\mathbf{z}_2^T\}, E\{\mathbf{z}_2\mathbf{z}_1^T\} = \mathbf{0}$. Then the cross-covariance:

$$\begin{aligned} E\{(\mathbf{y}_1 - \boldsymbol{\mu}_1)(\mathbf{y}_2 - \boldsymbol{\mu}_2)^T\} &= E\{\mathbf{L}_1\mathbf{z}_1(\rho\mathbf{z}_1^T\mathbf{L}_2^T + \sqrt{1-\rho^2}\mathbf{z}_2^T\mathbf{L}_2^T)\} \\ &= \rho\mathbf{L}_1E\{\mathbf{z}_1\mathbf{z}_1^T\}\mathbf{L}_2^T + \sqrt{1-\rho^2}\mathbf{L}_1E\{\mathbf{z}_1\mathbf{z}_2^T\}\mathbf{L}_2^T \\ &= \rho\mathbf{L}_1\mathbf{L}_2 \end{aligned} \tag{5.12}$$

Now, $\rho \in [0;1]$ is a tuneable parameter that determines the strength of the correlation between Gaussian distributed vectors, but $\mathbf{L}_1$ and $\mathbf{L}_2$ should be determined

as the square root of the actual covariance matrices. It can be always done because covariance matrices are positive semi-definite. The procedure is the same as in (4.14) where the LOST equations were square root decomposed.

The state and the feature covariances are defined as Gaussian fields with different dimensions, therefore the covariance decomposition is applied to covariances existing in the pixel space, and the decomposed covariance matrices are $\mathbf{H}_x\mathbf{P}_x^-\mathbf{H}_x^T$ and $\mathbf{H}_f\mathbf{P}_f\mathbf{H}_f^T$.

CHAPTER 6

DEVELOPMENT

In this chapter, I delve into the intricate construction of the simulation environment, unveiling the tools employed and the nuanced calibration of parameters essential to its functionality. The selected tools will be introduced that serve as the bedrock for the experimental framework. This section sheds light on the nuanced decisions made to improve the simulation's fidelity and optimize the filter's performance.

6.1 Simulation environment

The initial simulation was developed by SZTAKI System Control Laboratory (SCL) using Matlab/Simulink under the R2019b version, which formed the foundation for subsequent development. The simulation incorporates various toolboxes, including aerospace, UAV, navigation, and real-time Simulink, among others, to enhance its realism. The aircraft model utilizes the parameters of Sindy[1], while the environment is represented using the WGS84 convention.

During the development, I had to integrate a filter into the simulation, which is mainly based on a user-defined Matlab function block. This block gets the true state of the aircraft as input and uses this data to project feature points on a camera screen. It also runs an ESKF at 50 Hz, and the new visual measurements are available at 10 Hz and generated from coordinates based on pinhole equations.

[1] http://uav.sztaki.hu/sindy/home.html

6.2 Noise calibration

The system required setting five different noise values. Four of them are connected to process noise that was defined in (3.12), and one of them models the visual measurement error σ_u. The measurement noises σ_{η_ω} and σ_{η_a} are usually given as rate density in IMU datasheets, which is a sampling frequency-dependent quantity, therefore, their unit of measurement is $\frac{^\circ}{\mathrm{s}\sqrt{\mathrm{Hz}}}$ and $\frac{\mu\mathrm{g}}{\sqrt{\mathrm{Hz}}}$, where $1\mu\mathrm{g} = 9.81\frac{\mathrm{m}}{\mathrm{s}^2} \cdot 10^{-6}$. The change of biases over time are temperature-dependent parameters, but the current solution doesn't take into account this impact, therefore their value was chosen for a small value. The exact dispersion parameters are presented in Table 6.1 and were calculated for 50Hz sampling frequency.

Name	Notation	Value	UoM
Angular velocity measurement noise	σ_{η_ω}	$0.1 \cdot \sqrt{50} \approx 0.707$	$\frac{^\circ}{\mathrm{s}}$
Acceleration measurement noise	σ_{η_a}	$1 \cdot \sqrt{50} = 7.07$	mg
Accelerometer bias variation	$\sigma_{\eta_{\beta a}}$	10^{-6}	$\frac{\mathrm{m}}{\mathrm{s}^2\sqrt{\mathrm{Hz}}}$
Gyroscope bias variation	$\sigma_{\eta_{\beta\omega}}$	10^{-7}	$\frac{\mathrm{rad}}{\mathrm{s}\sqrt{\mathrm{Hz}}}$
Camera measurement noise	σ_u	0.7	px

Table 6.1: Noise summary

6.3 Camera configuration

For simulation purposes, I utilized the intrinsic parameters of a Basler camera, and its parameters are presented in table 6.2.

Name	Notation	Value [px]
Focal length	f	1177.5
Principal point X	p_x	1024
Principal point Y	p_y	768
Image width	W	2048
Image height	H	1536

Table 6.2: Camera intrinsic parameters

6.3.1 Feature point selection

The feature point selection is a crucial part of the algorithm and may have the most potential to improve. It is not exactly just a camera configuration because

this impacts the whole algorithm's performance and schedule. The feature points have to be chosen carefully because a good choice can maintain the stability of the system for a longer period.

In [38], it was proposed in VIO systems choosing features from the edge of the image can be beneficial, since the pixel coordinates are moving the most near the edges. That phenomenon is related to the optical flow equations that describe the derivatives of the pixel coordinates. In [39], the optical flow equations were presented in (1) and it shows that the derivative of image coordinates depends on the image coordinates either at the first or higher order, thus the values of the derivatives increase as the image coordinates increase. Important to emphasize in this context we talk about the principal point compensated measurements, therefore the (0,0) image coordinate is near to the center of the image.

In [7], they claim their monocular VIO system performance degrades when the aircraft flies straight and level due to velocity being less observable for a monocular VIO, and it could also relate to the fact during straight and level flight the tracked features move less on the image plane.

To sum up, it has the beneficial properties of choosing features from the edge of the image, but the current algorithm combines the results of two estimators (EKSF and LOST) therefore it's very important to keep estimations near to their real value. If features are visible for only a short period the LOST algorithm is not able to produce high-quality estimates which leads to fewer updates on the state, and we are caught in a vicious circle. In the current solution, I select feature points in the area inward from the edge of the image for example in Figure 6.1.

When the algorithm starts it selects 18 features equally distributed from the inward area. After initialization, if a tracked feature leaves the image, a new one will be selected so that it is furthest away from the other tracks. It's advantageous because it provides equal distribution concerning the image, and the independent measurement noise assumption is more accurate. Furthermore, this approach keeps a configurable zone inward from the image edges therefore the features are visible for a longer period.

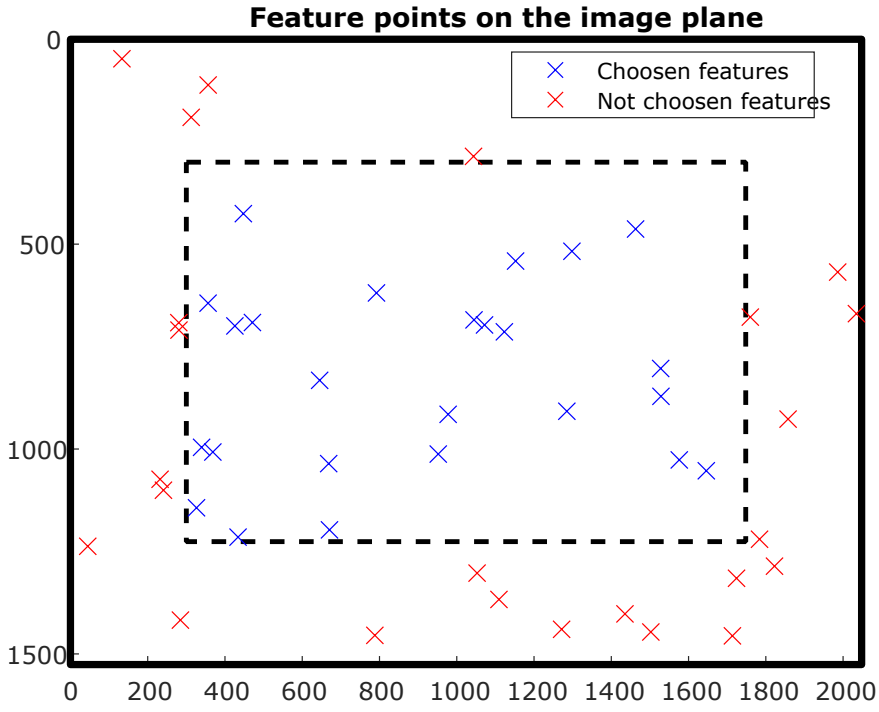


Figure 6.1: Camera image example

6.4 Simulation results

Once the parameters have been configured, only the flight trajectory needs to be defined which is described with waypoints that the aircraft must pass through. The similarity in all cases is that the LOST algorithm uses true values of the camera state to estimate feature positions until 30s, from the 30s the true state is not available and LOST starts to utilize ESKF state for estimation. The currently optimized feature is included in the ESKF update if the square root of its largest eigenvalue is less than 10m with the 3-σ rule. This rule states that the value of a Gaussian distributed probabilistic variable is in the interval $[\mu - 3\sigma; \mu + 3\sigma]$ with a 99.7% confidence level.

6.4.1 Straight and level flight path

The actual and estimated flight trajectory considering only 2-D position is shown in Figure 6.2 and related estimations below.

Summarizing, the charts show that

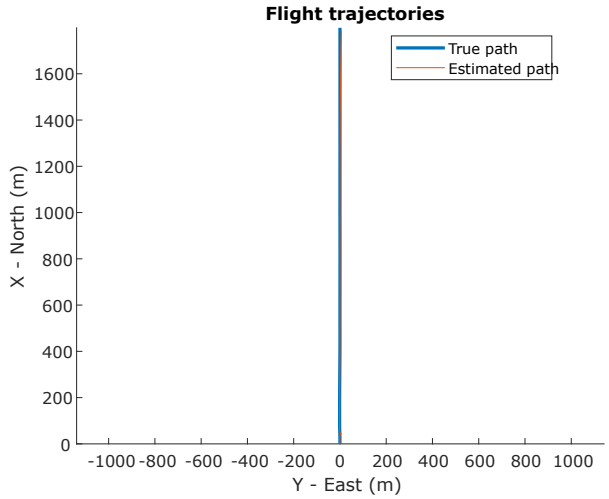
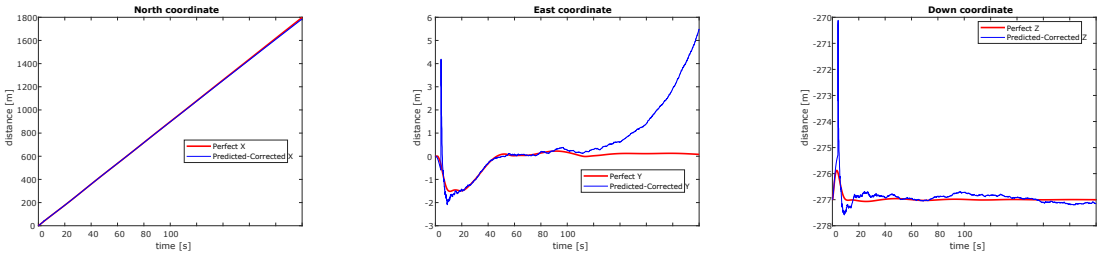
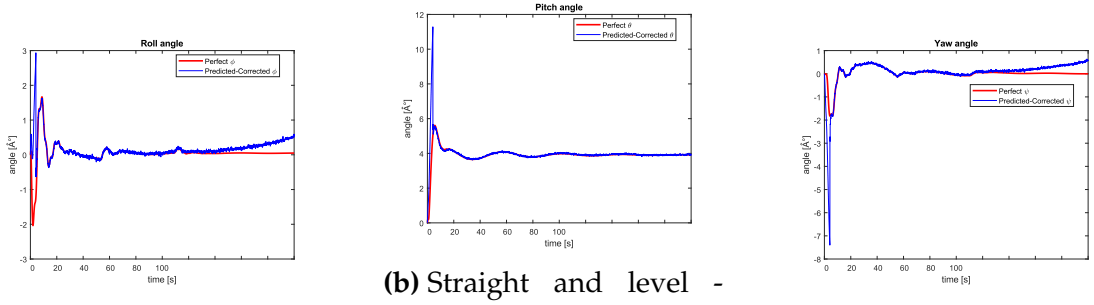


Figure 6.2: Straight and level flight path



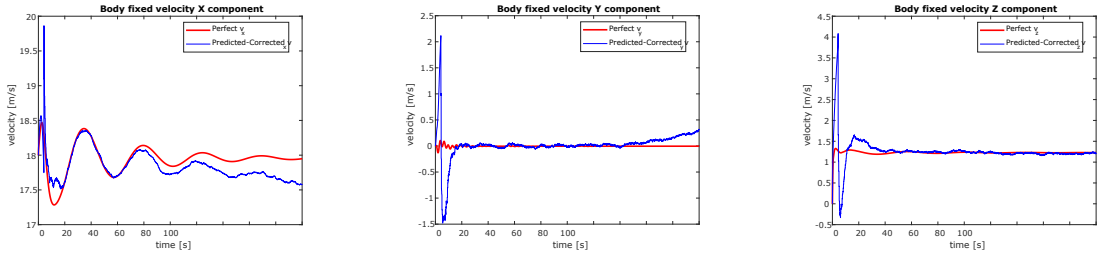
(a) Straight and level - X **(b)** Straight and level - Y **(c)** Straight and level - Z

Figure 6.3: Straight and level - position



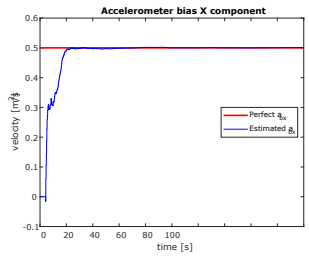
(a) Straight and level - roll **(b)** Straight and level - pitch **(c)** Straight and level - yaw

Figure 6.4: Straight and level - orientation

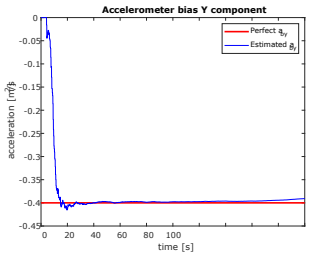


(a) Straight and level - $v_{b,x}$ **(b)** Straight and level - $v_{b,y}$ **(c)** Straight and level - $v_{b,z}$

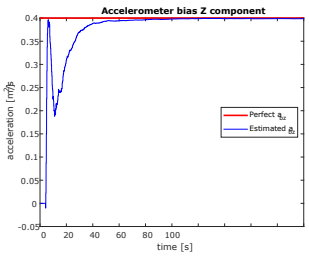
Figure 6.5: Straight and level - velocity



(a) Straight and level - $a_{b,x}$

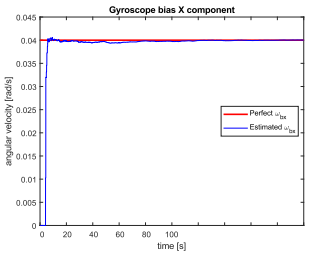


(b) Straight and level - $a_{b,y}$

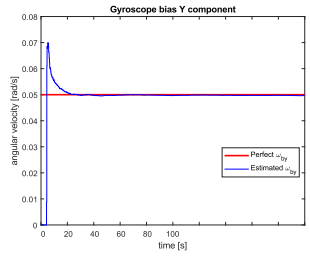


(c) Straight and level - $a_{b,z}$

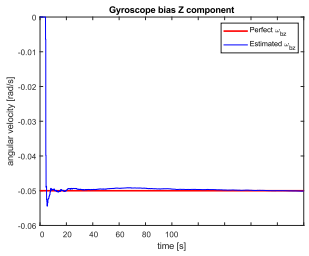
Figure 6.6: Straight and level - acceleration bias



(a) Straight and level - $\omega_{b,x}$



(b) Straight and level - $\omega_{b,y}$



(c) Straight and level - $\omega_{b,z}$

Figure 6.7: Straight and level - angular velocity bias

6.4.2 Straight and descending flight path

6.4.3 Zig-zag flight path

CHAPTER 7

CONCLUSION

In summary, the integration of ESKF and LOST with assuming known GPS coordinates in the triangulation method can be regarded as a success. However, further advancements are necessary to ensure its practical viability in real-world scenarios. These enhancements encompass the following:

1. Creating a useful cross-covariance estimator method and integrating it into the filter framework.
2. The integration of genuine image processing algorithms into the ESKF framework, leading to:
   (a) The incorporation of camera imperfections, such as lens distortion errors into the simulation.
   (b) The enhancement of the measurement update process to rely not solely on a single camera image but also on track information pertaining to feature points.
3. The development of a back-end component for the filter.

APPENDIX A

QUATERNION OPERATIONS

All of the following formulas can be found in [21].

The Hamiltonian-quaternion multiplication ($\otimes$) is defined as:

$$\mathbf{q} \otimes \mathbf{p} \triangleq \begin{bmatrix} p_w q_w - p_x q_x - p_y q_y - p_z q_z \\ p_w q_x + p_x q_w + p_y q_z - p_z q_y \\ p_w q_y - p_x q_z + p_y q_w + p_z q_x \\ p_w q_z + p_x q_y - p_y q_x + p_z q_w \end{bmatrix} = \begin{bmatrix} p_w q_w - \mathbf{p}_v^T \mathbf{q}_v \\ p_w \mathbf{q}_v + q_w \mathbf{p}_v + \mathbf{p}_v \times \mathbf{q}_v \end{bmatrix} \tag{A.1}$$

The conjugate of a $\mathbf{q}$ quaternion is defined as:

$$\mathbf{q}^* \triangleq q_w - \mathbf{q}_v = \begin{bmatrix} q_w \\ -\mathbf{q}_v \end{bmatrix} \tag{A.2}$$

The inverse of $\mathbf{q}$ quaternion is defined as:

$$\mathbf{q}^{-1} = \frac{\mathbf{q}^*}{||\mathbf{q}||^2} \Rightarrow \forall \mathbf{q}, (||\mathbf{q}|| = 1) \rightarrow \mathbf{q}^* = \mathbf{q}^{-1}, \tag{A.3}$$

where the equivalence of the conjugate and the inverse of unit quaternions is similar to the unitary property of the rotation matrices, which leads to the equivalence of the transpose and inverse matrix.

The next important property of quaternions is that the quaternion product can be expressed in matrix form:

$$\mathbf{q}_1 \otimes \mathbf{q}_2 = \Big[\mathbf{q}_1\Big]_L \mathbf{q}_2 \iff \mathbf{q}_1 \otimes \mathbf{q}_2 = \Big[\mathbf{q}_2\Big]_R \mathbf{q}_1, \tag{A.4}$$

where $\left[\mathbf{q}_1\right]_L$ and $\left[\mathbf{q}_2\right]_R$ are the left- and right- quaternion-product matrices, which can be derived from (A.1) and (A.4):

$$\left[\mathbf{q}\right]_L = q_w\mathbf{I}_{4\times4} + \begin{bmatrix} 0 & -\mathbf{q_v}^T \\ \mathbf{q_v} & \left[\mathbf{q_v}\right]_\times \end{bmatrix}, \quad \left[\mathbf{q}\right]_R = q_w\mathbf{I}_{4\times4} + \begin{bmatrix} 0 & -\mathbf{q_v}^T \\ \mathbf{q_v} & -\left[\mathbf{q_v}\right]_\times \end{bmatrix} \tag{A.5}$$

The relation between quaternions and rotation matrices can be derived from (2.10), (A.4), and (A.5):

$$\begin{bmatrix} 0 \\ \mathbf{v}' \end{bmatrix} = \mathbf{q} \otimes \begin{bmatrix} 0 \\ \mathbf{v} \end{bmatrix} \otimes \mathbf{q}^* = \left[\mathbf{q}^*\right]_R \left[\mathbf{q}\right]_L \begin{bmatrix} 0 \\ \mathbf{v} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{Rv} \end{bmatrix} \tag{A.6}$$

From above the direct conversion from quaternion to rotation matrix can be obtained using (A.5) and results in:

$$\mathbf{R}\{\mathbf{q}\} = (q_w^2 - \mathbf{q}_v^T\mathbf{q})\mathbf{I}_{3\times3} + 2\mathbf{q}_v\mathbf{q}_v^T + 2q_w\left[\mathbf{q}_v\right]_\times \tag{A.7}$$

A.1 Rotation vector to quaternion formula

The $\mathbf{v} = \theta\mathbf{u}$ rotation vector represents rotation around $\mathbf{u}$ axis with θ angle. The operator is denoted by $\mathbf{q}\{\mathbf{v}\}$ throughout this paper and can be written in the form of:

$$\begin{gathered} \mathbf{v} = \theta\mathbf{u} \Rightarrow \mathbf{u} = \frac{\mathbf{v}}{||\mathbf{v}||},\ \theta = ||\mathbf{v}|| \Rightarrow \\ \mathbf{q}\{\mathbf{v}\} = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right)\mathbf{u} \end{bmatrix} = \begin{bmatrix} \cos\left(\frac{||\mathbf{v}||}{2}\right) \\ \sin\left(\frac{||\mathbf{v}||}{2}\right)\frac{\mathbf{v}}{||\mathbf{v}||} \end{bmatrix} \end{gathered} \tag{A.8}$$

A.2 The time derivative of quaternion

When determining the time derivative of a vector, the goal is usually to specify the formula for the derivative in an inertial (fixed) frame with parameters known in a body (not fixed) frame which is only rotating, but not translating in the inertial frame.

Two approaches exist to writing the derivative of a quaternion: one is using an inertial frame-, other is using body frame- known parameters to describe the angular velocity. The angular velocity measurements are typically captured in the body frame, hence it is more practical to choose the second way.

Firstly, I will give the quaternion form of the angular velocity:

$$\begin{aligned}\boldsymbol{\omega}_{\mathcal{L}}(t) &\triangleq \frac{d\boldsymbol{\phi}_{\mathcal{L}}(t)}{dt} = \lim_{\Delta t\to 0}\frac{\Delta\boldsymbol{\phi}_{\mathcal{L}}}{\Delta t} \overset{(A.8)}{=} \lim_{\Delta t\to 0}\frac{\mathbf{q}\{\Delta\boldsymbol{\phi}_{\mathcal{L}}\}}{\Delta t} = \lim_{\Delta t\to 0}\frac{\Delta\mathbf{q}_{\mathcal{L}}}{\Delta t} \\ &= \lim_{\Delta t\to 0}\frac{\begin{bmatrix}\cos(\Delta\theta_{\mathcal{L}}/2)\\ \sin(\Delta\theta_{\mathcal{L}}/2)\mathbf{u}\end{bmatrix}}{\Delta t} \approx \lim_{\Delta t\to 0}\frac{\begin{bmatrix}1\\ (\Delta\theta_{\mathcal{L}}/2)\mathbf{u}\end{bmatrix}}{\Delta t} = \lim_{\Delta t\to 0}\frac{\begin{bmatrix}1\\ \Delta\boldsymbol{\phi}_{\mathcal{L}}/2\end{bmatrix}}{\Delta t}\end{aligned} \tag{A.9}$$

Now, the next step is to write the time derivative of quaternion and use results from the previous equation:

$$\begin{aligned}\frac{d\mathbf{q}(t)}{dt} &= \lim_{\Delta t\to 0}\frac{\mathbf{q}(t+\Delta t)-\mathbf{q}(t)}{\Delta t} = \lim_{\Delta t\to 0}\frac{\mathbf{q}_{\mathcal{L}}\otimes\mathbf{q}-\mathbf{q}}{\Delta t} \\ &= \lim_{\Delta t\to 0}\frac{(\mathbf{q}_{\mathcal{L}}-\mathbf{q}_1)\otimes\mathbf{q}}{\Delta t} = \lim_{\Delta t\to 0}\frac{\begin{bmatrix}0\\ \Delta\boldsymbol{\phi}_{\mathcal{L}}/2\end{bmatrix}\otimes\mathbf{q}}{\Delta t} = \frac{1}{2}\begin{bmatrix}0\\ \boldsymbol{\omega}_{\mathcal{L}}\end{bmatrix}\otimes\mathbf{q},\end{aligned} \tag{A.10}$$

where the distributive property of quaternion product over sum was used, and $\mathbf{q}_1$ denotes the identity quaternion[1]. It is crucial to note that, the $\mathbf{q}(t+\delta t)$ quaternion is utilized by multiplying $\mathbf{q}$ with $\mathbf{q}_{\mathcal{L}}$ from left because it stands for Earth-to-body rotation.

[1] $\mathbf{q}_1 = \begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix}^T$

A.3 Jacobian of quaternion rotation with respect to the vector

To derive the derivative respected to the vector is very easy because the rotation matrix form can be applied:

$$\frac{\partial \mathbf{q} \otimes \mathbf{a} \otimes \mathbf{q}^*}{\partial \mathbf{a}} = \frac{\partial \begin{bmatrix} 0 \\ \mathbf{Ra} \end{bmatrix}}{\partial \mathbf{a}} = \mathbf{R} \tag{A.11}$$

A.4 Jacobian of quaternion rotation with respect to the quaternion

The derivative of the rotation with respect to the quaternion $\mathbf{q}$ is a bit more difficult, but can be derived straightforwardly with the usage of (A.6) and (A.7). Using a lighter notation for the quaternion: $\mathbf{q} = \begin{bmatrix} w \\ \mathbf{v} \end{bmatrix}$, the rotation is:

$$\mathbf{a}' = \mathbf{q} \otimes \mathbf{a} \otimes \mathbf{q}^* = \mathbf{Ra} = w^2\mathbf{a} - \mathbf{v}^T\mathbf{va} + 2\mathbf{vv}^T\mathbf{a} - 2w \left[\mathbf{a}\right]_\times \mathbf{v} \tag{A.12}$$

The Jacobian by the quaternion can be achieved by calculating the derivative of the scalar- and vector- part:

$$\begin{aligned} \frac{\partial \mathbf{a}'}{\partial w} &= 2(w\mathbf{a} + \mathbf{v} \times \mathbf{a}) \\ \frac{\partial \mathbf{a}'}{\partial \mathbf{v}} &= 2(\mathbf{v}^T\mathbf{a}\mathbf{I}_3 + \mathbf{va}^T - \mathbf{av}^T - w \left[\mathbf{a}\right]_\times) \end{aligned} \tag{A.13}$$

Using these results the Jacobian with respect to quaternion is:

$$\frac{\partial(\mathbf{q} \otimes \mathbf{a} \otimes \mathbf{q}^*)}{\partial \mathbf{q}} = 2 \left[w\mathbf{a} + \mathbf{v} \times \mathbf{a} \quad | \quad \mathbf{v}^T\mathbf{a}\mathbf{I}_3 + \mathbf{va}^T - \mathbf{av}^T - w \left[\mathbf{a}\right]_\times \right] \tag{A.14}$$

APPENDIX B

ESKF-RELATED EQUATIONS

All of the following formulas with little difference can be found in [21].

B.1 True rotation matrix

Firstly, I would like to introduce the ODE of rotation matrices. The rotation matrices have orthogonal properties, therefore their inverse is equal to their transpose, which results in the:

$$\mathbf{R}\mathbf{R}^T = \mathbf{I} \tag{B.1}$$

Considering the time derivative of the above yields:

$$\begin{aligned} \frac{d}{dt}(\mathbf{R}\mathbf{R}^T) = \dot{\mathbf{R}}\mathbf{R}^T + \mathbf{R}\dot{\mathbf{R}}^T = 0 \\ \mathbf{R}\dot{\mathbf{R}}^T = -\dot{\mathbf{R}}\mathbf{R}^T \qquad /^T \\ \dot{\mathbf{R}}\mathbf{R}^T = -(\dot{\mathbf{R}}\mathbf{R}^T)^T, \end{aligned} \tag{B.2}$$

which means that, $\dot{\mathbf{R}}\mathbf{R}^T$ is skew-symmetric matrix, therefore there exits an $\boldsymbol{\omega}$ vector which results in:

$$\begin{aligned} \dot{\mathbf{R}}\mathbf{R}^T = \Big[\boldsymbol{\omega}\Big]_\times \qquad / \leftarrow \cdot\mathbf{R} \\ \dot{\mathbf{R}} = \Big[\boldsymbol{\omega}\Big]_\times \mathbf{R} \end{aligned} \tag{B.3}$$

Here, the second row represents the ODE of rotation matrices, which creates a relationship between the derivative of a rotation function, denoted as $r(t)$, and a quantity represented by $\boldsymbol{\omega}$. When considering the scenario around the origin, the certain equation simplifies to $\dot{\mathbf{R}} = \Big[\boldsymbol{\omega}\Big]_\times$. Here, $\boldsymbol{\omega}$ can be interpreted as the vector representing instantaneous angular velocities. This understanding sheds light on

the Lie algebra so(3), which can be seen as the space of derivatives of $r(t)$ at the origin. It also serves as the tangent space to SO(3), the special orthogonal group describing three-dimensional rotations.

If $\boldsymbol{\omega}$ is constant, (B.3) can be time integrated as:

$$\mathbf{R}(t) = e^{\left[\boldsymbol{\omega} t\right]_\times} \mathbf{R}(0) \tag{B.4}$$

The $e^{\left[\boldsymbol{\omega} t\right]_\times}$ expression stands for the rotation matrix related to $\boldsymbol{\omega} t = \mathbf{v}$ rotation vector. This means that the error rotation vector $\delta\boldsymbol{\theta}$ has the connection between the nominal- and the true- rotation matrix:

$$\mathbf{R}_t = \delta\mathbf{R}\mathbf{R} = e^{\left[\delta\boldsymbol{\theta}\right]_\times} \mathbf{R}, \tag{B.5}$$

where the error rotation matrix is described as the exponential of the skew matrix of the error rotation vector, which can be written in Taylor series:

$$e^{\left[\delta\boldsymbol{\theta}\right]_\times} = \sum_{k=0}^{k\to\infty} \frac{1}{k!} \left[\delta\boldsymbol{\theta}\right]_\times^k \tag{B.6}$$

After neglecting the second and higher order members we got the form in (3.4).

B.2 Error-state equations

The error-state can be expressed easily as the composition of the true-state (3.5) and the nominal-state (3.8). As it was detailed in Chapter 3 the error state remains small therefore second-order errors are negligible.

B.2.1 Position error

$$\begin{aligned} \delta\dot{\mathbf{p}}_n &= \dot{\mathbf{p}}_{n,t} - \dot{\mathbf{p}}_n = \mathbf{R}^T \left(\mathbf{I} - \left[\delta\boldsymbol{\theta}\right]_\times\right) (\mathbf{v}_b - \delta\mathbf{v}_b) - \mathbf{R}^T\mathbf{v}_b \\ &= -\mathbf{R}^T \left[\delta\boldsymbol{\theta}\right]_\times \mathbf{v}_b + \mathbf{R}^T\delta\mathbf{v}_b, \end{aligned} \tag{B.7}$$

where nominal state and second-order error were simplified. Using the anti-commutative property of cross-product, it can be written in the form of:

$$\delta\dot{\mathbf{p}}_n = \mathbf{R}^T\delta\mathbf{v}_b + \mathbf{R}^T\left[\mathbf{v}_b\right]_\times \delta\boldsymbol{\theta} \tag{B.8}$$

B.2.2 Angular error

Calculating the angular error equation requires more complicated steps because it desires to use the derivative rule of the quaternion product:

$$(\delta\mathbf{q}\dot{\otimes}\mathbf{q}) = \dot{\delta\mathbf{q}}\otimes\mathbf{q} + \delta\mathbf{q}\otimes\dot{\mathbf{q}} \tag{B.9a}$$

$$= \delta\dot{\mathbf{q}}\otimes\mathbf{q} + \delta\mathbf{q}\otimes\frac{1}{2}\begin{bmatrix}0\\ \boldsymbol{\omega}\end{bmatrix}\otimes\mathbf{q} \tag{B.9b}$$

In (B.9b) the definition of quaternion derivative was used for local angular velocities and earth-to-body rotation. This equation is equal to the dynamics of the true quaternion state, which is defined in (3.5b), therefore simplifying with the terminal $\mathbf{q}$ and isolating $\delta\mathbf{q}$ yields:

$$\begin{aligned} 2\dot{\delta\mathbf{q}} &= \begin{bmatrix}0\\ \boldsymbol{\omega}_t\end{bmatrix}\otimes\delta\mathbf{q} - \delta\mathbf{q}\otimes\begin{bmatrix}0\\ \boldsymbol{\omega}\end{bmatrix} \\ &= \left(\left[\mathbf{q}\right]_L\left\{\begin{bmatrix}0\\ \boldsymbol{\omega}_t\end{bmatrix}\right\} - \left[\mathbf{q}\right]_R\left\{\begin{bmatrix}0\\ \boldsymbol{\omega}\end{bmatrix}\right\}\right)\delta\mathbf{q} \end{aligned} \tag{B.10}$$

Converting $\delta\mathbf{q}$ to $\delta\boldsymbol{\theta}$ and performing the matrix substraction results in:

$$\begin{aligned} \begin{bmatrix}0\\ \dot{\delta\boldsymbol{\Theta}}\end{bmatrix} &= \begin{bmatrix}0 & -(\boldsymbol{\omega}_t-\boldsymbol{\omega})^T\\ \boldsymbol{\omega}_t-\boldsymbol{\omega} & \left[\boldsymbol{\omega}_t+\boldsymbol{\omega}\right]_\times\end{bmatrix}\begin{bmatrix}1\\ \frac{\delta\boldsymbol{\Theta}}{2}\end{bmatrix} + O(||\delta\boldsymbol{\Theta}||^2) \\ &= \begin{bmatrix}0 & -\delta\boldsymbol{\omega}^T\\ \delta\boldsymbol{\omega} & \left[2\boldsymbol{\omega}+\delta\boldsymbol{\omega}\right]_\times\end{bmatrix}\begin{bmatrix}1\\ \frac{\delta\boldsymbol{\Theta}}{2}\end{bmatrix} + O(||\delta\boldsymbol{\Theta}||^2) \end{aligned} \tag{B.11}$$

From the above arises a scalar- and a vector- equality. The scalar part is formed by second-order infinitesimals, which is not very useful, therefore only the vector part should be expressed. After neglecting second-order terms and substituting parameters from Table 3.1 into the nominal- and error- parameters yields:

$$\dot{\delta\boldsymbol{\Theta}} = \left[\boldsymbol{\omega}_m - \boldsymbol{\beta}_\omega\right]_\times \delta\boldsymbol{\Theta} - \delta\boldsymbol{\beta}_\omega - \boldsymbol{\eta}_\omega \tag{B.12}$$

B.2.3 Velocity error

The velocity error is derived very similarly to the position error:

$$\begin{aligned}\dot{\delta\mathbf{v}}_b = \dot{\mathbf{v}}_{b,t} - \dot{\mathbf{v}}_b = \mathbf{a} + \delta\mathbf{a} + \left(\mathbf{I} + \left[\delta\boldsymbol{\Theta}\right]_\times\right)\mathbf{R}\mathbf{g} - \left[\boldsymbol{\omega} + \delta\boldsymbol{\omega}\right]_\times (\mathbf{v}_b + \delta\mathbf{v}_b) \\ - (\mathbf{a} + \mathbf{R}\mathbf{g} - \left[\boldsymbol{\omega}\right]_\times \mathbf{v}_b)\end{aligned} \tag{B.13}$$

Simplifying with nominal state and neglecting second-order terms gives the following equation:

$$\dot{\delta\mathbf{v}}_b = \delta\mathbf{a} + \left[\delta\boldsymbol{\Theta}\right]_\times \mathbf{R}\mathbf{g} + \left[\delta\boldsymbol{\omega}\right]_\times \mathbf{v}_b - \left[\boldsymbol{\omega}\right]_\times \delta\mathbf{v}_b \tag{B.14}$$

To achieve the final form of the equation, parameters inserted from Table 3.1 into the nominal- and error-state, which results in:

$$\begin{aligned}\dot{\delta\mathbf{v}}_b = -\left[\mathbf{R}\mathbf{g}\right]_\times \delta\boldsymbol{\Theta} - \left[\boldsymbol{\omega}_m - \boldsymbol{\beta}_\omega\right]_\times \delta\mathbf{v}_b - \delta\boldsymbol{\beta}_a - \left[\mathbf{v}_b\right]_\times \delta\boldsymbol{\beta}_\omega \\ - \boldsymbol{\eta}_a - \left[\mathbf{v}_b\right]_\times \boldsymbol{\eta}_\omega\end{aligned} \tag{B.15}$$

B.3 Jacobian of true quaternion with respect to the error rotation vector

The relationship between the true quaternion and the error rotation vector can be determined using the chain rule, which can be formulated as follows:

$$\begin{aligned}\frac{\partial(\delta\mathbf{q}\otimes\mathbf{q})}{\partial\delta\boldsymbol{\theta}} &= \frac{\partial(\delta\mathbf{q}\otimes\mathbf{q})}{\partial\delta\mathbf{q}}\frac{\partial\delta\mathbf{q}}{\partial\delta\boldsymbol{\theta}} \\ &= \frac{\partial\left(\left[\mathbf{q}\right]_R\delta\mathbf{q}\right)}{\partial\delta\mathbf{q}}\frac{\partial\begin{bmatrix}1\\ \frac{1}{2}\delta\boldsymbol{\theta}\end{bmatrix}}{\partial\delta\boldsymbol{\theta}} = \frac{1}{2}\left[\mathbf{q}\right]_R\begin{bmatrix}\mathbf{0}^T\\ \mathbf{I}_3\end{bmatrix} \\ &= \frac{1}{2}\begin{bmatrix}-q_x & -q_y & -q_z\\ q_w & q_z & -q_y\\ -q_z & q_w & q_x\\ q_y & -q_x & q_w\end{bmatrix}\end{aligned} \tag{B.16}$$

B.4 Jacobian of reset function (g) with respect to the error rotation vector

The goal is to determine the derivative of (3.31b) by the error rotation vector $\delta\boldsymbol{\theta}$. Firstly, I aim to formalize that equation with rotation vector parameters, neglecting the constant terms:

$$\begin{aligned}\mathbf{q}\{\delta\boldsymbol{\theta}\}\otimes\mathbf{q}\{-\hat{\delta\boldsymbol{\theta}}\} &= \left[\mathbf{q}\right]_R\left\{\begin{bmatrix}1\\ -\frac{1}{2}\hat{\delta\boldsymbol{\theta}}\end{bmatrix}\right\}\begin{bmatrix}1\\ \frac{1}{2}\delta\boldsymbol{\theta}\end{bmatrix} + \mathcal{O}(||\delta\boldsymbol{\theta}||^2) \\ &= \begin{bmatrix}1 & \frac{1}{2}\hat{\delta\boldsymbol{\theta}}^T\\ \frac{1}{2}\hat{\delta\boldsymbol{\theta}} & \mathbf{I}+\left[\frac{1}{2}\hat{\delta\boldsymbol{\theta}}\right]_\times\end{bmatrix}\begin{bmatrix}1\\ \frac{1}{2}\delta\boldsymbol{\theta}\end{bmatrix} + \mathcal{O}(||\delta\boldsymbol{\theta}||^2) \\ &= \begin{bmatrix}1+\frac{1}{4}\hat{\delta\boldsymbol{\theta}}^T\delta\boldsymbol{\theta}\\ \frac{1}{2}\hat{\delta\boldsymbol{\theta}}+\frac{1}{2}\left(\mathbf{I}+\left[\frac{1}{2}\hat{\delta\boldsymbol{\theta}}\right]_\times\right)\delta\boldsymbol{\theta}\end{bmatrix} + \mathcal{O}(||\delta\boldsymbol{\theta}||^2)\end{aligned} \tag{B.17}$$

This is equal to the error quaternion after the reset. Denotating the error rotation vector with $\delta\boldsymbol{\theta}^{+}$ the following equation can be written:

$$\begin{bmatrix} 1 \\ \frac{1}{2}\delta\boldsymbol{\theta}^{+} \end{bmatrix} = \begin{bmatrix} 1 + \frac{1}{4}\hat{\delta\boldsymbol{\theta}}^{T}\delta\boldsymbol{\theta} \\ \frac{1}{2}\hat{\delta\boldsymbol{\theta}} + \frac{1}{2}\left(\mathbf{I} + \left[\frac{1}{2}\hat{\delta\boldsymbol{\theta}}\right]_{\times}\right)\delta\boldsymbol{\theta} \end{bmatrix} + \mathcal{O}(||\delta\boldsymbol{\theta}||^{2}) \tag{B.18}$$

The above equation splits into a scalar- and a vector- part, where the scalar part is formed by second-order infinitesimals, therefore it is not very useful, additionally the constant member in the original (3.31b) equation is responsible for exactly this neglect. Only considering the vector part, the Jacobian:

$$\frac{\partial g(\delta\boldsymbol{\theta})}{\partial\delta\boldsymbol{\theta}} = \frac{\partial\delta\boldsymbol{\theta}^{+}}{\partial\delta\boldsymbol{\theta}} = \frac{\partial\left(\hat{\delta\boldsymbol{\theta}} + \left(\mathbf{I} + \left[\frac{1}{2}\hat{\delta\boldsymbol{\theta}}\right]_{\times}\right)\delta\boldsymbol{\theta}\right)}{\partial\delta\boldsymbol{\theta}} = \mathbf{I} + \left[\frac{1}{2}\hat{\delta\boldsymbol{\theta}}\right]_{\times} \tag{B.19}$$

ABBREVIATIONS

BME	Budapesti Műszaki Egyetem
EKF	Extended Kalman Filter
ESKF	Error-State Kalman Filter
GPS	Global Positioning System
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
KF	Kalman Filter
NED	North-East-Down
RMSE	Root Mean Square Error
SCL	System Control Laboratory
SZTAKI	Számítástechnikai és Automatizálási Kutatóintézet
UAS	Unmanned Aerial System
UAV	Unmanned Aerial Vehicle
3D	Three-dimensional

List of Figures

1.1 Block diagram of relative navigation ([7]: page 2, Figure 2) 3

1.2 Realistic drawing of Sindy ([15]: title page) 4

2.1 Applied coordinate systems 6

2.2 NED coordinate system [16] 7

2.3 Frames to define Euler angles ([10]: pages 13-15, Figures 2.4-2.7) . . 8

2.4 Camera and image system ([17]: page 7) 10

4.1 Illustration of intersection (top) and resection (bottom) forms of the triangulation problem ([23]: page 3) 31

6.1 Camera image example 47

6.2 Straight and level flight path 48

6.3 Straight and level - position 48

6.4 Straight and level - orientation 48

6.5 Straight and level - velocity 48

6.6 Straight and level - acceleration bias 49

6.7 Straight and level - angular velocity bias 49

List of Tables

3.1 ESKF states and inputs 18

6.1 Noise summary 45

6.2 Camera intrinsic parameters 45

BIBLIOGRAPHY

[1] David Erdos, Abraham Erdos, and Steve E. Watkins. An experimental uav system for search and rescue challenge. *IEEE Aerospace and Electronic Systems Magazine*, 28(5):32–37, 2013.

[2] Friedrich Fraundorfer, Lionel Heng, Dominik Honegger, Gim Hee Lee, Lorenz Meier, Petri Tanskanen, and Marc Pollefeys. Vision-based autonomous mapping and exploration using a quadrotor mav. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4557–4564, 2012.

[3] Abdulla Al-Kaff, Francisco Miguel Moreno, Luis Javier San José, Fernando García, David Martín, Arturo de la Escalera, Alberto Nieva, and José Luis Meana Garcéa. Vbii-uav: Vision-based infrastructure inspection-uav. In Álvaro Rocha, Ana Maria Correia, Hojjat Adeli, Luís Paulo Reis, and Sandra Costanzo, editors, *Recent Advances in Information Systems and Technologies*, pages 221–231, Cham, 2017. Springer International Publishing.

[4] V. Tirronen H. Salo and F. Neri. Evolutionary regression machines for precision agriculture. In *Applications of Evolutionary Computation*, pages 356–365, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[5] Stefan Leutenegger and Roland Y. Siegwart. A low-cost and fail-safe inertial navigation system for airplanes. In *2012 IEEE International Conference on Robotics and Automation*, pages 612–618, 2012.

[6] Cesario Vincenzo Angelino, Vincenzo Rosario Baraniello, and Luca Cicala. High altitude uav navigation using imu, gps and camera. In *Proceedings of the 16th International Conference on Information Fusion*, pages 647–654, 2013.

[7] Gary Ellingson, Kevin Brink, and Tim McLain. Relative navigation of fixed-wing aircraft in gps-denied environments. *NAVIGATION*, 67(2):255–273, 2020.

[8] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, 2010.

[9] David O. Wheeler, Daniel P. Koch, James S. Jackson, Timothy W. McLain, and Randal W. Beard. Relative navigation: A keyframe-based approach for observable gps-degraded navigation. *IEEE Control Systems Magazine*, 38(4):30–48, 2018.

[10] RANDAL W. BEARD and TIMOTHY W. McLAIN. *Small Unmanned Aircraft: Theory and Practice*. Princeton University Press, 2012.

[11] Stephan Weiss, Markus W. Achtelik, Simon Lynen, Margarita Chli, and Roland Siegwart. Real-time onboard visual-inertial state estimation and self-calibration of mavs in unknown environments. In *2012 IEEE International Conference on Robotics and Automation*, pages 957–964, 2012.

[12] Timothy D. Barfoot and Paul T. Furgale. Associating uncertainty with three-dimensional poses for use in estimation problems. *IEEE Transactions on Robotics*, 30(3):679–693, 2014.

[13] Robert C. Leishman, Timothy W. McLain, and Randal W. Beard. Relative navigation approach for vision-based aerial gps-denied navigation. *Journal of Intelligent & Robotic Systems*, 74(1–2):97–111, 2013.

[14] David O. Wheeler, Paul W. Nyholm, Daniel P. Koch, Gary J. Ellingson, Timothy W. McLain, and Randal W. Beard. Relative navigation in gps-degraded environments. *Encyclopedia of Aerospace Engineering*, page 1–10, 2016.

[15] István Gőzse, Máté Kisantal, Péter Onódi, and Pierre Arnaud. *Sindy UAV test platform assembly manual*. Institute for Computer Sciences and Control, 2016.

[16] Gincsainé Szádeczky-Kardoss Emese and Kis László. Navigáció és pályatervezés: A gps alapjai.

[17] Kris Kitani. 8.2 camera matrix. Presentation slide, March 2017.

[18] Wikipedia contributors. Pinhole camera - wikipedia, 2023. Accessed: December 4, 2023.

[19] Kenji Hata and Silvio Savarese. *CS231A Course Notes 1: Camera Models*. Stanford University, 2021. Accessed: December 4, 2023.

[20] W. R. Hamilton. On a new species of imaginary quantities, connected with the theory of quaternions. *Proceedings of the Royal Irish Academy (1836-1869)*, 2:424–434, 1840.

[21] Joan Solà. Quaternion kinematics for the error-state kalman filter. *CoRR*, abs/1711.02508, 2017.

[22] Gabriel A. Terejanu. Discrete kalman filter tutorial.

[23] Sébastien Henry and John A. Christian. Absolute triangulation algorithms for space exploration. *Journal of Guidance, Control, and Dynamics*, 46(1), 2022.

[24] G. T. McCaw. Resection in survey. *The Geographical Journal*, 52(2):105–123, 1918.

[25] Arun Bernard, Akhter Mahmud Nafi, and David Geller. Using triangulation in optical orbit determination. In *2018 AAS/AIAA Astrodynamics Specialist Conference*, 09 2018.

[26] Long Chen, Chengzhi Liu, Zhenwei Li, and Zhe Kang. A new triangulation algorithm for positioning space debris. *Remote Sensing*, 13(23), 2021.

[27] Richard I. Hartley and Peter Sturm. Triangulation. *Comput. Vis. Image Underst.*, 68(2):146–157, nov 1997.

[28] M Pollefeys, R Koch, M Vergauwen, and L Van Gool. Automated reconstruction of 3d scenes from sequences of images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 55(4):251–267, 2000.

[29] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Modeling the world from internet photo collections. *Int. J. Comput. Vision*, 80:189–210, nov 2008.

[30] James L. Poirot and Gerald V. McWilliams. Navigation by back triangulation. *IEEE Transactions on Aerospace and Electronic Systems*, AES-12(2):270–274, 1976.

[31] Wikipedia contributors. Moore–penrose inverse, 2023. Accessed: December 7, 2023.

[32] Gene H. Golub and Charles F. Van Loan. *Matrix Computations - 4th Edition*. Johns Hopkins University Press, Philadelphia, PA, 2013.

[33] Joel N. Franklin. *Matrix Theory*. Dover Publications, 1968.

[34] D. S. Oliver. Gaussian cosimulation: Modelling of the cross-covariance. *Mathematical Geology*, 2003.

[35] C. Radhakrishna Rao. *Linear Statistical Inference and its Applications*, page 521. John Wiley & Sons, Ltd, 1973.

[36] Francois Alabert. The practice of fast conditional simulations through the lu decomposition of the covariance matrix. *Mathematical Geology*, 19(5):369–386, July 1 1987.

[37] Michael W. Davis. Production of conditional simulations via the lu triangular decomposition of the covariance matrix. *Mathematical Geology*, 19(2):91–98, February 1 1987.

[38] Szabolcs Kun. Image-based inertial navigation. Master's thesis, Budapest University of Technology and Economics, Budapest, Hungary, 5 2021.

[39] Peter Bauer and Szabolcs Kun. Optical flow-based angular rate sensor fault detection on uavs*. *IFAC-PapersOnLine*, 55(14):46–51, 2022. 11th IFAC Symposium on Intelligent Autonomous Vehicles IAV 2022.