Budapest University of Technology and Economics

Faculty of Electrical Engineering and Informatics

Department of Control Engineering and Information Technology

HUN-REN 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 17, 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

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 17.

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. Linear Optimal Sine Triangulation (LOST) 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. Fő célom az algoritmus tesztelése egy olyan szimulációs környezetben, amelyben a kamera felvételek Unreal Car Learning to Act (CARLA) szimulációs környezet segítségével készültek.

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. My primary objective is to test the developed navigation system in a simulation environment where the camera images are generated with the help of the Unreal Car Learning to Act (CARLA) simulation environment.

Chapter 1

Introduction

Until the early 2000s Unmanned Aerial Vehicles (UAVs) were 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 become 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 also advancement in traditional navigation methods, where the Global Positioning System (GPS) is combined with the Inertial Navigation System (INS), creating a 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. The 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 rela-

tively 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.

1.1 Possible sources of global information

When an algorithm fails to close the navigation loop, particularly in the absence of external absolute positioning references, 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.

Of course, the most straightforward solution to this problem is to use the signals of Global Navigation Satellite Systems (GNSS) such as the US GPS, the Russian GLONASS (Global Navigation Satellite System), the European Galileo [8], or the Chinese BDS (BeiDou Navigation Satellite System) [9]. The problem with these systems they are not always available for example, in indoor environments, the GPS signal can be shadowed, and in outdoor environments, it can be jammed or spoofed.

Another solution is to use a map of the environment, which is usually called Simultaneous Localization and Mapping (SLAM) [10]. The main idea behind SLAM is to estimate the position of the vehicle and the map of the environment simultaneously. The map can be used as a reference to correct the drift errors. The map can be built up by using the measurements of the exteroceptive sensors, such as cameras, laser scanners, etc. The problem with this approach is that the map can be built up only in the explored area, therefore it is not a solution for the first flight of the aircraft.

The last remedy can involve leveraging Signals of Opportunity (SOPs), which are signals not originally intended for navigation but can be used for this purpose. In [11], they propose SOPs can be terrestrial (e.g., FM radio, cellular, digital television) or space-based (e.g., Low Earth Orbit (LEO) satellites). In naviga-

tion, SOPs leverage known synchronization sequences or beacons from the used sources. These signals enable cognitive navigation by providing measurements like Time-of-Arrival (TOA), indicating signal travel time; Direction-of-Arrival (DOA), specifying signal direction; and Frequency-of-Arrival (FOA), denoting signal frequency [12]. In the case of LEO satellites, their broadband communication signals can be used for navigation [13].

1.2 Relative navigation

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 [14], Unmanned Aerial Systems (UAS) [15, 16], 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 are unobservable, which eventually leads to the divergence of estimates [17, 18]. 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 [19, 20], which can handle these observability and consistency problems. With exteroceptive sensors, the 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.

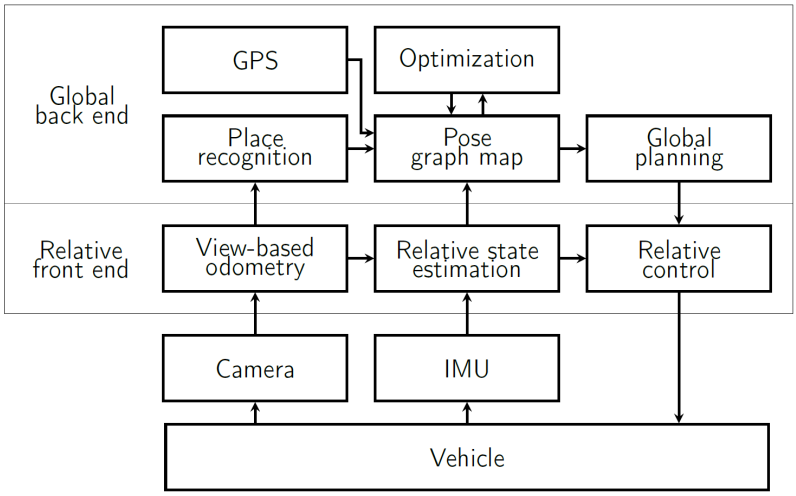


Figure 1.1: Block diagram of relative navigation ([7]: page 2, Figure 2)

The front end provides an estimation of the state relative to the local environment, while the back end is a mapping algorithm, which uses a pose graph based on the results of the front end to produce global estimates. Several important observability and computational benefits are 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 (ideal) state according to the original non-linear dynamics and the linearization is performed on the so-called error state which is defined around the nominal state.

- On the other hand, the back end uses a pose 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. The target UAV is a fixed-wing aircraft called Sindy, shown in Figure 1.2.

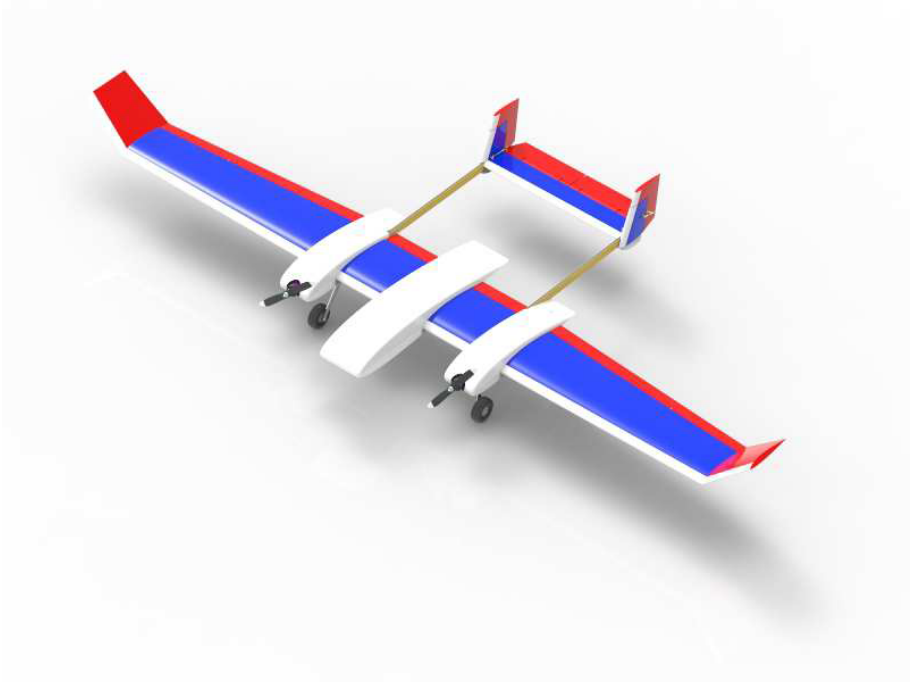


Figure 1.2: Realistic drawing of Sindy ([21]: title page)

1.3 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 me. The simulation contained 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, pixelization, and later the usage of real image processing algorithms. Firstly, I implemented the 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 integration of the two methods.

The paper is structured as follows: Chapter 2 provides an introduction to several fundamental mathematical concepts that are crucial for comprehending this 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 introduc-

ing the integration issues of ESKF and LOST. Chapter 6 presents the results of the filter in the Matlab/Simulink simulation 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, it is not applied 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.

In the following, I employ notations n, b, and c as a subscript for vectors to denote their representation in the inertial, body, and camera frames respectively, while the superscript indicates the origin of the respective coordinate system.

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:

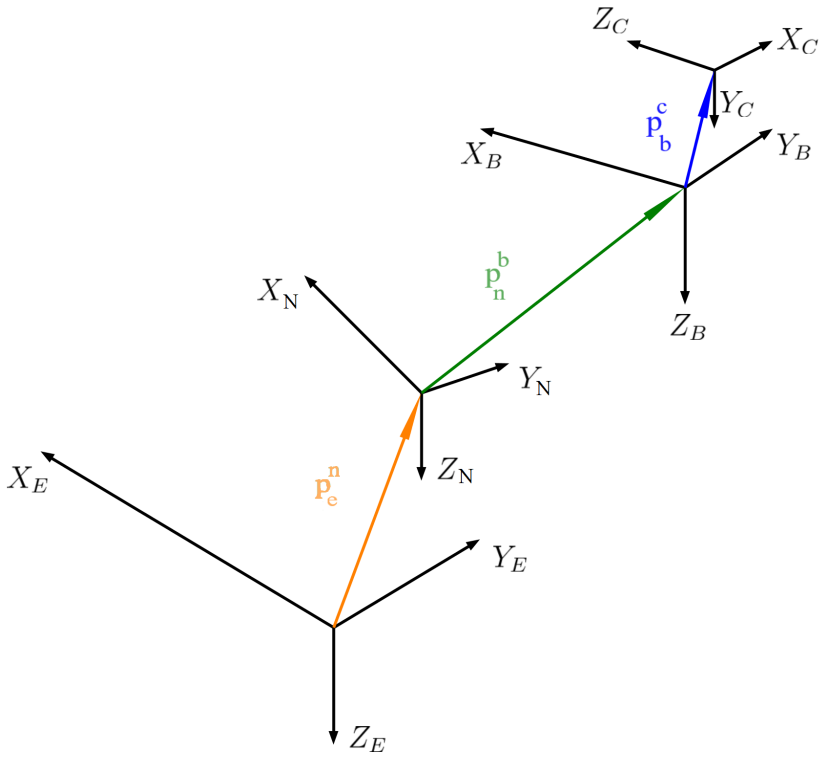


Figure 2.1: Applied coordinate systems

- 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 (NED) 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[1]. The NED frame can be

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

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 pointing inwards to the Earth. Its X-Y plane is the local tangent plane of the Earth's ellipsoid.

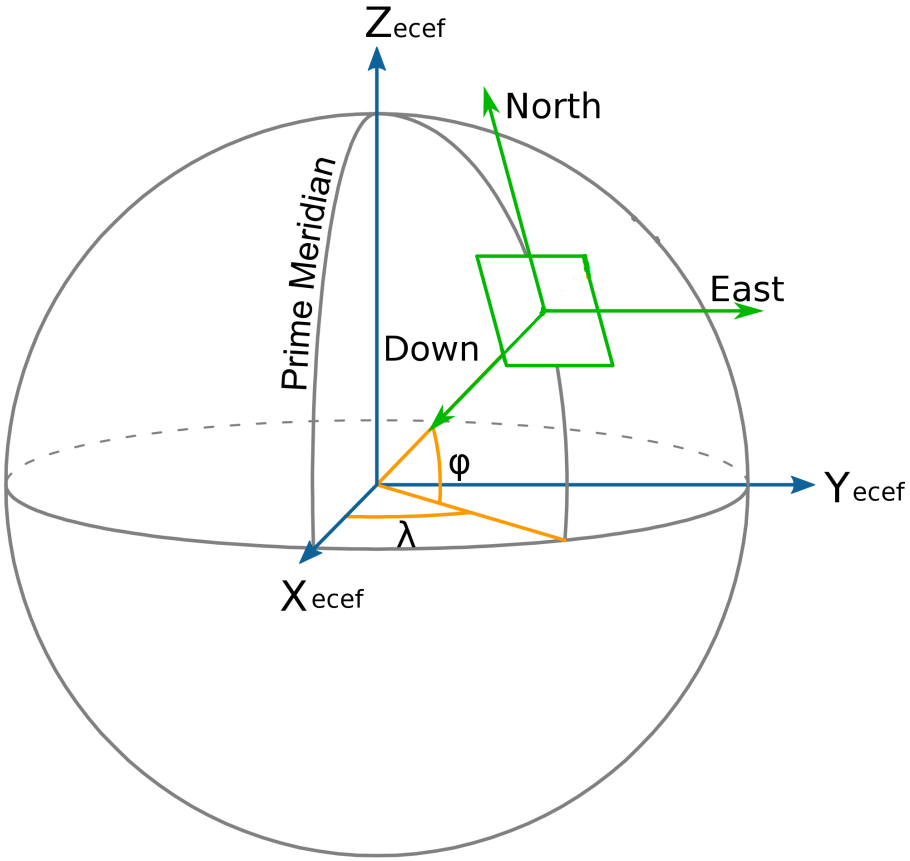


Figure 2.2: NED coordinate system [22]

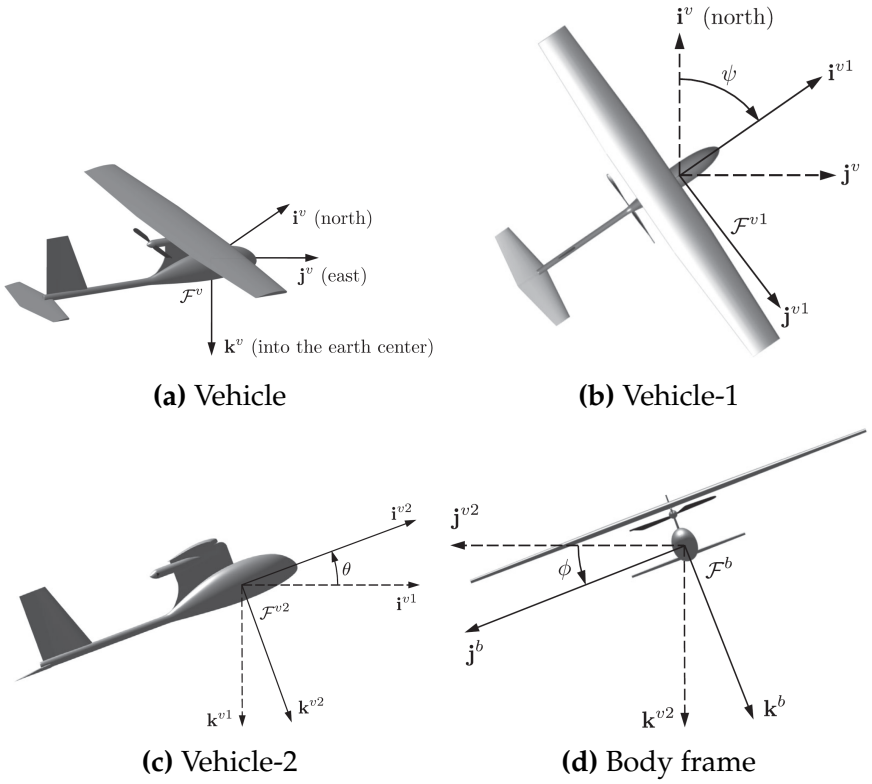
In relative navigation methods, the front end adopts the node system as an inertial frame because from time to time the uncertainty of the position gets reset. For example, in [7] the node frame remains locally fixed and undergoes redeclaration whenever a measurement update is performed. The back end is responsible for producing global estimates, such as NED coordinates, utilizing the outcomes yielded by the filter.

In this paper, since there is no node frame I am denoting the applied inertial frame (which is currently the NED system) with n.

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 the relation to the inertial frame can be accounted for. The origin is aligned with the center of the aircraft's mass, and it is worth mentioning the fact in realizations the IMU is placed nearby. 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 half 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 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 [16], 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 ([16]: pages 13–15, Figures 2.4–2.7)

The vehicle frame is the NED coordinate system with shifted origin into the center of mass, the vehicle-1 frame is rotated with the yaw angle (ψ) around the Z-axis of

the vehicle frame, and the 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 to a general 3-D vector $\mathbf{r}_b$ as:

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

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 in-

terpreted 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 utilized. For the pinhole model, the intrinsic parameters of the camera are the focal length (f), the principal point (p_x, p_y), and a constraint for the visibility that is described with the field of view (FOV) or image size. Extrinsic parameters represent the position and orientation of the camera in the inertial frame.

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 denoted as $\mathbf{p}_b^c$. Regarding the orientation, an at least partially downward-facing camera is mandatory, whereas the navigation happens concerning the surroundings 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.

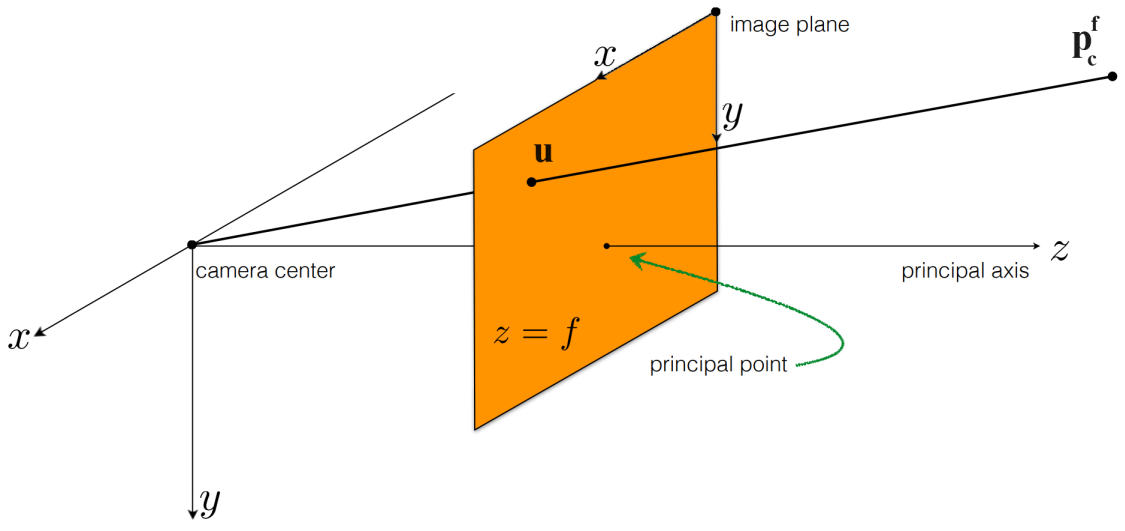


Figure 2.4: Camera and image system ([23]: page 7)

The axes swap operation is readily apparent considering how the rotated body frame and camera frame axes are defined. The transformation is outlined as $\mathbf{i}_b' = \mathbf{k}_c$, $\mathbf{j}_b' = \mathbf{i}_c$, and $\mathbf{k}_b' = \mathbf{j}_c$, where $\mathbf{i}$, $\mathbf{j}$, and $\mathbf{k}$ are unit vectors along the 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 axis swapping transformation, and $\mathbf{R}_{CB}$ represents the camera rotation with a β angle around the Y-axis (for a downward-facing camera it should be negative). The resulting matrix is still unitary, therefore $\mathbf{T}_{BC} = \mathbf{T}_{CB}^T$.

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

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

Finally, I mention a few words about the image coordinate system. As can be seen in Figure 2.4, 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 the center of the image and shifting with it results in a top left corner origin of the pixel coordinate system. This is because the pinhole projection produces an inverted image, but it's detailed in the next section.

2.3 Pinhole camera projection

The pinhole camera model serves as a mathematical representation of 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 position $\mathbf{p}_c^f$ in the camera frame.

The pinhole model framework describes the aperture of the camera as a point, therefore the lens distortion errors are neglected [24, 25]. 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 $\overline{\mathbf{u}}$ is the resulting measurement which pixel coordinates are u and v and its z coordinate equals the focal length since the vector points onto the image. The interpretation domain of the pinhole model contains all $\mathbf{p}_c^f$ for which $p_{c,z}^f > 0$, but as a final point, I want to emphasize that the finite image size (W, H) or the FOV angles constraint has 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 [26] 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 [27]. 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 rotations, described by $\mathbf{q}_{AB}$ and $\mathbf{q}_{BC}$ quaternions can also be evaluated using the 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 [27], 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, $\mathbf{R}\{\mathbf{q}\}$ denotes 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. This optimal estimate is the best linear unbiased estimate of the state, making the Kalman filter an unbiased minimum variance filter. It achieves 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 [28].

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 the 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 the ESKF has several advantages:

- Since the error state remains small the linear Kalman filter approach is applicable 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 Error-state kinematics in continuous time

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 available as IMU measurements, the position ($\mathbf{p}_n^b$), velocity ($\mathbf{v}_b$), and orientation ($\mathbf{q}_{BN}$) are included in the state. Moreover, the state is supplemented by estimating the biases of the sensors ($\boldsymbol{\beta}_a$, $\boldsymbol{\beta}_\omega$) for acceleration and angular rate sensors respectively.

The sensor-related quantities and the velocity are expressed in the body frame, but the position and the orientation are expressed in the inertial frame. However, I use $\mathbf{p}$ and $\mathbf{q}$ lighter notations for accounting for a relation between the body and

the inertial frame in the context of ESKF. This results in the state vector:

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

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 the biases, while the small and slowly varying effects are assigned to the error state. A few comments on the table:

- All quantity yields a simple sum operation on the nominal and the error state to get the true state except the orientation.
- In the nominal state, orientation is described with quaternion ($\mathbf{q}$). However, the error in the rotation is described with a rotation vector ($\delta\boldsymbol{\theta}$) that represents a small deviation from a reference rotation. The relation was presented between the quaternion and rotation vector in Appendix A.2. 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 into (3.2), the true value of the acceleration and angular rate can be expressed:

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

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

- In the context of 3-D rotations, the unit quaternions and rotation matrices belong to the $\mathrm{SO}(3)$ group, where $\mathrm{SO}(3)$ represents Special Orthogonal group in 3-D. The skew-symmetric matrix of rotation vector $\left[\delta\boldsymbol{\theta}\right]_\times$ belongs to the $\mathfrak{so}(3)$ Lie Algebra and in [27] 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 the Taylor series, and neglecting the second- and higher-order terms:

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

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

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 detailed in Appendix A.3. (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:

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

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

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

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

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

The nominal biases are modeled as constants.

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:

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

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

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

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

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

The calculations are detailed in Appendix B.2.

3.2.2 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 [27]: 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:

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

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

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

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

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

$\boldsymbol{\theta}_i$, $\mathbf{v}_{b,i}$, $\mathbf{a}_i$ and $\boldsymbol{\omega}_i$ are the random impulses that form the stochastic part of the equation. Those noises that appear only negatively in the equations can be changed to positive freely as the probability of negative or positive white noise components is the same. Their covariance matrices are integrated as (see [27] Appendix E for details):

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

where σ_{η_a}, σ_{η_ω}, σ_{β_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 the 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 a priori unless a visual map of the surroundings is available onboard.

3.4 ESKF framework

The ESKF directly estimates the nominal ($\mathbf{x}$) and the error state ($\delta\mathbf{x}$), and the system is governed by the IMU measurements $\mathbf{m}$ and the noise perturbations $\mathbf{i}$:

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

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), considering $\delta\mathbf{x}_{k+1} = f(\delta\mathbf{x}_k)$:

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

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

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

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

$$\delta \hat{\mathbf{x}}_{k+1} = \mathbf{F}_x \delta \hat{\mathbf{x}}_k = \mathbf{0}, \quad \mathbf{0} \in \mathbb{R}^{15\times 1} \tag{3.19b}$$

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

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 is performed and the error is injected into the nominal state it gets reset, 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 the feature point

The Jacobian of the projection is calculated by linearizing (2.7). The transformation results in a 3-D vector, but the 3^{rd} coordinate is always equal to the focal length, and its derivative leads to zeros, which contain no information, hence 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:

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

From the elements of the state are only the position and orientation required to calculate the feature position, 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 (3.26), then the derivative should be decomposed into separate components using the chain rule: one with respect to the vector and the other with respect to the quaternion:

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

Using partial results for the two Jacobian above from Appendices A.4 and A.5, the outcome is:

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

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

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

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. The Jacobian of the quaternion with respect to the rotation vector results

in:

$$\frac{\partial(\delta\mathbf{q}\otimes\mathbf{q})}{\partial\delta\boldsymbol{\theta}} = \mathbf{Q}_{\delta\theta} = \frac{1}{2}\begin{bmatrix}\mathbf{q}\end{bmatrix}_R \begin{bmatrix}\mathbf{0}^T \\ \mathbf{I}_3\end{bmatrix}, \quad \mathbf{Q}_{\delta\theta}\in\mathbb{R}^{4\times3} \tag{3.27}$$

Detailed calculations can be found in Appendix B.3. Using the formula above leads to the Jacobian:

$$\frac{\partial\mathbf{x}_t}{\partial\delta\mathbf{x}} = \mathbf{X}_{\delta x} = \begin{bmatrix}\frac{\partial\mathbf{p}_{n,t}}{\partial\delta\mathbf{p}_n} & \cdots & \mathbf{0}_{3\times3} \\ \vdots & \ddots & \vdots \\ \mathbf{0}_{3\times3} & \cdots & \frac{\partial\boldsymbol{\beta}_{\omega,t}}{\partial\delta\boldsymbol{\beta}_\omega}\end{bmatrix} = \begin{bmatrix}\mathbf{I}_3 & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{4\times3} & \mathbf{Q}_{\delta\theta} & \mathbf{0}_{4\times9} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_9\end{bmatrix} \tag{3.28}$$

3.4.3 Error injection into the nominal-state

After calculating the measurement matrix, the Kalman gain ($\mathbf{K}_g$) 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 a priori, therefore it's a complex step and I will detail its calculation later. The Kalman gain and the residual ($\mathbf{r}$) can be utilized to calculate the state update:

$$\delta\hat{\mathbf{x}} = \mathbf{K}_g\mathbf{r} \quad \mathbf{K}_g\in\mathbb{R}^{15\times2k}, \quad \mathbf{r}\in\mathbb{R}^{2k\times1}, \tag{3.29}$$

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:

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

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

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

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

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

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:

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

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 estimating the state using camera measurements, assuming that the positions of the visible feature points are known. However, in practical applications, assuming prior knowledge of the feature positions is often untenable. Consequently, there arises the need to estimate these positions, 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 [29], called Linear Optimal Sine Triangulation (LOST). The core equations are utilized as described therein, although 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 the position and the orientation of aircraft are involved in the triangulation equations, and how the recursive version of LOST can be formalized.

4.1 Problem statement

The modern triangulation problem takes on one of two forms: intersection or resection [30], as shown in Figure 4.1.

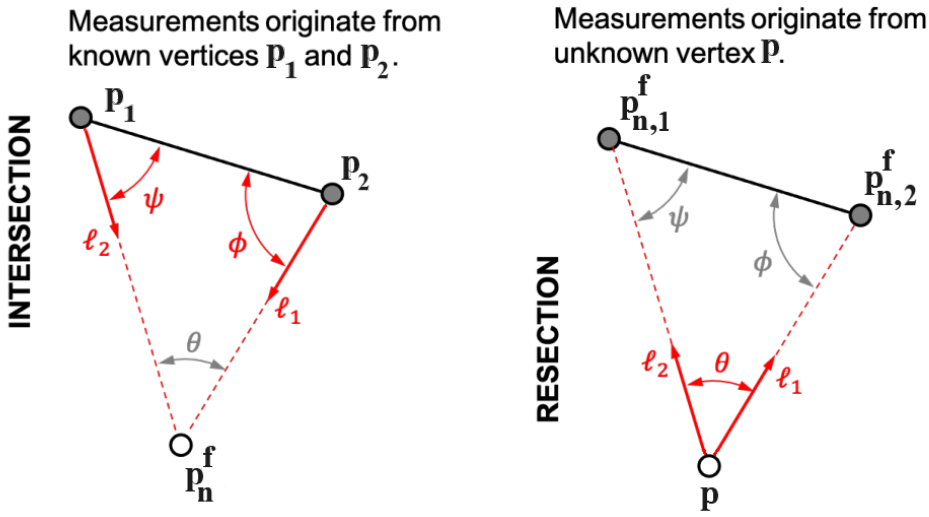


Figure 4.1: Illustration of intersection (top) and resection (bottom) forms of the triangulation problem ([29]: page 3)

It is crucial to highlight that in Figure 4.1, $\mathbf{p}_i$ vertices are the camera positions at each measurement expressed in the NED frame ($\mathbf{p}_n^c$). The main difference between the two formalizations is that the intersection assumes measurements taken from known vertices ($\mathbf{p}_1$, $\mathbf{p}_2$), 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 [31, 32], and 3-D scene reconstruction usually called Structure from Motion (SfM) [33, 34, 35]. The resection problem describes the vehicle localization problem that is generally included in navigation applications [19, 20, 36]. 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.

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 in the camera system ($\mathbf{p}_c^f$) 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 [29], 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 Transformation (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 camera measurements and aircraft 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. Using the error model $\mathbf{u}_t = \mathbf{u} + \delta\mathbf{u}$ for the camera measurement, 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 a priori, but the scaling factor can be calculated by applying the Law of Sines. Let's consider the intersection problem in Figure 4.1, the travelled 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 the law of sines on vertex $\mathbf{p}_{n,j}^b$ and $\mathbf{p}_n^f$:

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

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

$\boldsymbol{\epsilon_i}$ can be expressed with $\delta\mathbf{x}$ too:

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

In this case, let's denote the filter state covariance as $E\{\delta\mathbf{x}\delta\mathbf{x}^T\} = \mathbf{P}_x$. The image measurement covariance is modeled with independent white Gaussian noise $E\{\delta\mathbf{u}\delta\mathbf{u}^T\} = \mathbf{V} \in \mathbb{V}^{3\times 3}$. It is crucial to highlight that the LOS and actual measurement covariance are the same if only the principal point is 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 denoted as $E\{\delta\mathbf{x}\delta\mathbf{u}^T\} = \mathbf{P}_{xv}$. Firstly, it could seem a bit strange why a correlation between the state and the measurement is defined and it's connected to the schedule of the algorithm. When a visual measurement is taken the algorithm performs an update on the state first, 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 ($\delta\mathbf{p}_n^f$), and the measurement. I will detail it later, but now just look at the result:

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

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

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

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:

$$\mathbf{P}_{\epsilon_i} = E\{\boldsymbol{\epsilon}_i \boldsymbol{\epsilon}_i^T\} = \begin{bmatrix} \mathbf{L}_x & \rho \left[\mathbf{a}_i\right]_\times \end{bmatrix} \begin{bmatrix} \mathbf{P}_x & \mathbf{P}_{xv} \\ \mathbf{P}_{vx} & \mathbf{V} \end{bmatrix} \begin{bmatrix} \mathbf{L}_x^T \\ \rho \left[\mathbf{a}_i\right]_\times^T \end{bmatrix} \tag{4.9}$$

4.2.2 The optimal solution

In [29], 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 additionally the pixel error is only 2-D, henceforth the approximation $\mathbf{P}_{\epsilon_i}^{-1} \rightarrow \mathbf{P}_{\epsilon_i}^{+}$ will be used, where $^+$ denotes the Moore-Penrose inverse [37]. Returning to (4.10), the next step is to substitute (4.2) in, then rearrange terms containing $\mathbf{p}_n^f$ and neglecting those that do not contain $\mathbf{p}_n^f$ the cost function to minimize:

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

Minimization can be performed by applying the 1^{st} differential condition on (4.11) which 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 (LS) 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 [38].

Since the weighting matrix $\mathbf{P}_{\epsilon_i}^{+}$ is calculated from a covariance matrix which is positive semidefinite, its eigenvalues are real non-negative and its eigenvectors are real. With the help of eigendecomposition [39], $\mathbf{P}_{\epsilon_i}^{+}$ 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:

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

The above equation must be solved in LS terms, which leads to the estimator:

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

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 Recursive LOST

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):

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

Now, everything is known or can be calculated in (4.18) to perform the update on the feature position hence, this formalization of the method is called recursive LOST.

CHAPTER 5

SYNERGIZING LOST AND ESKF FOR ROBUST NAVIGATION

THE applied filter technique and the triangulation approach have been already 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 [28], 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 feature position are not known, then its uncertainty should be taken into account too.

5.1.1 The error propagation from measurement to measurement

The real error propagation is described in (3.17), but the actual error propagation is presented in (3.19b). There is usually more than one prediction step between two measurement updates, therefore extending this equation for N predictions

yields:

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

$$\delta\hat{\mathbf{x}}_{k+N}^- = \left(\prod_{j=0}^{N-1}\mathbf{F}_{x,k+j}\right)\delta\hat{\mathbf{x}}_k = \mathbf{P}\mathbf{F}_x^N\delta\hat{\mathbf{x}}_k, \tag{5.1b}$$

where $^-$ denotes the a priori estimate.

The aforementioned equations yield a crucial insight, that the error propagation can be divided into two components. The first concerns the propagation of the previous a posteriori error state, denoted as $\mathbf{P}\mathbf{F}_x^N\delta\mathbf{x}_k$, while the second is associated with accounting for the integration of Additive White Gaussian Noise (AWGN) within the system, denoted as $\mathbf{S}\mathbf{F}_i^N\mathbf{i}$.

5.1.2 The measurement model

The actual measurement is produced by the nonlinear function $h(.)$ with the true values of the state and the feature position plus an AWGN on the visual measurement hence, the measurement can be constructed as:

$$\begin{aligned}\mathbf{z}_{k+N} &= h(\mathbf{x}_{t,k+N}, \mathbf{p}_{n,t}^f) + \delta\mathbf{u}_{k+N} \\ &\approx h(\mathbf{x}_{k+N}, \mathbf{p}_n^f) + \mathbf{H}_{x,k+N}\delta\mathbf{x}_{k+N} + \mathbf{H}_{f,k+N}\delta\mathbf{p}_n^f + \delta\mathbf{u}_{k+N},\end{aligned} \tag{5.2}$$

where the linearized error approximation was utilized, and $\mathbf{H}_{f,k+N}$ is the pixel measurement Jacobian with respect to the feature position which can be calculated by taking the derivative of (4.1), that yields:

$$\frac{\partial\mathbf{p}_c^f}{\partial\mathbf{p}_n^f} = \mathbf{R}_{CN} \tag{5.3}$$

To form the residual, the measurements have to be predicted ($\hat{\mathbf{z}}_{k+N}$), but at this point only the nominal states ($\mathbf{x}_{k+N}$, $\mathbf{p}_n^f$) and the approximated error states

$(\delta\hat{\mathbf{x}}^{-}_{k+N}, \delta\hat{\mathbf{p}}^{f}_{n})$ are available:

$$\hat{\mathbf{z}}_{k+N} = h(\mathbf{x}_{k+N}, \mathbf{p}^{f}_{n}) + \mathbf{H}_{x,k+N}\delta\hat{\mathbf{x}}^{-}_{k+N} + \mathbf{H}_{f,k+N}\delta\hat{\mathbf{p}}^{f}_{n} \tag{5.4}$$

The Kalman filter forms the new estimation $(\delta\hat{\mathbf{x}}_{k+N})$ as a composition of the a priori state and the residual $\mathbf{r}_{k+N} = \mathbf{z}_{k+N} - \hat{\mathbf{z}}_{k+N}$:

$$\begin{aligned}\delta\hat{\mathbf{x}}_{k+N} &= \delta\hat{\mathbf{x}}^{-}_{k+N} + \mathbf{K}_g(\mathbf{z}_{k+N} - \hat{\mathbf{z}}_{k+N}) \\ &= \delta\hat{\mathbf{x}}^{-}_{k+N} + \mathbf{K}_g(\mathbf{H}_{x,k+N}\delta\tilde{\mathbf{x}}^{-}_{k+N} + \mathbf{H}_{f,k+N}\delta\tilde{\mathbf{p}}^{f}_{n} + \delta\mathbf{u}_{k+N}),\end{aligned} \tag{5.5}$$

where $\delta\tilde{\mathbf{x}}^{-}_{k+N} = \delta\mathbf{x}_{k+N} - \delta\hat{\mathbf{x}}^{-}_{k+N}$ expresses the a priori error in the error state estimation, and $\delta\tilde{\mathbf{p}}^{f}_{n} = \delta\mathbf{p}^{f}_{n} - \delta\hat{\mathbf{p}}^{f}_{n}$. Then the error of the estimation can be expressed:

$$\begin{aligned}\delta\tilde{\mathbf{x}}_{k+N} = \delta\mathbf{x}_{k+N} - \delta\hat{\mathbf{x}}_{k+N} =& \mathbf{P}\mathbf{F}^{N}_{x}\delta\tilde{\mathbf{x}}_{k} + \mathbf{S}\mathbf{F}^{N}_{i}\mathbf{i} \\ &- \mathbf{K}_g(\mathbf{H}_{x,k+N}\delta\tilde{\mathbf{x}}^{-}_{k+N} + \mathbf{H}_{f,k+N}\delta\tilde{\mathbf{p}}^{f}_{n} + \delta\mathbf{u}_{k+N})\end{aligned} \tag{5.6}$$

Now, substituting (5.1a) and (5.1b) into $\tilde{\mathbf{x}}^{-}_{k+N}$, then the current error in the a posteriori estimate can be expressed with the previous a posteriori error:

$$\delta\tilde{\mathbf{x}}_{k+N} = \overbrace{(\mathbf{I} - \mathbf{K}_g\mathbf{H}_{x,k+N})}^{\mathbf{T}_x}\left(\mathbf{P}\mathbf{F}^{N}_{x}\delta\tilde{\mathbf{x}}_{k} + \mathbf{S}\mathbf{F}^{N}_{i}\mathbf{i}\right) - \overbrace{\mathbf{K}_g\mathbf{H}_{f,k+N}}^{\mathbf{T}_f}\delta\tilde{\mathbf{p}}^{f}_{n} - \mathbf{K}_g\delta\mathbf{u}_{k+N} \tag{5.7}$$

5.1.3 The optimal solution for the Kalman gain

Before I discuss in detail the optimal solution, let's establish a few mathematical notations. The propagated covariance of the state can be calculated based on (3.19c) and is denoted with $E\{\delta\tilde{\mathbf{x}}^{-}_{k+N}\delta\tilde{\mathbf{x}}^{-}_{k+N}{}^{T}\} = \mathbf{P}^{-}_{x,k+N}$. The feature covariance is denoted with $E\{\delta\tilde{\mathbf{p}}^{f}_{n}\delta\tilde{\mathbf{p}}^{f}_{n}{}^{T}\} = \mathbf{P}_f$, and the measurement covariance is still $\mathbf{V}$.

To give an optimal solution for the Kalman gain the a posteriori covariance should be expressed, but first, let's take some considerations. Both the state and the feature position are optimized with the visual measurements further-

more, they are utilized in each other calculations, which leads to a correlation between $\delta\mathbf{x}$ and $\delta\mathbf{p}_n^f$. The correlation of their errors can be described with the cross-covariance matrix, which is denoted as $E\{\delta\tilde{\mathbf{x}}_{k+N}\delta\tilde{\mathbf{p}}_n^{f^T}\} = \mathbf{P}_{xf,k+N}$. An important property of the cross-covariance matrices is $\mathbf{P}_{xf} = \mathbf{P}_{fx}^T$.

From now on, I will omit the indexing, because it is clear to which time step the calculations refer. Let's express the a posteriori covariance of the state considering the cross-covariance:

$$\begin{aligned}\mathbf{P}_x &= E\{\delta\tilde{\mathbf{x}}_{k+N}\delta\tilde{\mathbf{x}}_{k+N}^T\} \\ &= \mathbf{T}_x\mathbf{P}_x^-\mathbf{T}_x^T + \mathbf{T}_f\mathbf{P}_f\mathbf{T}_f^T + \mathbf{K}_g\mathbf{V}\mathbf{K}_g^T - \mathbf{T}_x\mathbf{P}_{xf}\mathbf{T}_f^T - \mathbf{T}_f\mathbf{P}_{fx}\mathbf{T}_x^T\end{aligned} \tag{5.8}$$

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

$$\begin{aligned}\mathbf{P}_x =& \mathbf{P}_x^- - \mathbf{K}_g\left(\mathbf{H}_x\mathbf{P}_x^- + \mathbf{H}_f\mathbf{P}_{fx}\right) - \overbrace{\left(\mathbf{P}_x^-\mathbf{H}_x^T + \mathbf{P}_{xf}\mathbf{H}_f^T\right)}^{E}\mathbf{K}_g^T \\ &+ \mathbf{K}_g\underbrace{\left(\mathbf{H}_x\mathbf{P}_x^-\mathbf{H}_x^T + \mathbf{H}_f\mathbf{P}_f\mathbf{H}_f^T + \mathbf{V} + \mathbf{H}_x\mathbf{P}_{xf}\mathbf{H}_f^T + \mathbf{H}_f\mathbf{P}_{fx}\mathbf{H}_x^T\right)}_{\mathbf{D}}\mathbf{K}_g^T\end{aligned} \tag{5.9}$$

Important to highlight the property $\mathbf{E} = \mathbf{E}^T$ which derives from the positive semidefinite property of the covariance matrices. Finally, the optimal Kalman gain can be determined from the above equation, because $\mathrm{tr}(\mathbf{P}_x)$ is a function of $\mathbf{K}_g$ and $\mathbf{K}_g$ is the only unknown variable, and we request to minimize the trace of the a posteriori covariance with respect to $\mathbf{K}_g$. The partial derivative of the trace is easily given using matrix calculus rules [40]:

$$\frac{\partial\mathrm{tr}(\mathbf{P}_x)}{\partial\mathbf{K}_g} = -\mathbf{E}^T - \mathbf{E} + 2\mathbf{K}_g\mathbf{D} = 0 \tag{5.10}$$

Rearranging the terms for $\mathbf{K}_g$ the optimal solution is given as:

$$\mathbf{K}_g = \mathbf{E}\mathbf{D}^{-1} \tag{5.11}$$

5.2 Cross-covariance estimation

The method for cross-covariance modeling is utilized from [41], 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:

$$\mathbf{y} = \boldsymbol{\mu} + \mathbf{L}\mathbf{z}, \quad \mathbf{P} = \mathbf{L}\mathbf{L}^T, \tag{5.12}$$

where $\mathbf{L} \in \mathbb{R}^{p \times m}$ matrix of rank m and $\mathbf{z}$ is a vector with m independent identically distributed standard normal variable [42]. Both [43, 44] propose the usage of the Lower-Upper (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.

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

$$\begin{aligned} \mathbf{y}_1 &= \boldsymbol{\mu}_1 + \mathbf{L}_1\mathbf{z}_1 \\ \mathbf{y}_2 &= \boldsymbol{\mu}_2 + \mathbf{L}_2\left(\rho\mathbf{z}_1 + \sqrt{1-\rho^2}\mathbf{z}_2\right) \end{aligned} \tag{5.13}$$

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 that $\mathbf{z}_1$ 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 $E\{\mathbf{z}_1\mathbf{z}_2^T\}, E\{\mathbf{z}_2\mathbf{z}_1^T\} = \mathbf{0}$. Using these properties the cross-covariance can be calculated as:

$$\begin{aligned} E\{(\mathbf{y}_1 - \boldsymbol{\mu}_1)(\mathbf{y}_2 - \boldsymbol{\mu}_2)^T\} &= E\{\mathbf{L}_1\mathbf{z}_1(\rho\mathbf{z}_1^T\mathbf{L}_2^T + \sqrt{1-\rho^2}\mathbf{z}_2^T\mathbf{L}_2^T)\} \\ &= \rho\mathbf{L}_1E\{\mathbf{z}_1\mathbf{z}_1^T\}\mathbf{L}_2^T + \sqrt{1-\rho^2}\mathbf{L}_1E\{\mathbf{z}_1\mathbf{z}_2^T\}\mathbf{L}_2^T \\ &= \rho\mathbf{L}_1\mathbf{L}_2 \end{aligned} \tag{5.14}$$

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 covariance matrices. It can be always done because covariance matrices are positive semidefinite. The decomposition 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. The covariance matrices are projected into the pixel space with their measurement matrices $\mathbf{H}_x$ and $\mathbf{H}_f$ for the state and the feature position respectively. Applying the eigenvalue decomposition on the projected covariance matrices yields:

$$\begin{aligned} \mathbf{H}_x\mathbf{P}_x\mathbf{H}_x^T &= \overbrace{\left(\mathbf{V}_x\sqrt{\mathbf{D}_x}\right)}^{\mathbf{L}_x}\left(\mathbf{V}_x\sqrt{\mathbf{D}_x}\right)^T \\ \mathbf{H}_f\mathbf{P}_f\mathbf{H}_f^T &= \underbrace{\left(\mathbf{V}_f\sqrt{\mathbf{D}_f}\right)}_{\mathbf{L}_f}\left(\mathbf{V}_f\sqrt{\mathbf{D}_f}\right)^T \end{aligned} \tag{5.15}$$

Using the results from above the cross-covariance can be calculated in the pixel space as:

$$\mathbf{H}_x\mathbf{P}_{xf}\mathbf{H}_f^T = \mathbf{L}_x\mathbf{L}_f^T \tag{5.16}$$

Finally, (5.11) requires the determination of the term $\mathbf{P}_{xf}\mathbf{H}_f^T$, therefore the cross-covariance in the pixel state has to be transformed back into the state space at the side of the state. It can be performed with the state measurement matrix if a certain condition is met. $\mathbf{H}_x \in \mathbb{R}^{2k\times 15}$, where k is the number of features involved in the current update although, it contains a zero block matrix because of (3.26):

$$\mathbf{H}_x = \begin{bmatrix}\mathbf{A}_x & \mathbf{0}\end{bmatrix}, \quad \mathbf{A}_x \in \mathbb{R}^{2k\times 6}, \quad \mathbf{0} \in \mathbb{R}^{2k\times 9} \tag{5.17}$$

If $k \geq 3$ condition is met, then $\mathbf{H}_x^+ \mathbf{H}_x = \begin{bmatrix} \mathbf{I}_6 & \mathbf{0}_{6\times 9} \\ \mathbf{0}_{9\times 6} & \mathbf{0}_{9\times 9} \end{bmatrix}$, hence:

$$\mathbf{P}_{xf} \mathbf{H}_f^T = \mathbf{H}_x^+ \mathbf{L}_x \mathbf{L}_f^T \tag{5.18}$$

This short detour concludes that at least three features should be involved in the updates to be able to perform the calculations.

CHAPTER 6

DEVELOPMENT

In this chapter, I explain how I built the simulation environment and 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 the System Control Laboratory (SCL) of the Institute for Computer Science and Control (SZTAKI, Számítástechnikai és Automatizálási Kutatóintézet). The simulation environment is based on the Matlab/Simulink 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 (World Geodetic System) convention.

During the development, I had to integrate the algorithm into the simulation, which is mainly based on a user-defined Matlab function block. It runs the ESKF at 50 Hz, and the new visual measurements are available at 10 Hz. Initially, the visual measurements were generated from known feature coordinates based on the pinhole equations, because this block gets the true state of the aircraft as input, and this data was used to project the feature points onto the camera screen. Later synthetic images were generated in the Unreal Car Learning to Act (CARLA) simulation environment by the Computational Optical Sensing and Processing

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

Laboratory (COSPL) of SZTAKI. In this case, a real image processing algorithm, a Kanade-Lucas-Tomasi (KLT) feature tracker is used to track features across image frames [45], this algorithm was also developed by SZTAKI.

6.1.1 Noise and bias calibration

The simulation required setting five different noise values and the sensor biases. Four of the noises are connected to process noise that was defined in (3.12), and an additional one models the visual measurement error σ_u. The measurement noises σ_{η_ω} and σ_{η_a} were set slightly higher, than the noises of a real IMU, and these noises were simulated with the *randn* function of Matlab. The variation of biases over time are temperature-dependent parameters, but the current solution doesn't take into account this effect, therefore they were set to low values. The exact dispersion parameters and sensor biases are presented in Table 6.1 and were calculated for 50Hz sampling frequency.

Name	Notation	Value	UoM
Angular velocity measurement noise	σ_{η_ω}	1	$\frac{^\circ}{\mathrm{s}}$
Acceleration measurement noise	σ_{η_a}	0.1	$\frac{\mathrm{m}}{\mathrm{s}^2}$
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
Accelerometer bias	$\boldsymbol{\beta}_a$	$\begin{bmatrix}0.5 & -0.4 & 0.4\end{bmatrix}$	$\frac{\mathrm{m}}{\mathrm{s}^2}$
Gyroscope bias	$\boldsymbol{\beta}_\omega$	$\begin{bmatrix}0.04 & 0.05 & -0.05\end{bmatrix}$	$\frac{\mathrm{rad}}{\mathrm{s}}$

Table 6.1: Noise summary

6.1.2 Camera configuration

For simulation purposes, I utilized the intrinsic parameters of a Basler acA 2040 camera, but the synthetic camera images were generated for a Basler dcA 1280 camera. The parameters of the utilized cameras are introduced in Table 1 of [46], but the focal lengths were calculated with a slightly different FOV angle, than the presented ones, hence, I introduce the employed parameters for each camera that is presented in Table 6.2. I also included the constant extrinsic parameters of the cameras for example the camera position in the body frame ($\mathbf{p}_b^c$) and the camera tilt angle (β).

Type	Basler dcA 1280	Basler acA 2040
Focal length [px]	1108.513	1177.5
Principal point X [px]	640	1024
Principal point Y [px]	480	768
Image width [px]	1280	2048
Image height [px]	960	1536
Camera position [m]	$\begin{bmatrix}1 & 0 & 0\end{bmatrix}^T$	$\begin{bmatrix}0 & 0 & 0\end{bmatrix}^T$
Camera tilt angle [$\circ$]	-20	-45

Table 6.2: Camera parameters

6.1.3 Feature point selection

First of all, I want to clarify that the feature point selection is only relevant when I had to select the tracked features, but when a real image processing algorithm is included in the system the feature selection is its responsibility. The feature point selection 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 [47], they proposed that in Visual-Inertial Odometry (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 [48], the optical flow equations were presented in (1) and it shows that the derivative of the 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 the 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.

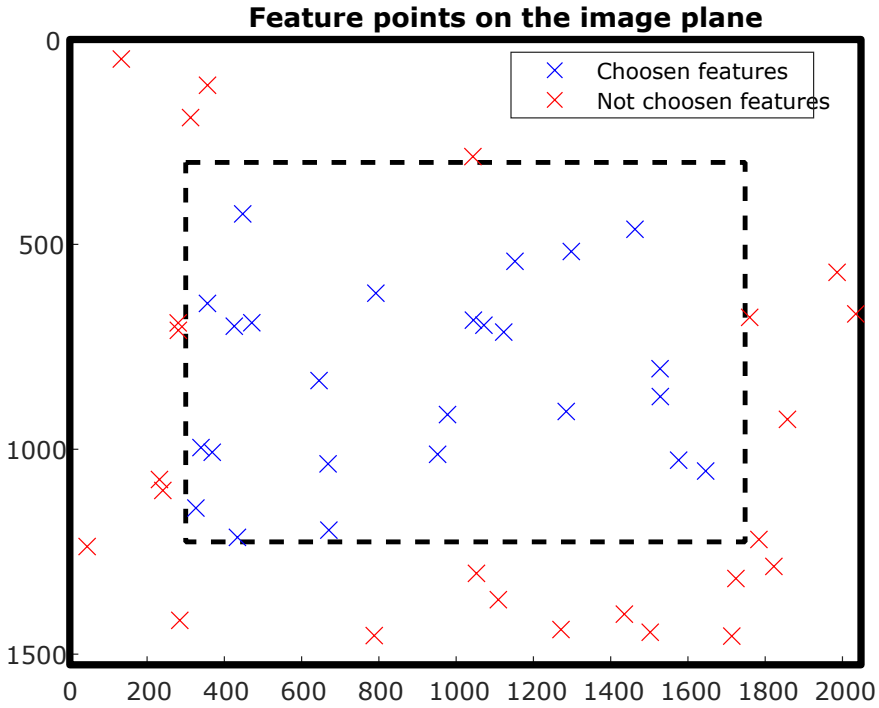


Figure 6.1: Camera image example

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 tracked features. This is 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.

6.2 Simulation results

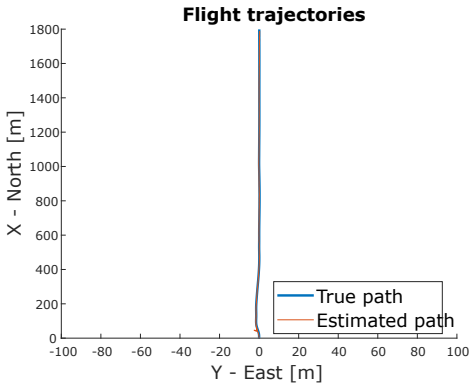
Once the parameters have been configured, only the flight trajectory needs to be defined which is described with waypoints that the aircraft must pass through.

The similarity in all cases is that the LOST algorithm uses true values of the camera position and orientation to estimate feature positions until 30s, from the 30s the true state is not available and the LOST starts to utilize the ESKF estimations. It's advantageous because the ESKF has time to converge to the true state.

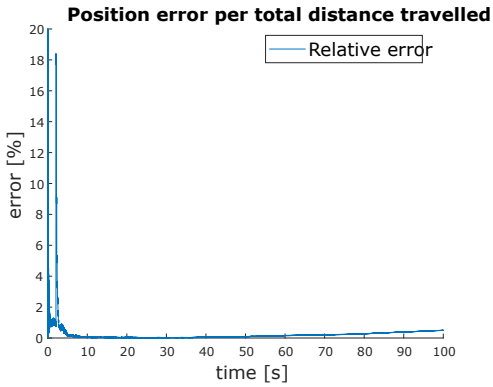
A tracked feature is included in the ESKF measurement update if the square root of its largest eigenvalue is less than 10m with the 1-σ rule. This is a beneficial choice because the eigenvalues of a 3-D vector's covariance matrix represent an ellipsoid in the 3-D space and the square root of the eigenvalues are the dispersion along each axes and they describe the length of the semi-axes. The mean of the vector points to the center of the ellipsoid and using the standard deviations, an interval can be defined in which, for a given confidence level, the endpoint of the vector can be found. The k-σ rule is also called as 68–95–99.7 rule because a Gaussian distributed variable is in the interval $[\mu - k\sigma; \mu + k\sigma]$ with the confidence level of 68%, 95%, or 99.7% with values of $k = 1, 2, 3$ respectively. I have chosen the 1-σ rule because it offers a quite high confidence interval, besides that it is the less strict constraint. I have also tried other values of k, but the results were not better because in these cases, the ESKF update condition is not met for a longer period which leads to divergence sooner.

6.2.1 Straight and level flight path

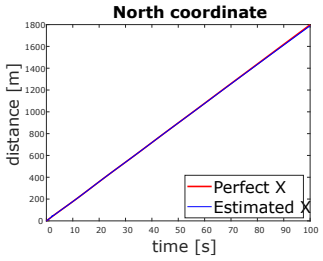
The actual and estimated flight trajectory considering only the 2-D position (X-Y plane) is shown in Figure 6.2a, and the accumulated error as a function of the distance travelled is illustrated in Figure 6.2b.



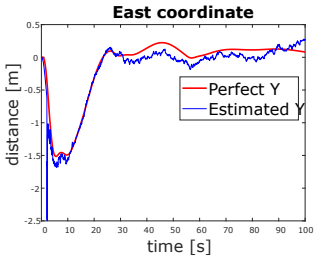
(a) Straight and level flight path



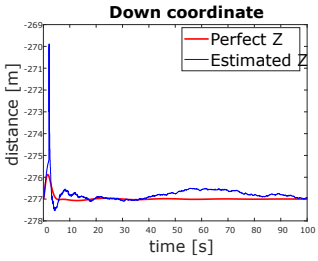
(b) Straight and level flight path relative error



(a) Straight and level path—X

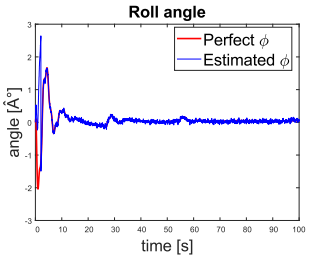


(b) Straight and level path—Y

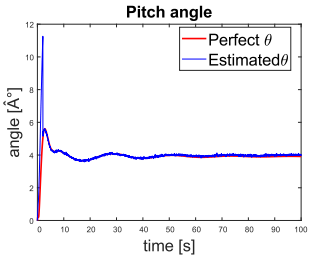


(c) Straight and level path—Z

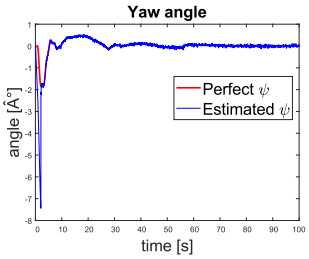
Figure 6.3: Straight and level path—position



(a) Straight and level path—roll

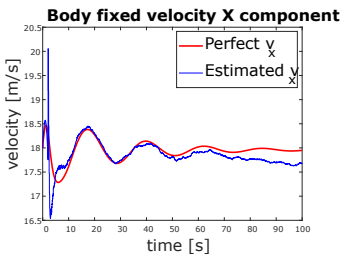


(b) Straight and level path—pitch

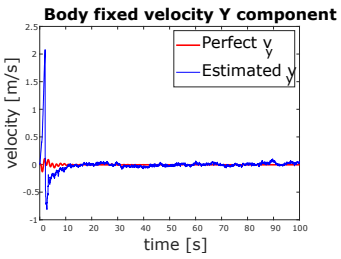


(c) Straight and level path—yaw

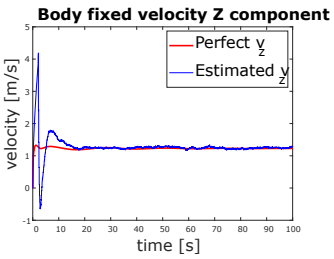
Figure 6.4: Straight and level path—orientation



(a) Straight and level path—$v_{b,x}$

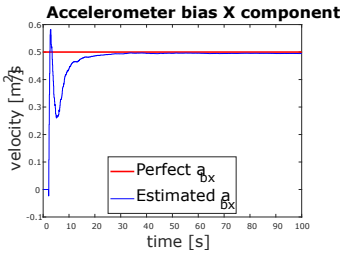


(b) Straight and level path—$v_{b,y}$

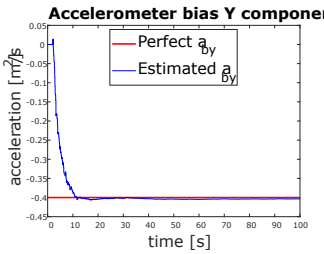


(c) Straight and level path—$v_{b,z}$

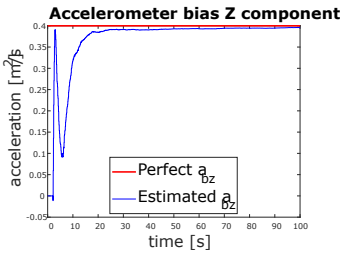
Figure 6.5: Straight and level path—velocity



(a) Straight and level path—$a_{b,x}$

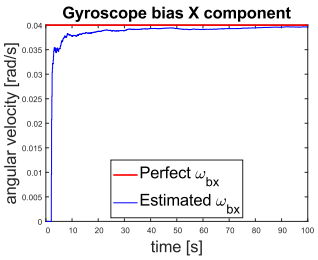


(b) Straight and level path—$a_{b,y}$

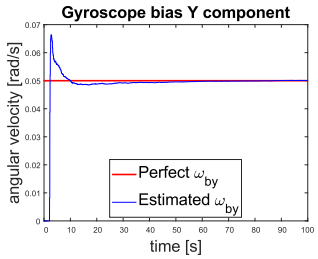


(c) Straight and level path—$a_{b,z}$

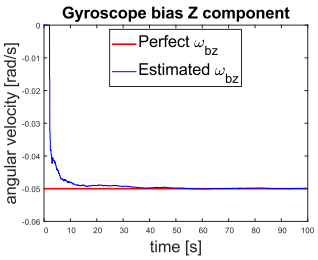
Figure 6.6: Straight and level path—acceleration bias



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



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



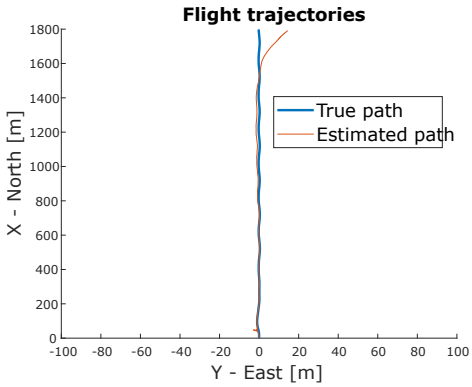
(c) Straight and level path—$\omega_{b,z}$

Figure 6.7: Straight and level path—angular velocity bias

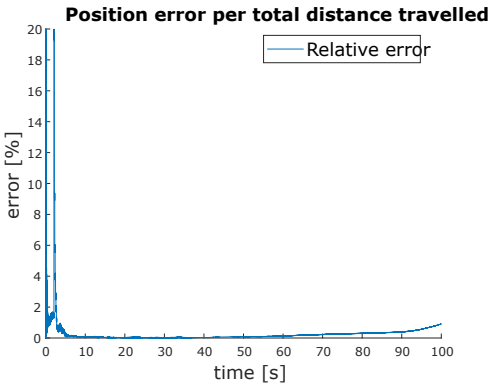
In summary, the charts show that in the case of straight and level flight, the developed algorithm performs quite well. The accumulated 2-D error remains under 1% of the travelled distance at the end of the simulation. The estimates remain close to the true values for the whole simulation. I want to emphasize that at the beginning of the simulation, the position, orientation, and velocity are not estimated well, but this is not surprising because the LOST estimates have not converged sufficiently until, furthermore, the biases are far away from their true values. After the LOST estimates and the biases have converged the ESKF estimates return close to the true values and remain there.

6.2.2 Straight and descending flight path

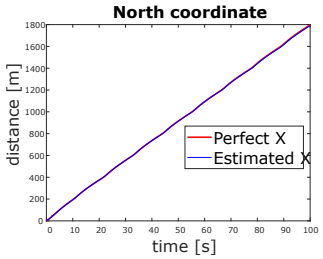
In the case of straight and descending flight trajectory, the estimated and the actual 2-D position is shown in Figure 6.8a, and the accumulated error as a function of the distance travelled is depicted in Figure 6.8b. In this simulation setup, I set a flight path where the aircraft descended 10m after every travelled 200m.



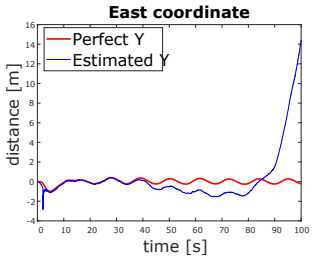
(a) Straight and descending path



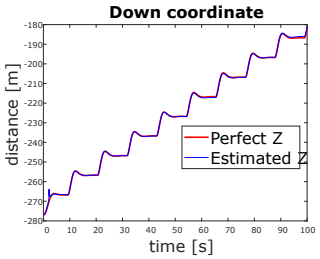
(b) Straight and descending path relative error



(a) Straight and descending path—X

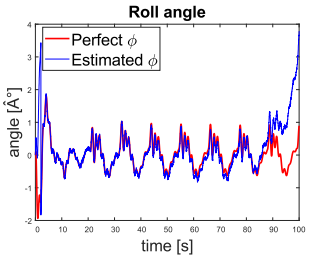


(b) Straight and descending path—Y

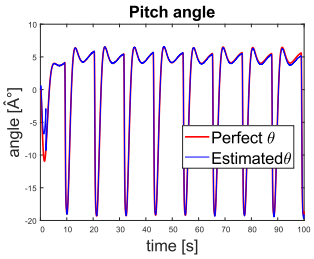


(c) Straight and descending path—Z

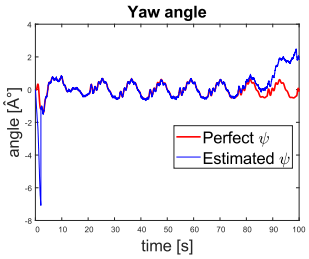
Figure 6.9: Straight and descending path—position



(a) Straight and descending path—roll

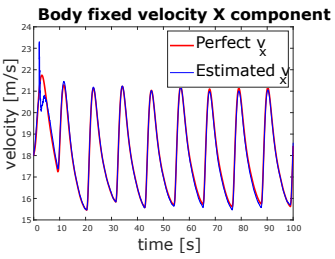


(b) Straight and descending path—pitch

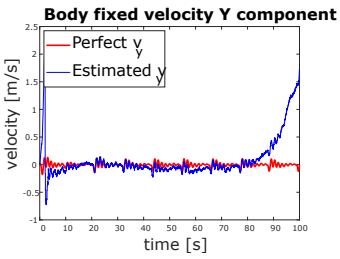


(c) Straight and descending path—yaw

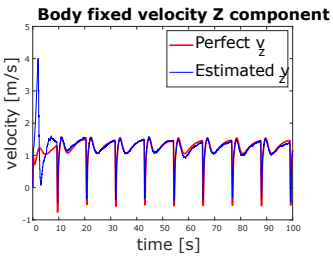
Figure 6.10: Straight and descending path—Euler angles



(a) Straight and descending path—v_x

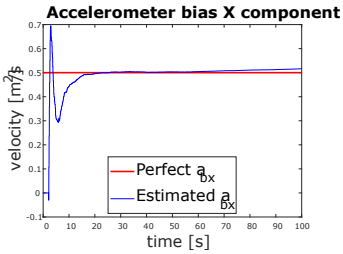


(b) Straight and descending path—v_y

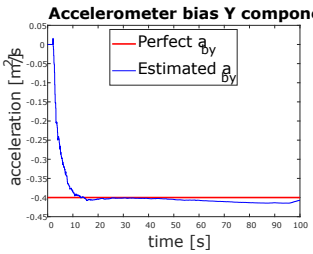


(c) Straight and descending path—v_z

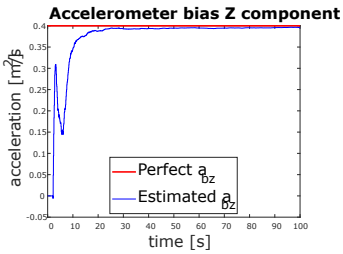
Figure 6.11: Straight and descending path—velocity



(a) Straight and descending path—$a_{b,x}$

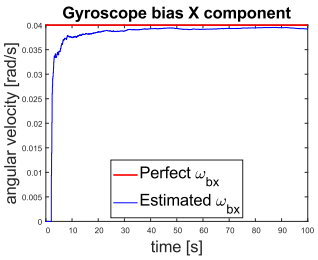


(b) Straight and descending path—$a_{b,y}$

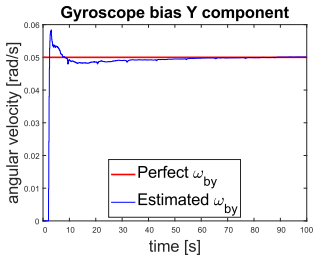


(c) Straight and descending path—$a_{b,z}$

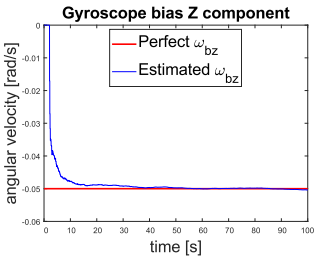
Figure 6.12: Straight and descending path—acceleration bias



(a) Straight and descending path—$\omega_{b,x}$



(b) Straight and descending path—$\omega_{b,y}$



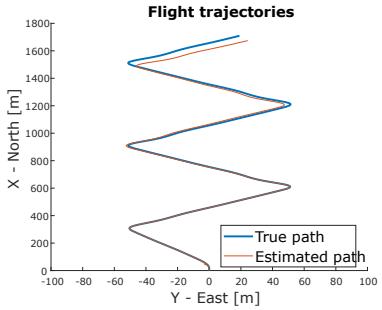
(c) Straight and descending path—$\omega_{b,z}$

Figure 6.13: Straight and descending path—angular velocity bias

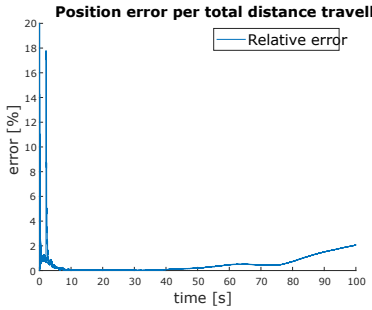
In brief, the algorithm performed also well in the case of the straight and descending flight path. The accumulated 2-D error remains under 2% of the travelled distance at the end of the simulation. At the beginning of the simulation, the experience is the same as in the case of the straight and level flight path. Remarkably, the Y, roll, yaw, and the Y component of the velocity start to degrade around 80s.

6.2.3 Zig-zag flight path

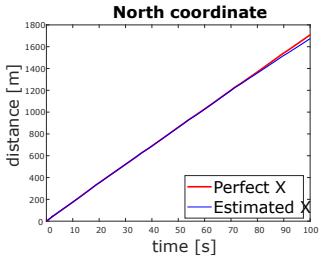
In the case of zig-zag and descending flight trajectory, the estimated and the actual 2-D position is illustrated in Figure 6.8a and the accumulated error as a function of the distance travelled is depicted in Figure 6.8b. In this simulation setup, I set a flight path where the aircraft went between -50m and 50m in the Y direction at every travelled 300m along the X axis.



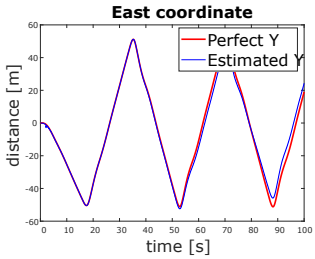
(a) Zig-zag and level path



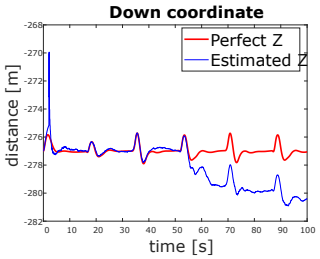
(b) Zig-zag and level path relative error



(a) Zig-zag and level path—X

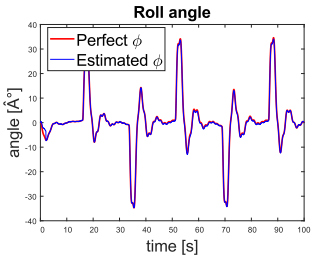


(b) Zig-zag and level path—Y

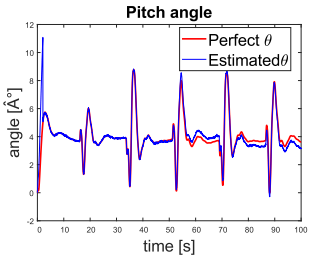


(c) Zig-zag and level path—Z

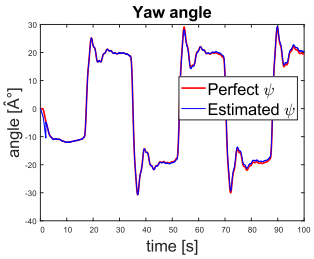
Figure 6.15: Zig-zag and level path—position



(a) Zig-zag and level path—roll

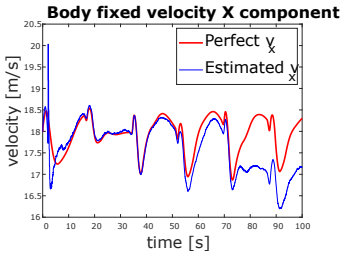


(b) Zig-zag and level path—pitch

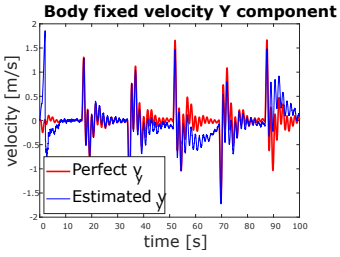


(c) Zig-zag and level path—yaw

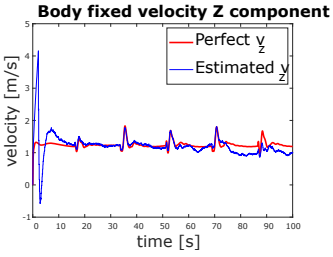
Figure 6.16: Zig-zag and level path—Euler angles



(a) Zig-zag and level path—v_x

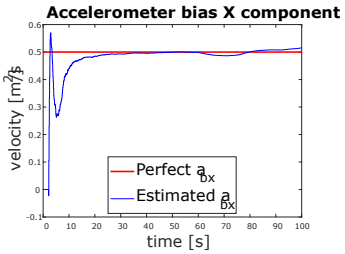


(b) Zig-zag and level path—v_y

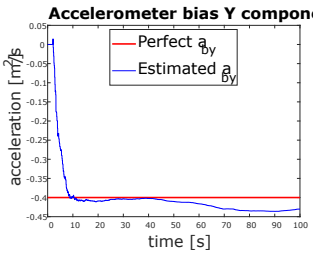


(c) Zig-zag and level path—v_z

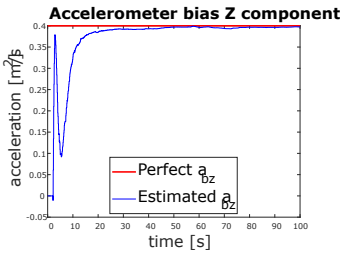
Figure 6.17: Zig-zag and level path—velocity



(a) Zig-zag and level path—$a_{b,x}$

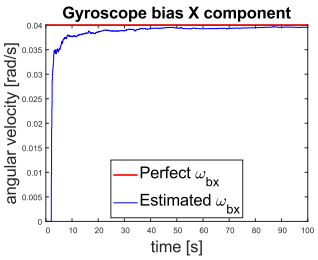


(b) Zig-zag and level path—$a_{b,y}$

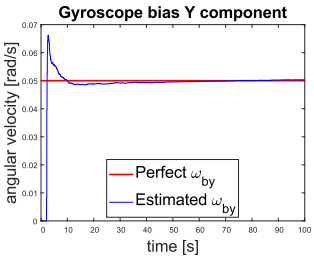


(c) Zig-zag and level path—$a_{b,z}$

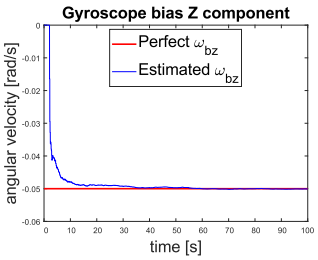
Figure 6.18: Zig-zag and level path—acceleration bias



(a) Zig-zag and level path—$\omega_{b,x}$



(b) Zig-zag and level path—$\omega_{b,y}$



(c) Zig-zag and level path—$\omega_{b,z}$

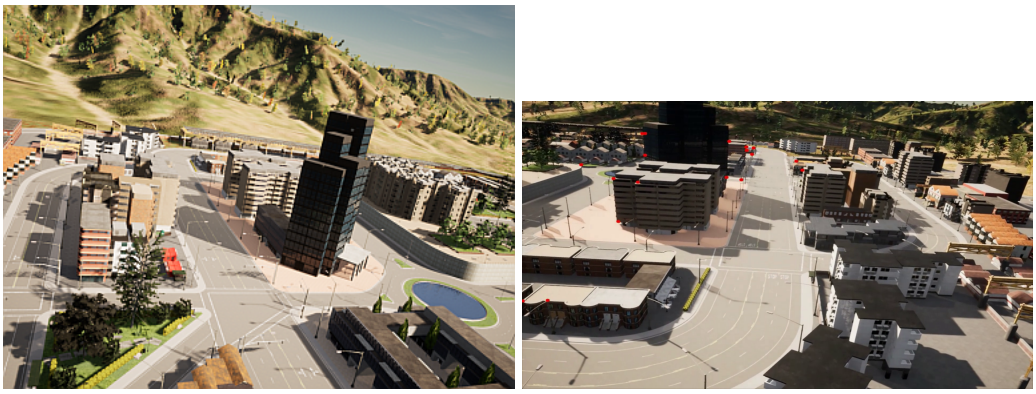
Figure 6.19: Zig-zag and level path—gyroscope bias

To sum up, in this case, the accumulated error is the highest. Probably this is because the aircraft changes its direction regularly therefore the features are visible for a shorter period, than in the previous two cases, but the results are still acceptable. The accumulated 2-D error remains under 3% of the travelled distance at the end of the simulation. Interestingly, in this case, the Z and the X component of the velocity starts to degrade around 55–70s.

6.2.4 Synthetic images based simulation

Synthetic images were produced within the Unreal CARLA simulation environment, a platform built upon the open-source CARLA simulator dedicated to autonomous driving research. In Figure 6.20a, an example is presented of the generated images which illustrate an urban environment. In Figure 6.20b, an example is shown of the detected corners by the KLT algorithm.

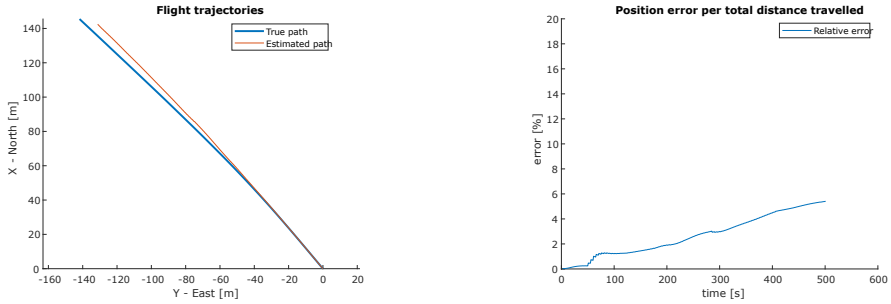
In this case, the estimated and the actual 2-D position is shown in Figure 6.21a, and the accumulated error as a function of the distance travelled is presented in



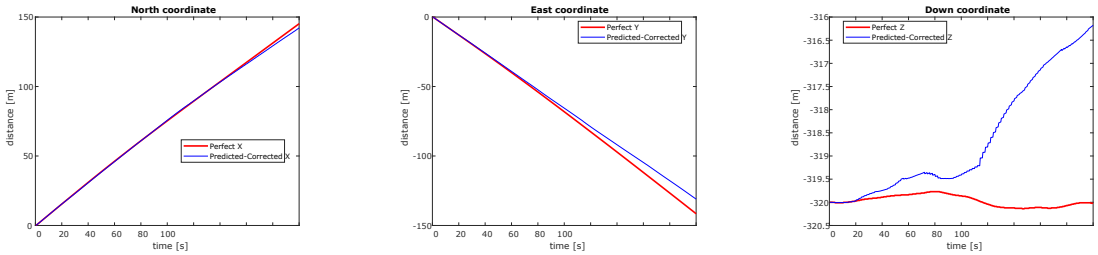
(a) Example of a generated image **(b)** Example of the detected corners

Figure 6.20: Synthetic images

Figure 6.21b. I want to highlight that in this case, I only had 10s of data to run the simulation.

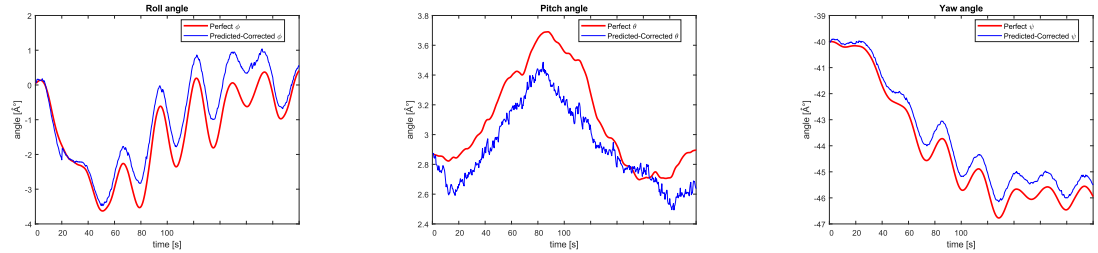


(a) The flight path of CARLA-based simulation **(b)** Relative error of CARLA-based simulation



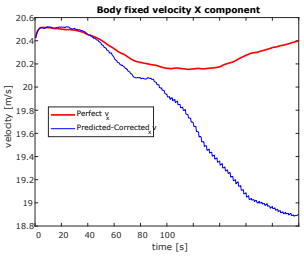
(a) CARLA-based—X **(b)** CARLA-based—Y **(c)** CARLA-based—Z

Figure 6.22: CARLA-based—position

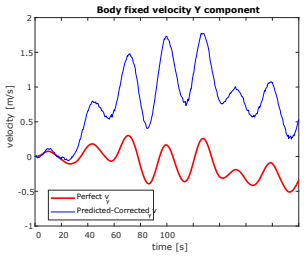


(a) CARLA-based—roll **(b)** CARLA-based—pitch **(c)** CARLA-based—yaw

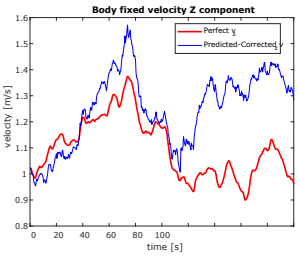
Figure 6.23: CARLA-based—Euler angles



(a) CARLA-based—v_x

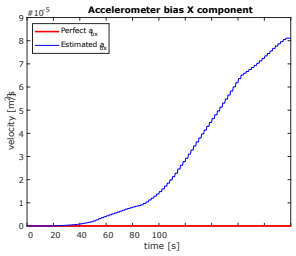


(b) CARLA-based—v_y

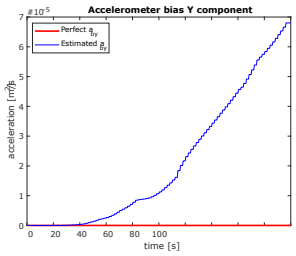


(c) CARLA-based—v_z

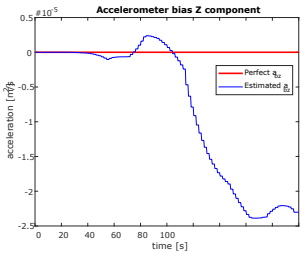
Figure 6.24: CARLA-based—velocity



(a) CARLA-based—$a_{b,x}$

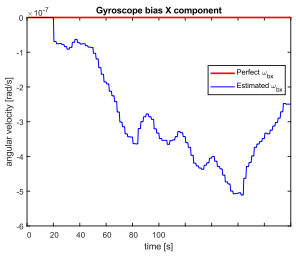


(b) CARLA-based—$a_{b,y}$

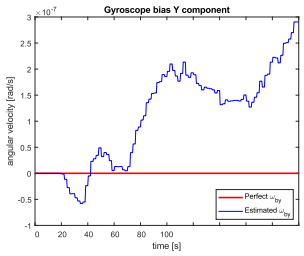


(c) CARLA-based—$a_{b,z}$

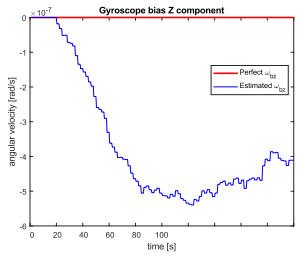
Figure 6.25: CARLA-based—acceleration bias



(a) CARLA-based—$\omega_{b,x}$



(b) CARLA-based—$\omega_{b,y}$



(c) CARLA-based—$\omega_{b,z}$

Figure 6.26: CARLA-based—gyroscope bias

In this case, all of the position and velocity values are degraded significantly through time, but the angles remain close to the true values. It's important to note that this was the first try to use this algorithm with a real image processing algorithm, hence, further developments and improvements are needed to achieve better results. Moreover, the ESKF estimates did not have time to converge, therefore, the results are not as good as in the previous cases. The accumulated 2-D error remains under 6% of the travelled distance at the end of the simulation.

6.2.5 Root mean square error (RMSE)

RMSE is a commonly used metric for evaluating the performance of regression models or estimating the quality of predictions. It is a remarkable approach to formalize and numerically express the error between ideal a and observed $\hat{a}$ values. Mathematically, it can be expressed as:

$$RMSE(a) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{a} - a)^2} \tag{6.1}$$

As a final point of the performance analysis, the RMSE of the ESKF states are presented in Table 6.3.

Variable	Straight& level	Straight& descending	Zig-zag& level	CARLA
X	3.19m	3.01m	10.76m	0.61m
Y	0.14m	2.71m	2.97m	4.01m
Z	0.35m	0.34m	1.77m	1.1m
ϕ	20.52°	38.75°	25.39°	26.87°
θ	29.46°	30.56°	31.62°	17.37°
ψ	26.82°	44.94°	36.87°	13.84°
v_x	0.204m/s	0.202m/s	0.546m/s	0.383m/s
v_y	0.173m/s	0.374m/s	0.269m/s	1.209m/s
v_z	0.315m/s	0.284m/s	0.333m/s	0.118m/s
$a_{b,x}$	0.074m/s^2	0.085m/s^2	0.087m/s^2	$\approx$ 0m/s^2
$a_{b,y}$	0.083m/s^2	0.074m/s^2	0.077m/s^2	$\approx$ 0m/s^2
$a_{b,z}$	0.006m/s^2	0.078m/s^2	0.083m/s^2	$\approx$ 0m/s^2
$\omega_{b,x}$	0.35°/s	0.35°/s	0.35°/s	$\approx$ 0°/s
$\omega_{b,y}$	0.43°/s	0.43°/s	0.43°/s	$\approx$ 0°/s
$\omega_{b,z}$	0.43°/s	0.44°/s	0.43°/s	$\approx$ 0°/s

Table 6.3: RMSE values

CHAPTER 7

SUMMARY

FINALLY, in this chapter, the results of the filter are evaluated, and the conclusions are drawn.

7.1 Spoofing detection

UAVs, also known as drones, have witnessed a significant increase in utilization across both military and civil sectors. Their applications range from surveillance and reconnaissance to cargo services, agriculture, and monitoring of critical infrastructures. However, the rapid growth in demand for UAVs and the pressure to reduce size, weight, power, and cost (SWaP-C) has often led to the neglect of security aspects, introducing serious safety and security threats.

One of the major threats faced by UAVs is GPS spoofing. Since UAVs rely heavily on GPS for positioning and navigation, they are susceptible to GPS jamming and spoofing attacks. These attacks can misdirect or even completely hijack drones for malicious purposes. This vulnerability is not limited to UAVs but extends to other GPS-dependent platforms, including manned aircraft, ground vehicles, and cellular systems. Hence, in the literature, there have been already proposed several methods to detect GPS spoofing attacks [49, 50].

In this paper, a GPS-free method was presented to estimate the position of a UAV using only the measurements of an IMU and a camera. The method is based on the VIO algorithm, which is a state-of-the-art method for estimating the pose of the vehicle. The algorithm can be run without any GPS measurements, and this makes it robust against GPS spoofing attacks. With the usage of the algorithm freed from GPS measurements, the UAV can estimate its position and it can be

used to compare the estimated position with the GPS measurements. If the difference between the two positions is significant in a short period, then the UAV can be under a GPS spoofing attack.

7.2 Conclusion

All in all, the developed algorithm showed great potential, although, further development is needed to make it more robust and reliable. The following steps are recommended to be taken in the future:

1. It is recommended to test and fine-tune the algorithm in the simulation environment, besides multiple flight trajectories where the visual measurements and feature selection are produced by the image processing algorithm.

2. The algorithm completly lack of a mapping algorithm which is crucial to estimate the global position of the UAV besides longer flights.

3. Prior to deploying the algorithm in a real-world setting, it is advisable to assess its performance within a simulation environment that incorporates the parameters of the IMU, accounting for the conditions anticipated in actual usage.

APPENDIX A

QUATERNION OPERATIONS

Most of the following formulas can be found in [27].

A.1 Basic definitions

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:

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

An interesting fact is that the unit length quaternion conjugate and inverse are equal, akin to the unitary property of rotation matrices, leading to the equivalence of the transpose and inverse matrices.

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

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

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):

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

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

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

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

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

A.2 Rotation vector to quaternion formula

The $\mathbf{v} = \theta\mathbf{u},\ \mathbf{v} \in \mathfrak{so}(3)$ 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.3 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 not fixed (e.g. body) 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 the quaternion and use results of the above 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] $\mathbf{q}_1 = \begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix}^T$

It is important to highlighting that $\omega_{\mathcal{L}}$ is expressing the rotation of the inertial frame compared to the body frame, but the gyroscopes usually measures the rotation of the body frame compared to the inertial frame. Therefore, the sign of $\omega_{\mathcal{L}}$ requires special attention.

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

To derive the derivative with respect 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.5 Jacobian of quaternion rotation with respect to the quaternion

The derivative of the rotation with respect to the quaternion is a bit tricky, but can be derived straightforwardly with the usage of (A.6) and (A.7). In this section, I will use a lighter notation for the quaternion: $\mathbf{q} = \begin{bmatrix} w \\ \mathbf{v} \end{bmatrix}$. The rotated vector can be expressed with this notation as:

$$\mathbf{a}' = \mathbf{q} \otimes \mathbf{a} \otimes \mathbf{q}^* = \mathbf{Ra} = w^2\mathbf{a} - \mathbf{v}^T\mathbf{va} + 2\mathbf{vv}^T\mathbf{a} - 2w \begin{bmatrix} \mathbf{a} \end{bmatrix}_\times \mathbf{v} \tag{A.12}$$

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

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

Using these results the Jacobian with respect to quaternion is:

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

APPENDIX B

ESKF-RELATED EQUATIONS

Most of the following formulas with little difference can be found in [27].

B.1 True rotation matrix

Firstly, I want to introduce the Ordinary Differential Equation (ODE) of rotation matrices. The rotation matrices have orthogonal properties, therefore their inverse is equal to their transpose, which results in:

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

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 an $\boldsymbol{\omega}$ vector exists for which the following holds:

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

Here, the second row represents the ODE of rotation matrices, which establish 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 (meaning near to a reference rotation), the above equation simplifies to $\dot{\mathbf{R}} = \left[\boldsymbol{\omega}\right]_\times$. Here, $\boldsymbol{\omega}$ can be interpreted as the vector representing instantaneous angular velocities. This understanding sheds light on the Lie algebra $\mathfrak{so}(3)$, which can be seen as the space of derivatives of $r(t)$ at the origin. It also serves as the tangent space to $\mathrm{SO}(3)$ group.

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 the Taylor series:

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

Neglecting the second and higher order members leads to (3.4).

B.2 Error state equations

The error state can be expressed 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:

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

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:

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

In (B.9) the definition of thequaternion 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 calculation the quaternion rotation matrices results in:

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

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:

$$\dot{\delta\boldsymbol{\theta}} = -\Big[\boldsymbol{\omega}_m-\boldsymbol{\beta}_\omega\Big]_\times\delta\boldsymbol{\theta}+\delta\boldsymbol{\beta}_\omega+\boldsymbol{\eta}_\omega \tag{B.12}$$

B.2.3 Velocity error

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

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

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

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

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

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

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) with respect to the error rotation vector $\delta\boldsymbol{\theta}$. Firstly, I aim to formalize that equation with rotation vector parame-

ters, neglecting the constant terms:

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

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:

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

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. Only considering the vector part, the Jacobian is:

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

ABBREVIATIONS

AWGN	Additive White Gaussian Noise
BDS	BeiDou Navigation Satellite System
BME	Budapesti Műszaki Egyetem
CARLA	Car Learning to Act
COSPL	Computational Optical Sensing and Processing Laboratory
DLT	Direct Linear Transformation
DOA	Direction-of-Arrival
ECEF	Earth-Centered Earth-Fixed
EKF	Extended Kalman Filter
ESKF	Error-State Kalman Filter
FOA	Frequency-of-Arrival
FOV	Field of View
GLONASS	Global Navigation Satellite System
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
KLT	Kanade-Lucas-Tomasi
LEO	Low Earth Orbit
LOS	Line of Sight
LOST	Linear Optimal Sine Triangulation
LS	Least Squares
LU	Lower-Upper
MLE	Maximum Likelihood Estimation
NED	North-East-Down
ODE	Ordinary Differential Equation

RMSE	Root Mean Square Error
SCL	System Control Laboratory
SfM	Structure from Motion
SLAM	Simultaneous Localization and Mapping
SOP	Signal of Opportunity
SO	Special Orthogonal
SZTAKI	Számítástechnikai és Automatizálási Kutatóintézet
TOA	Time-of-Arrival
UAS	Unmanned Aerial System
UAV	Unmanned Aerial Vehicle
VIO	Visual-Inertial Odometry
WGS	World Geodetic System
3-D	Three-dimensional

List of Figures

LIST OF TABLES

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] Bernhard Hofmann-Wellenhof, Herbert Lichtenegger, and Elmar Wasle. *GNSS–Global Navigation Satellite Systems: GPS, GLONASS, Galileo, and more*. Springer Science & Business Media, 2012.

[9] S. Tan. *GNSS Systems and Engineering: The Chinese Beidou Navigation and Position Location Satellite*. Wiley, 2018.

[10] Giorgio Grisetti, Rainer Kümmerle, Cyrill Stachniss, and Wolfram Burgard. A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010.

[11] Alexander A. Nguyen and Zaher M. Kassas. Efficient transmitter selection strategies for improved information gathering of aerial vehicle navigation in gnss-denied environments. *IEEE Transactions on Aerospace and Electronic Systems*, 2023. DOI. No. 10.1109/MAES.2023.3266179.

[12] Mohammad Neinavaie, Joe Khalife, and Zaher M. Kassas. Cognitive detection of unknown beacons of terrestrial signals of opportunity for localization. *IEEE Transactions on Wireless Communications*, August 2023.

[13] Mohammad Neinavaie and Zaher M. Kassas. Unveiling starlink leo satellite ofdm-like signal structure enabling precise positioning. *IEEE Transactions on Aerospace and Electronic Systems*, 2023.

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

[15] 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.

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

[17] 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.

[18] 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.

[19] 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.

[20] 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.

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

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

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

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

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

[26] 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.

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

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

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

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

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

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

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

[34] 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.

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

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

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

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

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

[40] Jon Dattorro. Convex optimization & euclidean distance geometry. 2004.

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

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

[43] 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.

[44] 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.

[45] Jianbo Shi and Tomasi. Good features to track. In *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[46] Peter Bauer, Akos Zarandy, Antal Hiba, and Jozsef Bokor. Camera-based in-time detection of intruder aircraft. In *CNNA 2018; The 16th International Workshop on Cellular Nanoscale Networks and their Applications*, pages 1–4, 2018.

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

[48] 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.

[49] Shah Zahid Khan, Mujahid Mohsin, and Waseem Iqbal. On gps spoofing of aerial platforms: a review of threats, challenges, methodologies, and future research directions. *PeerJ. Computer science*, 7:e507, 2021.

[50] Yinrong Qiao, Yuxing Zhang, and Xiao Du. A vision-based gps-spoofing detection method for small uavs. In *2017 13th International Conference on Computational Intelligence and Security (CIS)*, pages 312–316, 2017.