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

Budapest University of Technology and Economics

Faculty of Electrical Engineering and Informatics

Department of Control Engineering and Information Technology

Institute for Computer Science and Control

Systems and Control Laboratory

Visual-inertial navigation with intermittent or spoofed GPS information

MASTER'S THESIS

Author

Zsombor Novozánszki

Advisor

Dr. Emese Szádeczky-Kardoss Gincsainé

Dr. Péter Bauer

August 29, 2024

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, 2024. augusztus 29.

Novozánszki Zsombor
hallgató

KIVONAT

Dolgozatomban bemutatom a Számítástechnikai és Automatizálási Kutatóintézetben végzett munkámat, amelynek témája vizuális-inerciális navigáció. A tanulmány során megvalósított rendszer célja meghatározni egy légijármű pozícióját Inertial Measurement Unit (IMU) mérések és mono kameraképek segítségével.

A javasolt rendszer az IMU és a kamera adatok integrációját foglalja magába egy Kálmán-szűrő alapú keretrendszerbe. Az IMU mérésekből a repülőgép pozícióját, sebességét és orientációját lehet becsülni, amelynek hibáját a kamera képekből nyert információval koorigálom. A rendszernek képesnek kell lennie arra is, hogy kezelje az időszakosan vehető GPS-jeleket.

A dolgozat bemutatja az algoritmus fejlesztését, amely fúziónálja az inerciális és vizuális adatokat, valamint az ehhez szükséges elméleti alapismereteket és módszertanokat. Az alap algoritmusokat szakirodalomból vettem, amelyeket úgy módosítottam, hogy a saját rendszerbe optimálisan integrálható legyen.

Összefoglalva, a dolgozat részletesen bemutatja az elkészített algoritmusokat, és a hozzájuk kapcsolodó matematikát. A kutatás során egy szimulációt hoztam létre, amelyet fokozatosan közelítettem a valóságos körülményeket leginkább modellező rendszerhez. Főcélom az elkészített navigációs rendszer megvalósítása és tesztelése szintetikus vagy valós felvételeken.

ABSTRACT

In this work, I will present my work at the Institute for Computer Science and Control. The topic of my thesis is the use of visual-inertial navigation to determine the position of an aircraft based on (IMU) measurements and images from a monocular camera. The objective of this study is to develop a robust and accurate system that can provide reliable estimates of the aircraft's position.

The suggested system implicates the integration of the IMU and camera data using an Error-State Kalman filter. The IMU measurements are used to estimate the aircraft's position, velocity and orientation, while the camera images are utilized to estimate the aircraft's position relative to the features on the ground. The system should also be capable of handling intermittent GPS signals.

This paper presents the development of the algorithm that fuses the inertial and visual data, and the necessary theoretical knowledge and required methods for its implementation. I have chosen the basic algorithm from the literature however, the implementation involves several successive steps.

In summary, the thesis introduces a detailed description of the algorithms and the related mathematics. For the first semester of my research, I focused on creating a simulation, which I gradually approximated to a system that best models real-world conditions. The main goal is to apply the algorithms to one of the experimental aircraft of SZTAKI.

Chapter 1

Introduction

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

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

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

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

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

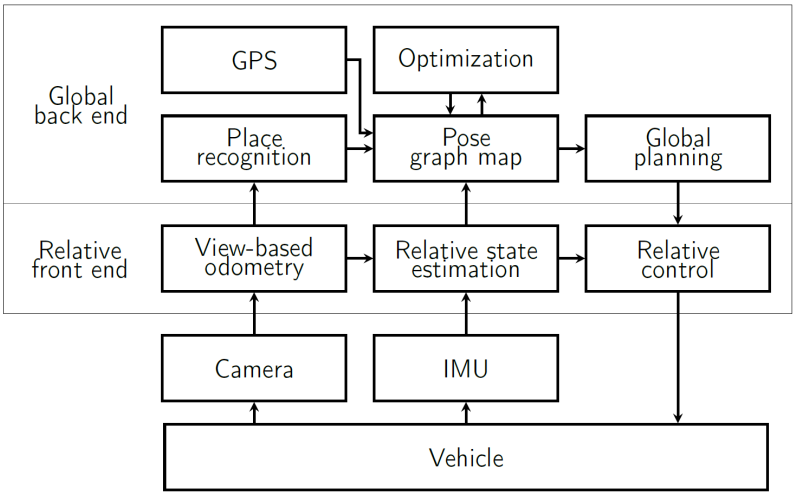


Figure 1.1: Block diagram of relative navigation ([7]: page 2, Figure 2)

duce 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 computational advantage of an Error-State Kalman Filter (ESKF) to be utilized, which is a better solution, than an ordinary EKF because it calculates the nominal state according to the original non-linear dynamics, therefore the linearization around this state is a better approximation.

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

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

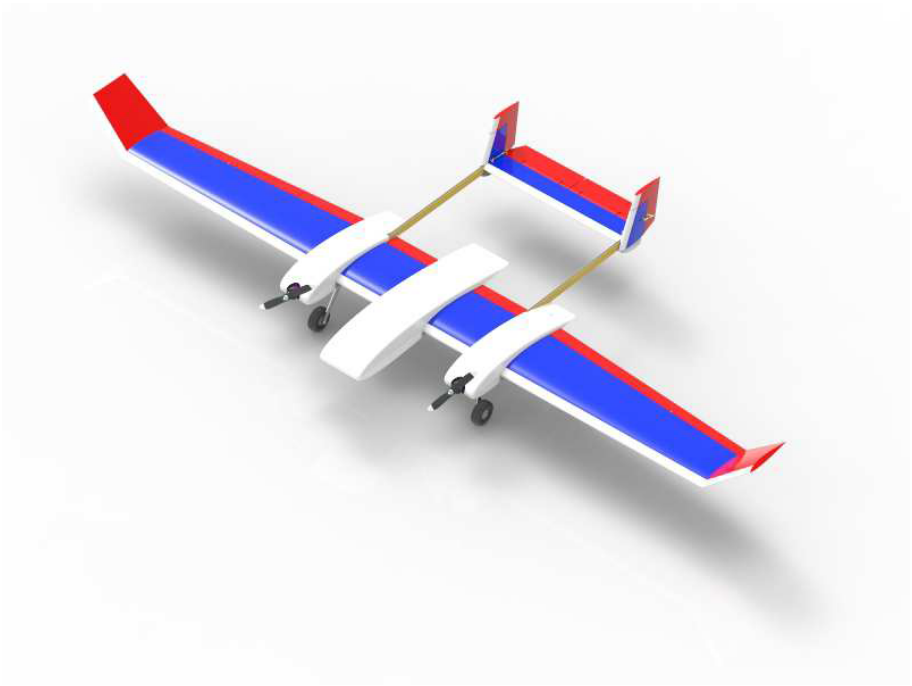


Figure 1.2: Realistic drawing of Sindy ([15]: title page)

1.1 Structure of the paper

In the first semester of this project, the primary objective was to integrate a visual-inertial ESKF into a simulation that was already available to me. The simulation consisted of a grid of feature points, and the goal was to use an ESKF-based relative navigation front-end 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, and pixelization, but further imperfections needed to be integrated related to the lens of the camera and the biases of the sensors.

The paper is structured as follows: Chapter 2 provides an introduction to several fundamental mathematical concepts that are crucial for comprehending our approach. In Chapter 3, the mathematical foundation of the filter is explained. Chapter 4 outlines the filter's parameters and establishes its procedural steps. Chapter 5 presents the estimation results in Matlab/Simulink environment. Finally, Chapter 6 offers a summary of the work conducted in the first semester, and highlights certain future development steps.

CHAPTER 2

MATHEMATICAL FOUNDATIONS

Before I would 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 my approach, filter-related mathematics uses four frames for the mathematical notation: Earth (E), node (N)-, body (B), and camera (C) frame. These are shown in Figure 2.1.

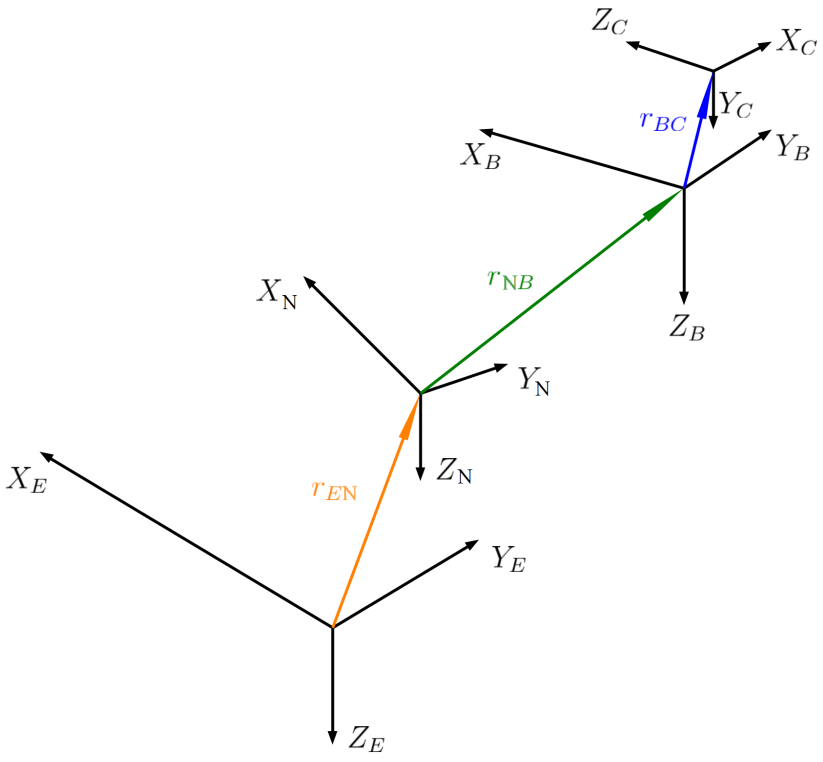


Figure 2.1: Applied coordinate systems

The Earth frame means an Earth fixed frame, and the Nort-East-Down (NED) coordinate system is chosen for this purpose in such UAV applications where the vehicle covers only a few kms. The node frame is a locally declared fixed frame, which usage is necessary for relative navigation methods since the filter produces estimates in this frame. The body- and camera frames are detailed in the following sections.

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:

- The motion of the aircraft is most easily described in a body-fixed frame, however, Newton's equations of motion are derived relative to a fixed, 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 gyroscope measure concerning the body frame. This is essential in the case of strap-down IMUs since they provide measurements with respect to the body frame.
- Finally, the mission requirements of the aircraft, e.g. flight path require a global frame too.

2.1.1 North-East-Down reference frame

The NED coordinate system has emerged as a standard in UAV applications over limited distances, typically spanning a few kilometers. Notably, this reference frame is conventionally anchored to a stationary point on Earth, affording its use as an inertial reference frame in analytical computations. Consistent with its name, the NED system aligns its X-axis with the geographic north, its Y-axis

with the east, and its Z-axis points inwards to the Earth. Its X-Y plane is the local tangent plane of the Earth's ellipsoid.

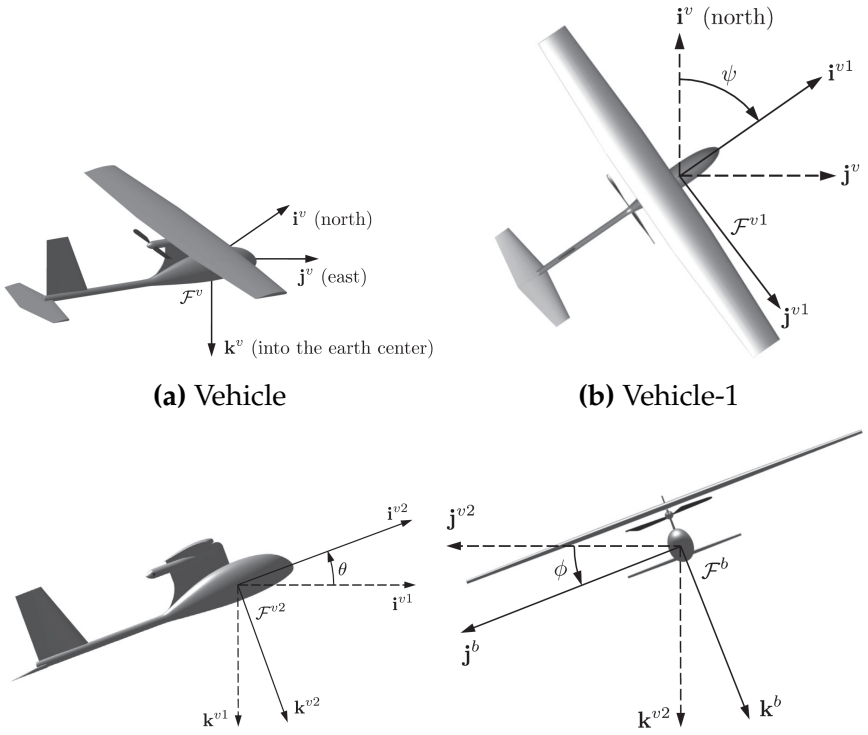
The front end of relative navigation methods adopts the node coordinate system as an inertial frame, a crucial precondition for the functioning of the filter. The node frame remains locally fixed and undergoes redeclaration following each measurement. The back end of the system facilitates the derivation of global estimates, such as NED coordinates, utilizing the outcomes yielded by the filter.

2.1.2 Body frame

The body coordinate system is the most important besides the inertial frame. It appears directly in the kinematic equations, therefore it is crucial to understand properly how it is defined and what is the relation to the inertial frame. The origin is the center of the mass of the aircraft, and at this point, I would like to highlight the fact in almost every application the IMU is placed here. The axes of the body frame are defined as follows: the X-axis points out the nose of the aircraft, the Y-axis points out the right wing and the Z-axis points downwards.

Once the coordinate system has been defined, it is necessary mentioning the relation to the inertial frame. The most commonly applied approach to obtain the orientation of the aircraft in the NED system is to express it with Euler angles which means three rotations one after the other. Defining the Euler angles I use the same terminology as in [10], their approach is to introduce three additional coordinate systems: vehicle, vehicle-1, and vehicle-2. They are shown in Figure 2.2a-2.2d.

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



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

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

Figure 2.2: Frames to define Euler angles ([10]: pages 13-15, Figures 2.4-2.7)

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

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

Finally, I would like to mention the usage of homogeneous transformation is widespread regarding vision-based applications, thanks to the fact it allows the calculation of three-dimensional (3D) coordinates from one frame to another with a single matrix multiplication. With known translation $\mathbf{r}_{BE}$ and rotation ($\mathbf{R}_{BE} \equiv \mathbf{R}_{BV}$) between the Earth frame (NED) and body frame, the homogeneous transformation can be expressed as:

$$\mathbf{H}_{BE} = \begin{bmatrix} \mathbf{R}_{BE} & \mathbf{r}_{BE} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{2.2}$$

The transformation can be applied as:

$$\mathbf{H}_{BE} = \begin{bmatrix} \mathbf{R}_{BE} & \mathbf{r}_{BE} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{v}_b \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{BE}\mathbf{v}_b + \mathbf{r}_{BE} \\ 1 \end{bmatrix} \tag{2.3}$$

2.2 Camera frames

It is important to say a few words about camera-related frames and transformation in view of the fact that, camera pictures are used to improve navigation. I would like to clarify that, we distinguish between a camera- and an image coordinate system. This means that a point with known coordinates in the Earth frame can be projected onto the image plane with two operations: a transformation is needed from the Earth frame to the camera system and after that, a projection is needed.

2.2.1 Camera system

It is very important to determine sufficiently accurate the transformation from body-to-camera system. Generally, the camera is not placed in the body center, therefore the transformation first requires a translation with $\mathbf{r}_{BC}$. Since navigation happens with respect to the surroundings an at least partially downward-facing camera is mandatory, which means a rotation is needed 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 2.3 shows why the swapping is needed: almost every application defines the axes of the camera system as follows: the X-axis points to the right, Y-axis points to the bottom and Z-axis points out of the principal point of the image.

It means that the basis vectors of the rotated body frame should be transformed as outlined below: $\vec{i}'_b = \vec{k}_c$, $\vec{j}'_b = \vec{i}_c$ and $\vec{k}'_b = \vec{j}_c$, the transformation from body-to-

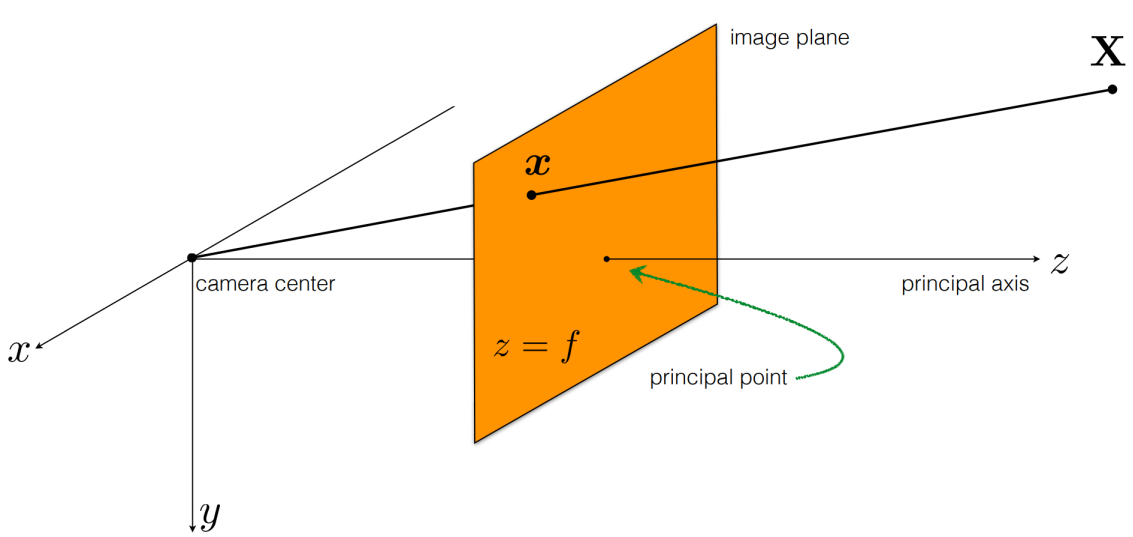


Figure 2.3: Camera and image system ([16]: page 7)

camera frame:

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

where **S** stands for the axes swapping transformation, and β is the angle between the camera and the body. Considering the translation the homogeneous transformation can be written in the form of:

$$\mathbf{H}_{CB} = \begin{bmatrix} \mathbf{T}_{CB} & \mathbf{r}_{CB} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{2.5}$$

It is worth noting that this approach results in a top left corner origin of the image plane. However, there are other applications where the origin is located at the bottom left corner.

2.2.2 Pinhole camera projection

Here and now, every known point in the Earth frame can be transformed into the camera system with the usage of (2.2) and (2.5). The next task is to determine the pixel coordinates of the point in question, and this requires a projection $h(.)$. For the simulation, the most simple model was chosen, the pinhole camera projection model. The pinhole model framework describes the aperture of the camera as a point, therefore the lens distortion errors are neglected. As it can be seen in Figure 2.3, the model only requires two parameters:

1. The focal length (f) which shows the distance between the camera center and the image plane.
2. The second parameter is the principal point which is typically the center of the image.

With the mentioned parameters, the transformation includes the normalization of points with their distance from the camera center and the projection onto the image plane. In the pinhole framework, the transformation of the normalized points can be specified with a camera matrix, which can be derived from pinhole parameters:

$$\mathbf{C} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

Using (2.6) the whole projection is defined as:

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

As a final point, I would like to emphasize that the (2.7) transformation can determine the image coordinates of every three-dimensional point, but a real camera has limitations which can be described either with the image size (W, H) or the FOV angles.

2.3 Quaternions

Before embarking into the ESKF, it is worth mentioning a few words about quaternions. Originally formulated in [17] 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 [18]. 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 ($\mathbf{z} = e^{i\theta}$) are able to 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 3D 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 3D vector $\mathbf{v}$ by the following operation:

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

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

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

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

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

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

CHAPTER 3

MATHEMATICAL ANALYSIS 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 [18], but with three major differences:

1. I didn't insert $\mathbf{g}$ in the state vectors $\mathbf{x}, \delta\mathbf{x}$ to estimate because our final implementation will 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. This impacts the formulation of the mathematical equations, but in most cases, the equations have symmetrical properties.

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

3.1 Error-state kinematics

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

One of the tools which can be used for this purpose is the ESKF, which has multiple advantages if applied for nonlinear systems:

- The error state always operates close to the actual system state, thus far from possible parameter singularities, and gimbal lock issues.
- All second-order products are negligible because the error-state always remains small. This makes the computation of Jacobians very easy and fast.
- The dynamics of the error-state are slow, due to all large-signal dynamics being integrated into the nominal-state. This results in a highly beneficial property: the KF corrections can be applied lower rate, than the predictions.

3.1.1 State of KFs for IMU driven systems

The objective is estimating the position of the aircraft, which requires to use the kinematic equations of the aircraft. The kinematics express the relationships among position, orientation, velocity, acceleration, and angular velocity. Whereas acceleration and angular velocity are utilized as IMU measurements, the state includes position, velocity, and orientation. Moreover, it is supplemented by estimating the biases of the sensors.

The position of the body is expressed in a fixed frame, which is the node frame in relative navigation projects. The orientation of the body frame is described with a quaternion compared to the node frame too. On the contrary, the velocity and

the biases are body-frame fixed. This results in the state vector:

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

3.1.2 ESKF states

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

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

Table 3.1: All variables in ESKF

The states are divided in such a way that the large-signal dynamics are assigned to the nominal state, while the small-signal dynamics are assigned to the error state. A few comments on the table:

- The relation between the error- quaternion and rotation vector was presented, using the formula defined in Appendix A.1, then approximating cosine and sine as above, because $\delta\boldsymbol{\theta}$ is always near 0.

- The body referenced measurements are $\mathbf{a}_m$ and $\boldsymbol{\omega}_m$. Their values are captured as noisy IMU measurements, therefore:

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

  From above the true values can be obtained, and substituting nominal- and error- values into true variables:

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

- The biases small-signal dynamics $\delta\boldsymbol{\beta}_a$, $\delta\boldsymbol{\beta}_\omega$ are represented with Gaussian white noises $\boldsymbol{\eta}_{\beta_a}$, $\boldsymbol{\eta}_{\beta_\omega}$, just like the measurement noises $\boldsymbol{\eta}_a$, $\boldsymbol{\eta}_\omega$.

- The true rotation matrix can be approximated by composing the nominal rotation matrix with power series because $\delta\mathbf{R}$ can be written in Taylor-series. The (3.4) approximation is detailed in B.1, but the formula is provided here, therefore neglecting the second- and higher-order terms, it yields:

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

True-state

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

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

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

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

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

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

The (3.5b) is the time derivative formula of quaternion, and detailed in Appendix A.2. The (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 $\Big[.\Big]_\times$.

Nominal-state

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

$$\dot{\mathbf{p}}_n = \mathbf{R}^T \mathbf{v}_b \tag{3.7a}$$

$$\dot{\mathbf{q}}_n = \frac{1}{2} \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} \otimes \mathbf{q}_n \tag{3.7b}$$

$$\dot{\mathbf{v}}_b = \mathbf{a} + \mathbf{R}\mathbf{g} + \Big[\mathbf{v}_b\Big]_\times \boldsymbol{\omega} \tag{3.7c}$$

$$\dot{\boldsymbol{\beta}}_{a,t} = 0 \tag{3.7d}$$

$$\dot{\boldsymbol{\beta}}_{\omega,t} = 0 \tag{3.7e}$$

These equations reflect the dynamics of the system with slowly varying biases, which is beneficial because allows the integration of large-signal dynamics into the nominal-state.

Error-state

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

$$\delta\dot{\mathbf{p}}_n = \mathbf{R}^T\delta\mathbf{v}_b + \mathbf{R}^T\left[\mathbf{v}_b\right]_\times \delta\boldsymbol{\theta} \tag{3.8a}$$

$$\delta\dot{\boldsymbol{\theta}} = \left[\boldsymbol{\omega}_m - \boldsymbol{\beta}_\omega\right]_\times \delta\boldsymbol{\theta} - \delta\boldsymbol{\beta}_\omega - \boldsymbol{\eta}_\omega \tag{3.8b}$$

$$\begin{aligned}\delta\dot{\mathbf{v}}_b = &-\left[\mathbf{R}\mathbf{g}\right]_\times \delta\boldsymbol{\theta} - \left[\boldsymbol{\omega}_m - \boldsymbol{\beta}_\omega\right]_\times \delta\mathbf{v}_b - \delta\boldsymbol{\beta}_a - \left[\mathbf{v}_b\right]_\times \delta\boldsymbol{\beta}_\omega \\ &- \boldsymbol{\eta}_a - \left[\mathbf{v}_b\right]_\times \boldsymbol{\eta}_\omega\end{aligned} \tag{3.8c}$$

$$\delta\dot{\boldsymbol{\beta}}_a = \boldsymbol{\eta}_{\beta_a} \tag{3.8d}$$

$$\delta\dot{\boldsymbol{\beta}}_\omega = \boldsymbol{\eta}_{\beta_\omega} \tag{3.8e}$$

The calculations are detailed in Appendix B.2.

3.1.3 Error-state kinematics in discrete time

The previous equations are defined in continuous-time, but computer implementations use a discrete-time model. To incorporate discrete time intervals $\Delta t > 0$, the differential equations mentioned earlier 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 [18]: the deterministic part integrated according to the ZOH method, and the stochastic part modeled as random impulses applied to the velocity, orientation, and bias estimates. This results in the

nominal-state equations:

$$\mathbf{p}_{n,k+1} = \mathbf{p}_{n,k} + \mathbf{R}^T \mathbf{v}_{b,k} \Delta t + \frac{1}{2} \mathbf{R}^T \left(\mathbf{a}_k + \mathbf{R}\mathbf{g} + \left[\mathbf{v}_{b,k} \right]_\times \boldsymbol{\omega}_k \right) \Delta t^2 \tag{3.9a}$$

$$\mathbf{q}_{n,k+1} = \mathbf{q}\{-(\boldsymbol{\omega}_m - \boldsymbol{\beta}_\omega)\Delta t\} \otimes \mathbf{q}_{n,k} \tag{3.9b}$$

$$\mathbf{v}_{b,k+1} = \mathbf{v}_{b,k} + \left(\mathbf{a}_k + \mathbf{R}\mathbf{g} + \left[\mathbf{v}_{b,k} \right]_\times \boldsymbol{\omega}_k \right) \Delta t \tag{3.9c}$$

$$\boldsymbol{\beta}_{a,k+1} = \boldsymbol{\beta}_{a,k} \tag{3.9d}$$

$$\boldsymbol{\beta}_{\omega,k+1} = \boldsymbol{\beta}_{\omega,k} \tag{3.9e}$$

The position is integrated both from velocity and acceleration, and the rotation from angular velocity. The last one is given in a closed form assuming that the angular velocity is constant during the sampling interval. Using the same integration approach complemented by the integration of the stochastic part, the error-state equations result as:

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

$$\delta\boldsymbol{\theta}_{k+1} = \mathbf{R}\{-(\boldsymbol{\omega}_{m,k} - \boldsymbol{\beta}_{\omega,k})\Delta t\}\delta\boldsymbol{\theta}_k - \delta\boldsymbol{\beta}_{\omega,k}\Delta t + \boldsymbol{\theta}_{i,k} \tag{3.10b}$$

$$\begin{aligned} \delta\mathbf{v}_{b,k+1} = \delta\mathbf{v}_{b,k} + \Big(&- \left[\mathbf{R}\mathbf{g} \right]_\times \delta\boldsymbol{\theta}_k - \left[\boldsymbol{\omega}_{m,k} - \boldsymbol{\beta}_{\omega,k} \right]_\times \delta\mathbf{v}_{b,k} \\ &- \delta\boldsymbol{\beta}_{a,k} - \left[\mathbf{v}_{b,k} \right]_\times \delta\boldsymbol{\beta}_{\omega,k} \Big) \Delta t + \left[\mathbf{v}_{b,k} \right]_\times \boldsymbol{\theta}_{i,k} + \mathbf{v}_{i,k} \end{aligned} \tag{3.10c}$$

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

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

$\boldsymbol{\theta}_i, \mathbf{v}_{b,i}, \mathbf{a}_i$ and $\boldsymbol{\omega}_i$ are the random impulses which form the stochastic part of the equation. Hence, they modeled as Gaussian white noises, the negative sign freely can be changed to positive. Their covariance matrices are integrated as (see [18]

Appendix E for details):

$$\begin{aligned}\mathbf{\Theta}_i &= \sigma_{\eta_\omega}^2 \Delta t^2 \mathbf{I} \\ \mathbf{V}_i &= \sigma_{\eta_a}^2 \Delta t^2 \mathbf{I} \\ \mathbf{A}_i &= \sigma_{\beta_a}^2 \Delta t \mathbf{I} \\ \mathbf{\Omega}_i &= \sigma_{\beta_\omega}^2 \Delta t \mathbf{I}\end{aligned} \tag{3.11}$$

3.2 Measurement equations

The measurement equations in visual-inertial ESKF are based on the fundamental concept that the system can acquire pixel coordinates of feature points as measurements, therefore the filter has to create predicted pixel coordinates to form the residual.

The more sophisticated approaches do not assume known distances of the observed features but optimize the 3D coordinates of the observed features based on the vision tracker information. At this point, it is a further possible development, because currently, the algorithm assumes the knowledge of the exact coordinates. The essence of both methods is that, after transforming the feature points into the camera frame based on (2.2) and (2.5), they can be projected onto the image plane with (2.7). This can be formalized as:

$$h(\mathbf{T}_{CB}\mathbf{R}_{BE}(\mathbf{p}_n^f - \mathbf{p}_n^b)) = h(\mathbf{p}_c^f) = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{3.12}$$

The transformation above is assuming that the camera system and body frame have a common origin, and this simplification was applied at the beginning of the implementation.

CHAPTER 4

ESKF FRAMEWORK

The following chapter summarizes the parameters of the ESKF framework and clarifies the steps of the filter.

4.1 ESKF parameters

Every KF needs a state vector, but in the case of ESKF, there are two: one for the nominal- $\mathbf{x}_k$ and the other for the error-state $\delta\mathbf{x}_k$ vector. The system is governed by the input $\mathbf{u}_k$ and the noise perturbations $\mathbf{i}$:

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{p}_{n,k} \\ \mathbf{q}_{n,k} \\ \mathbf{v}_{b,k} \\ \boldsymbol{\beta}_{a,k} \\ \boldsymbol{\beta}_{\omega,k} \end{bmatrix}, \qquad \delta\mathbf{x}_k = \begin{bmatrix} \delta\mathbf{p}_{n,k} \\ \delta\boldsymbol{\theta}_k \\ \delta\mathbf{v}_{b,k} \\ \delta\boldsymbol{\beta}_{a,k} \\ \delta\boldsymbol{\beta}_{\omega,k} \end{bmatrix}, \qquad \mathbf{u}_k = \begin{bmatrix} \mathbf{a}_{m,k} \\ \boldsymbol{\omega}_{m,k} \end{bmatrix}, \qquad \mathbf{i} = \begin{bmatrix} \boldsymbol{\theta}_i \\ \mathbf{v}_i \\ \mathbf{a}_i \\ \boldsymbol{\omega}_i \end{bmatrix} \tag{4.1}$$

4.2 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, but the error-state has to be linearized. The transition- $\mathbf{F}_x$ and noise matrix $\mathbf{F}_i$ can be

derived from (3.10):

$$\mathbf{F}_x = \frac{\partial f}{\partial \delta \mathbf{x}_k} = \begin{bmatrix} \mathbf{I} & \mathbf{R}^T \left[\mathbf{v}_{b,k}\right]_\times \Delta t & \mathbf{R}^T \Delta t & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}\{-(\boldsymbol{\omega}_{m,k} - \boldsymbol{\beta}_{\omega,k})\Delta t\} & \mathbf{0} & \mathbf{0} & -\mathbf{I}\Delta t \\ \mathbf{0} & -\left[\mathbf{R}\mathbf{g}\right]_\times \Delta t & \mathbf{I} - \left[\boldsymbol{\omega}_{m,k} - \boldsymbol{\beta}_{\omega,k}\right]_\times \Delta t & -\mathbf{I}\Delta t & -\left[\mathbf{v}_{b,k}\right]_\times \Delta t \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{4.2}$$

$$\mathbf{F}_i = \frac{\partial f}{\partial \mathbf{i}} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \left[\mathbf{v}_{b,k}\right]_\times & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}, \tag{4.3}$$

where $\mathbf{I}, \mathbf{0} \in \mathbb{R}^{3\times 3}$. Now the error-state is:

$$\delta \mathbf{x}_{k+1} = \mathbf{F}_x \delta \mathbf{x}_k + \mathbf{F}_i \mathbf{i} \tag{4.4}$$

Using formulas from above and denote equations in (3.9) with $f(.)$ the prediction step involves:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) \tag{4.5a}$$

$$\delta \hat{\mathbf{x}}_{k+1} = \mathbf{F}_x \delta \hat{\mathbf{x}}_k \tag{4.5b}$$

$$\mathbf{P}_{k+1} = \mathbf{F}_x \mathbf{P}_k \mathbf{F}_x^T + \mathbf{F}_i \mathbf{Q}_i \mathbf{F}_i^T \tag{4.5c}$$

Here, $\delta \hat{\mathbf{x}}$ is the mean of the error-state, therefore $\delta \mathbf{x} \sim \mathcal{N}(\delta \hat{\mathbf{x}}, \mathbf{P})$.

4.3 Update step

The update steps usually occur rarely compared to prediction steps, and in the current solution, the update is performed based on one picture with multiple feature points on it. The goal is to linearize the measurement equations and derive the observation matrix with respect to the error-state. The most straightforward approach to do that for a feature point involves the utilization of the chain rule:

$$\mathbf{H}_x = \frac{\partial h(\mathbf{p}_c^f)}{\partial \delta \mathbf{x}} = \frac{\partial h(\mathbf{p}_c^f)}{\partial \mathbf{p}_c^f} \frac{\partial \mathbf{p}_c^f}{\partial \mathbf{x}_t} \frac{\partial \mathbf{x}_t}{\partial \delta \mathbf{x}} = \mathbf{J}\mathbf{P}_{x_t}\mathbf{X}_{\delta x}, \tag{4.6}$$

where h is the nonlinear transformation that projects the $\mathbf{p}_c^f$ 3D point onto the image plane. The subscript denotes that, the coordinates of the point interpreted in the camera frame and the superscript conveys that, it describes the feature point.

Assuming measurements with Gaussian white noise that distribution is $\mathcal{N}(0, \mathbf{V})$, the update step as follows:

$$\mathbf{K} = \mathbf{P}\mathbf{H}_x^T(\mathbf{H}_x\mathbf{P}\mathbf{H}_x^T + \mathbf{V})^{-1} \tag{4.7a}$$

$$\delta\hat{\mathbf{x}}_{k+1} = \mathbf{K}(\mathbf{z} - h(\hat{\mathbf{x}}_{k,t})) \tag{4.7b}$$

$$\mathbf{P}_{k+1} = (\mathbf{I} - \mathbf{K}\mathbf{H}_x)\mathbf{P}_k(\mathbf{I} - \mathbf{K}\mathbf{H}_x)^T + \mathbf{K}\mathbf{V}\mathbf{K}^T \tag{4.7c}$$

4.3.1 Linearize projection with respect to feature point

The Jacobian of the projection is calculated by linearizing (2.7), whereas the transformation results in a 3-element column vector, and the point has 3 coordinates the Jacobian should be a 3×3 matrix, but the last column leads to only zeros which are not very useful, therefore it is neglected:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial h_1(\mathbf{p}_c^f)}{\partial x} & \frac{\partial h_1(\mathbf{p}_c^f)}{\partial y} & \frac{\partial h_1(\mathbf{p}_c^f)}{\partial z} \\ \frac{\partial h_2(\mathbf{p}_c^f)}{\partial x} & \frac{\partial h_2(\mathbf{p}_c^f)}{\partial y} & \frac{\partial h_2(\mathbf{p}_c^f)}{\partial z} \end{bmatrix} = \begin{bmatrix} \frac{f}{z} & 0 & -\frac{fx}{z^2} \\ 0 & \frac{f}{z} & -\frac{fy}{z^2} \end{bmatrix} \Bigg|_{\mathbf{p}_c^f} \tag{4.8}$$

4.3.2 Linearize feature point with respect to the true-state

The next step is to determine the derivative of $\mathbf{p}_c^f$ by the true-state. Since $\mathbf{p}^f$ is known in the node frame it has to be transformed into the camera frame as follows:

$$\frac{\partial \mathbf{p}_c^f}{\partial \mathbf{x}_t} = \frac{\partial \mathbf{T}_{CB}\mathbf{R}\{\mathbf{q}_n^b\}(\mathbf{p}_n^f - \mathbf{p}_n^b)}{\partial \mathbf{x}_t}, \tag{4.9}$$

Here, $\mathbf{p}_n^b$ and $\mathbf{q}_n^b$ are elements from the state-vector, just they got superscripts to clearly mark they describe the position and orientation of the body frame. It can be seen that there is no other state parameter in the expression, therefore further derivatives result in $\mathbf{0}_{3\times 3}$ matrices. The derivative by $\mathbf{p}_n^b$ can be determined easily:

$$\frac{\partial \mathbf{p}_c^f}{\partial \mathbf{p}_n^b} = -\mathbf{T}_{CB}\mathbf{R}\{\mathbf{q}_n^b\} \tag{4.10}$$

Calculating the derivative by the quaternion is tricky. Initially, the quaternion rotation formula needs to be employed on (4.9), then the derivative should be decomposed using the chain rule into separate components: one with respect to the vector and the other with respect to the quaternion:

$$\begin{aligned}\frac{\partial \mathbf{p}_c^f}{\partial \mathbf{q}_n} &= \left.\frac{\partial \mathbf{q}_{CB} \otimes \mathbf{q}_n \otimes (\mathbf{p}_n^f - \mathbf{p}_n^b) \otimes \mathbf{q}_n^* \otimes \mathbf{q}_{CB}^*}{\partial \mathbf{q}_n}\right|_{\mathbf{a}=\mathbf{p}_n^f-\mathbf{p}_n^b} \\ &= \frac{\partial \mathbf{q}_{CB} \otimes (\mathbf{q}_n \otimes \mathbf{a} \otimes \mathbf{q}_n^*) \otimes \mathbf{q}_{CB}^*}{\partial \mathbf{q}_n \otimes \mathbf{a} \otimes \mathbf{q}_n^*} \frac{\partial \mathbf{q}_n \otimes \mathbf{a} \otimes \mathbf{q}_n^*}{\partial \mathbf{q}_n}\end{aligned} \tag{4.11}$$

Using results from Appendices A.3 and A.4, the outcome is:

$$\frac{\partial \mathbf{p}_c^f}{\partial \mathbf{q}_n} = 2\mathbf{T}_{CB}\left[q_w\mathbf{a} + \mathbf{q}_v \times \mathbf{a} \mid \mathbf{q}_v^T\mathbf{a}\mathbf{I}_{3\times 3} + \mathbf{q}_v\mathbf{a}^T - \mathbf{a}\mathbf{q}_v^T - q_w\left[\mathbf{a}\right]_\times\right] \tag{4.12}$$

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

$$\mathbf{P}_{\mathbf{x}_t} = \begin{bmatrix}\frac{\partial \mathbf{p}_c^f}{\partial \mathbf{p}_n^b} & \frac{\partial \mathbf{p}_c^f}{\partial \mathbf{q}_n} & \mathbf{0}_{3\times 9}\end{bmatrix} \tag{4.13}$$

4.3.3 Linearize the true-state with respect to the error-state

Finally, $\partial \mathbf{x}_t / \partial \delta \mathbf{x}$ should be determined. All derivatives yield the identity block $\mathbf{I}_3$, except for the quaternion. The Jacobian of the true quaternion with respect to the error rotation vector is:

$$\frac{\partial(\delta \mathbf{q} \otimes \mathbf{q})}{\partial \delta \boldsymbol{\theta}} = \mathbf{Q}_{\delta\theta} = \frac{1}{2} \left[\mathbf{q}\right]_R \begin{bmatrix} \mathbf{0}^T \\ \mathbf{I}_3 \end{bmatrix} \tag{4.14}$$

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

$$\frac{\partial \mathbf{x}_t}{\partial \delta \mathbf{x}} = \mathbf{X}_{\delta x} = \begin{bmatrix} \frac{\partial \mathbf{p}_{n,t}}{\partial \delta \mathbf{p}_n} & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \frac{\partial \boldsymbol{\beta}_{\omega,t}}{\partial \delta \boldsymbol{\beta}_\omega} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_{\delta\Theta} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_9 \end{bmatrix} \tag{4.15}$$

4.4 Error injection into the nominal-state

After performing an update on the error-state, its mean has to be injected into the nominal-state. All quantities require a simple summary of the nominal- and error- states, except for the rotation which can be performed as a left-side quaternion multiplication by the error quaternion. These operations were provided in Table 3.1, therefore the injection procedure:

$$\mathbf{p}_n = \mathbf{p}_n + \hat{\delta \mathbf{p}}_n \tag{4.16a}$$

$$\mathbf{q}_n = \mathbf{q}\{\hat{\delta \boldsymbol{\theta}}\} \otimes \mathbf{q}_n \tag{4.16b}$$

$$\mathbf{v}_b = \mathbf{v}_b + \hat{\delta \mathbf{v}}_b \tag{4.16c}$$

$$\boldsymbol{\beta}_a = \boldsymbol{\beta}_a + \hat{\delta \boldsymbol{\beta}}_a \tag{4.16d}$$

$$\boldsymbol{\beta}_\omega = \boldsymbol{\beta}_\omega + \hat{\delta \boldsymbol{\beta}}_\omega \tag{4.16e}$$

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 $\delta \mathbf{x}^+ =$

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

$$\delta\mathbf{p}_n^+ = \delta\mathbf{p}_n - \hat{\delta\mathbf{p}}_n \tag{4.17a}$$

$$\delta\boldsymbol{\theta}^+ = 2\begin{bmatrix} 0 & \mathbf{0}^T \\ \mathbf{0} & \mathbf{I}_3 \end{bmatrix}\left(\mathbf{q}\{\delta\boldsymbol{\theta}\} \otimes \mathbf{q}\{-\hat{\delta\boldsymbol{\theta}}\}\right) \tag{4.17b}$$

$$\delta\mathbf{v}_b^+ = \delta\mathbf{v}_b - \hat{\delta\mathbf{v}}_b \tag{4.17c}$$

$$\delta\boldsymbol{\beta}_a^+ = \delta\boldsymbol{\beta}_a - \hat{\delta\boldsymbol{\beta}}_a \tag{4.17d}$$

$$\delta\boldsymbol{\beta}_\omega^+ = \delta\boldsymbol{\beta}_\omega - \hat{\delta\boldsymbol{\beta}}_\omega \tag{4.17e}$$

In fact, the error-state is not directly estimated by the filter, as a result only the mean and the covariance matrix have to be updated. Obviously, the mean resets to zeros, but the covariance matrix update depends on $g(.)$, thus the full error reset:

$$\hat{\delta\mathbf{x}} = \mathbf{0} \tag{4.18a}$$

$$\mathbf{P}^+ = \mathbf{G}\mathbf{P}\mathbf{G}^T \tag{4.18b}$$

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

$$\mathbf{G} = \left.\frac{\partial g}{\partial \delta\mathbf{x}}\right|_{\hat{\delta\mathbf{x}}} = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} + \left[\frac{1}{2}\hat{\delta\boldsymbol{\theta}}\right]_\times & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_9 \end{bmatrix} \tag{4.19}$$

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

CHAPTER 5

DEVELOPMENT

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

During the development, I had to integrate a filter into the simulation, which is mainly based on a user-defined Matlab function block. This block gets the true-state of the aircraft as input and uses this data to project feature points on a camera screen. It also runs an ESKF at 50 Hz, which performs a measurement update after every 10^{th} prediction meaning 5 Hz update rate. The results which will be presented are created with simulated noises. The noises are assumed to be isotropic Gaussian white noise, which means their distribution is $\mathcal{N}(0, \sigma\mathbf{I})$. The exact dispersion parameters are presented in Table 5.1.

Name	Notation	Value	UoM
Angular velocity measurement noise	σ_{η_ω}	1	$\frac{^\circ}{\text{s}}$
Acceleration measurement noise	σ_{η_a}	0.01	$\frac{\text{m}}{\text{s}^2}$
Accelerometer bias noise	$\sigma_{\eta_{\beta_a}}$	5	$\frac{\text{mg}}{\sqrt{\text{Hz}}}$
Gyroscope bias noise	$\sigma_{\eta_{\beta_\omega}}$	100	$\frac{^\circ}{\text{h}\sqrt{\text{Hz}}}$

Table 5.1: Noise summary

5.1 Camera configuration

In the real-world realization, a Basler camera will be used, therefore its parameters were used to configure camera projection. The focal length is 1177.5 px, and

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

the size of the image is 2048×1536 px. The principal point is the image's center, thus P_x=1024 px, P_y=768 px.

During the simulation, I placed a huge amount of feature points on the Z=0 plane under the flight path. The angle of the camera (β) is $-45°$, therefore most of the time a lot of feature points are visible in the camera image. The number of visible feature points through time is shown in Figure 5.1.

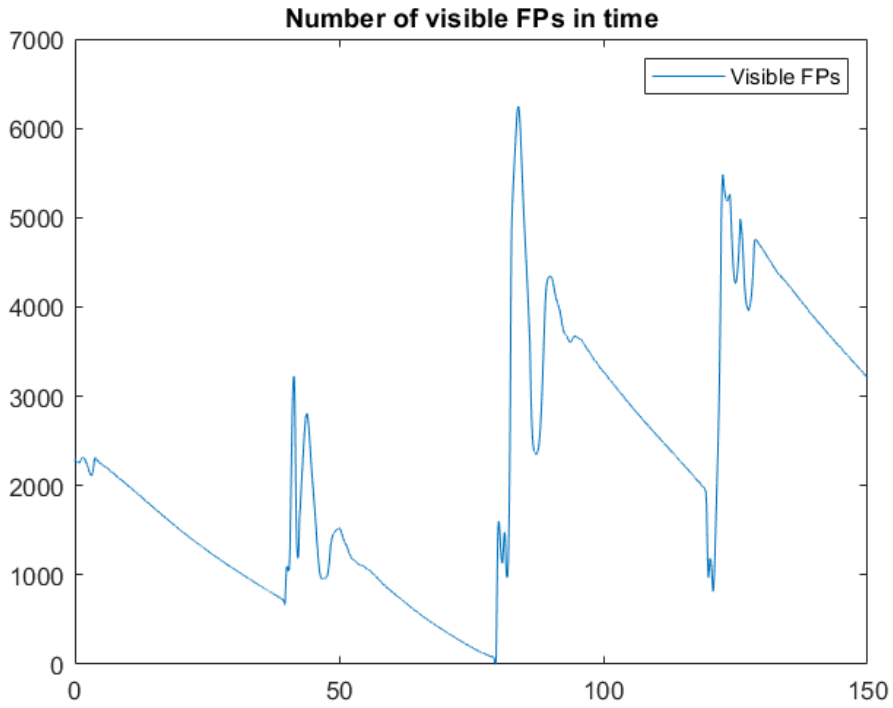


Figure 5.1: Number of visible FPs

It is noticeable the number of visible feature points drops down to 0 around 80 s, which applies only for a sample time interval. Most likely it happens because at that time the aircraft is close to the border of the feature point area and performs a turn. It will be shown in Figure 5.6 that the roll- and pitch angles become large during a turn, therefore the camera does not look down enough to see the feature point area.

Based on [19], the best practice is to choose feature points equally from each side of the image, because this is the best for line fitting and the feature points move mostly at the edges of the image. To sum up, the current solution selects the first 5-5 visible feature points in a band from the left, right, bottom, and top edges of

the image. The width of the edges is configurable, an example from the simulation can be seen in Figure 5.2 with 200 px width.

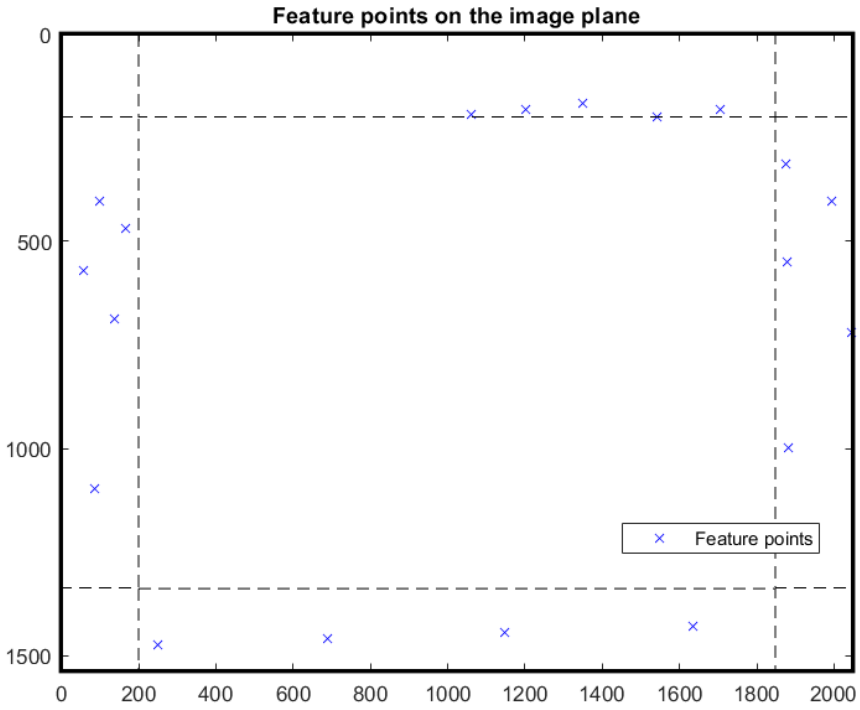


Figure 5.2: Camera image example

5.2 Simulation setup

The development was done in successive steps, hence initially, the nominal equations were integrated over time using ideal values. On the first try, I used $\mathbf{g} = 9.81\frac{\mathrm{m}}{\mathrm{s}^2}$ which resulted in a few meter difference for X-, Y- and a few tens of meters in Z-coordinates. It was caused by the fact that in the WGS84 convention, the gravitational acceleration differs a bit from the value 9.81 m/s^2 and changes with the position. After choosing $\mathbf{g}$ as the mean of the simulation values, the results have shown a minimal, a few cm difference in X-, Y- and 1-2 m in Z-coordinate. The differences are not only caused by inaccuracies in the gravitational acceleration but also by numerical errors due to the finite number representation.

Once the integration of the nominal equations resulted in satisfactory achievements, I added camera measurements to the simulation. Firstly, I have run a few

tests along a straight trajectory but it caused small changes in the signals, especially in the angles, therefore I have set up a trajectory to fly along the sides of a square, shown in Figure 5.3.

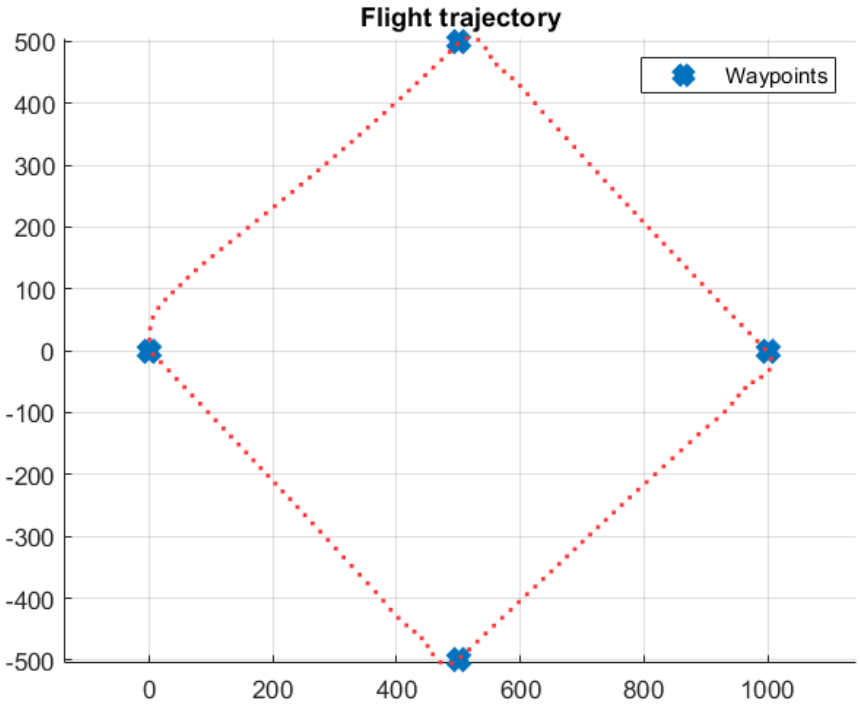


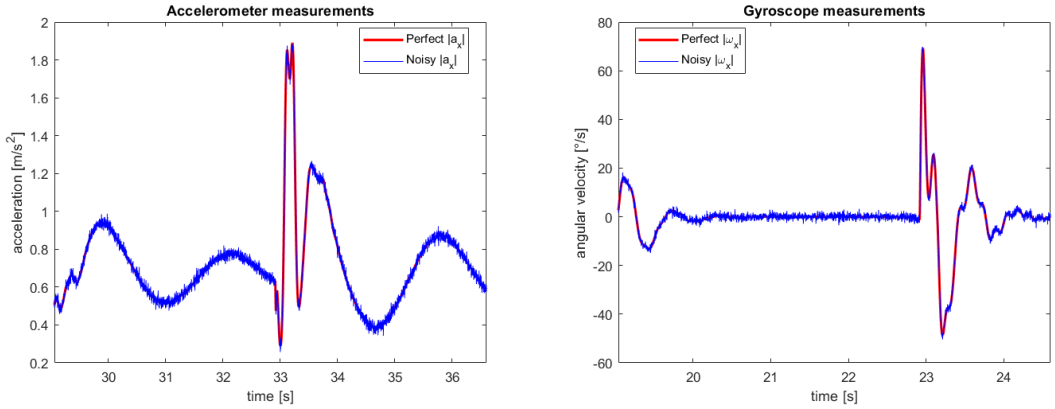
Figure 5.3: Flight path

5.3 Results

To evaluate the performance of the filter, some measure is needed, such as the "perfect" values known from simulation. Another good measure is to run a 2nd version of the nominal-state, which is calculated only by prediction without measurement update.

During the simulation, the filter got the noisy IMU measurements as input. Since the noises are isotropic, I have just provided the noisy measurements along X-axis in Figure 5.4.

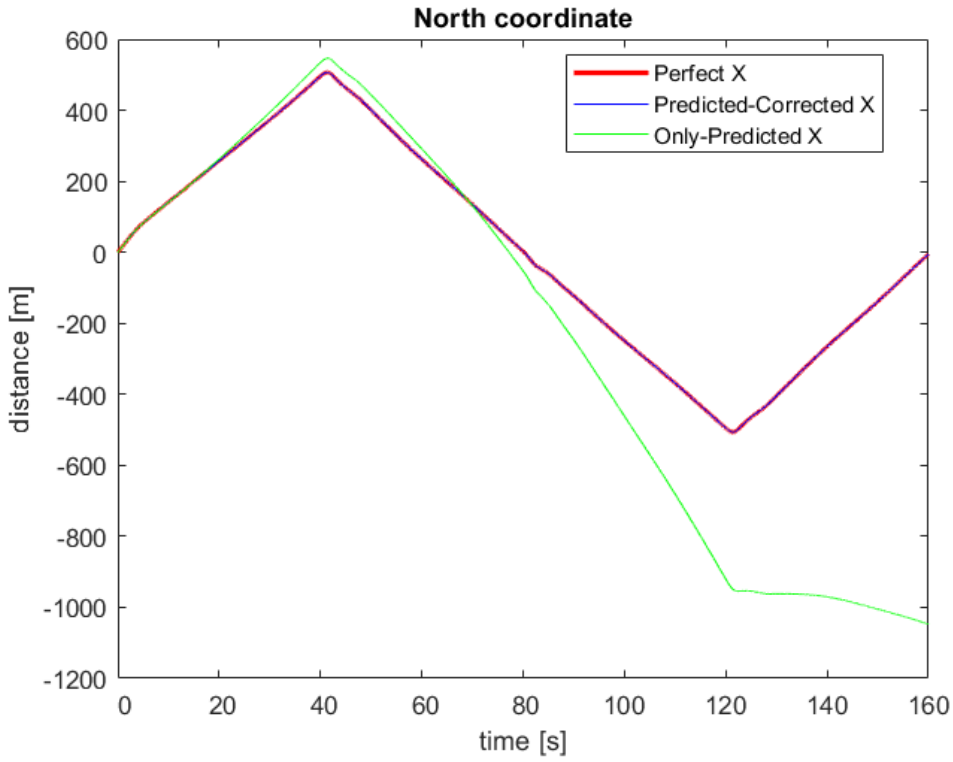
In light of lacking the simulation of sensor biases, the results are granted only for the position, orientation, and velocity in Figures 5.5-5.7. Every figure contains the true-, predicted-corrected-, and only-predicted values with red, blue, and green colors.



(a) Accelerometer **(b)** Gyroscope

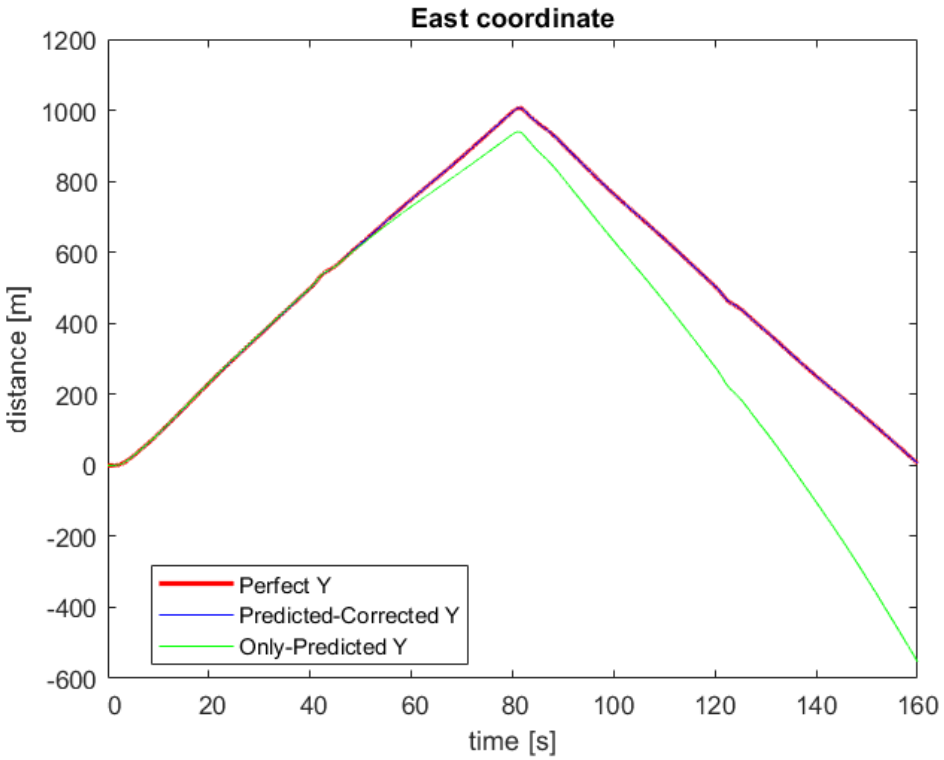
Figure 5.4: IMU measurements

To wrap up, the only-predicted values differed significantly at the end of the simulation due to accumulative error, this phenomenon was already referred to as drift. The noise in the accelerometer measurements spoils the velocity estimates, which leads to spoiling the position estimates too. However, it is apparent the orientation estimates remain close to the ideal. On the other hand, the predicted and corrected estimates follow pretty well the ideal values, and drift can not be seen.

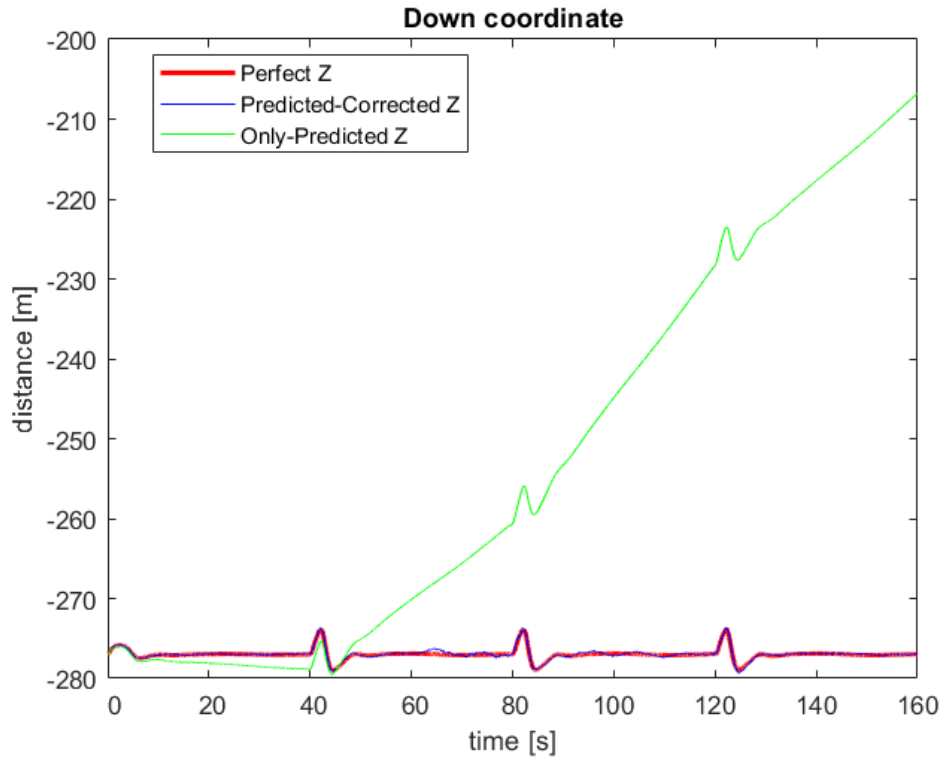


(a) North

Figure 5.5: NED coordinates

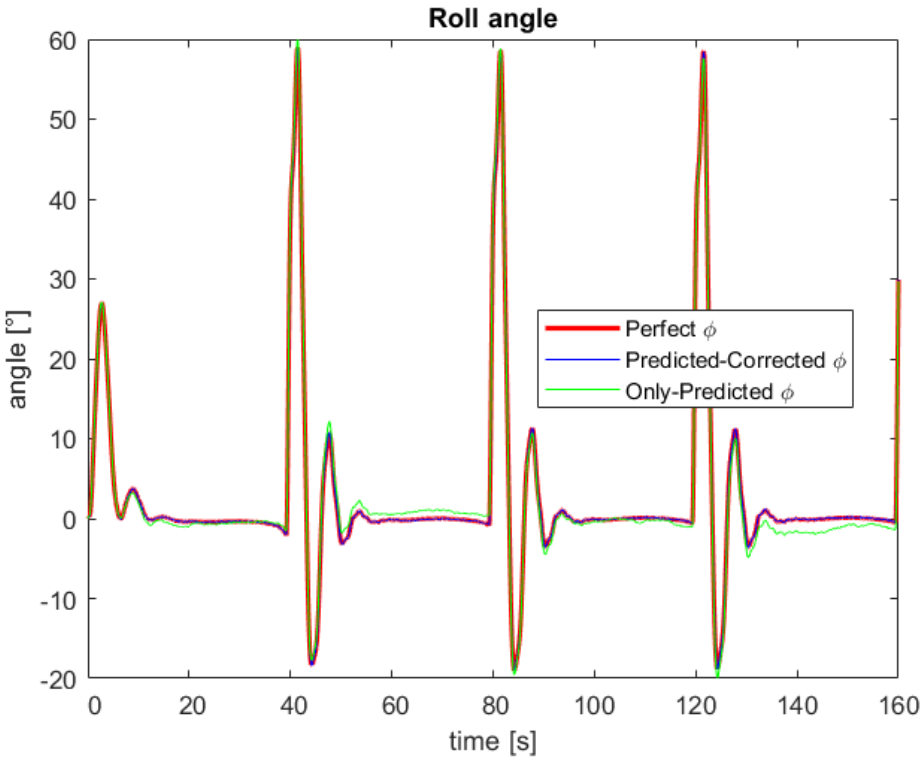


(b) East

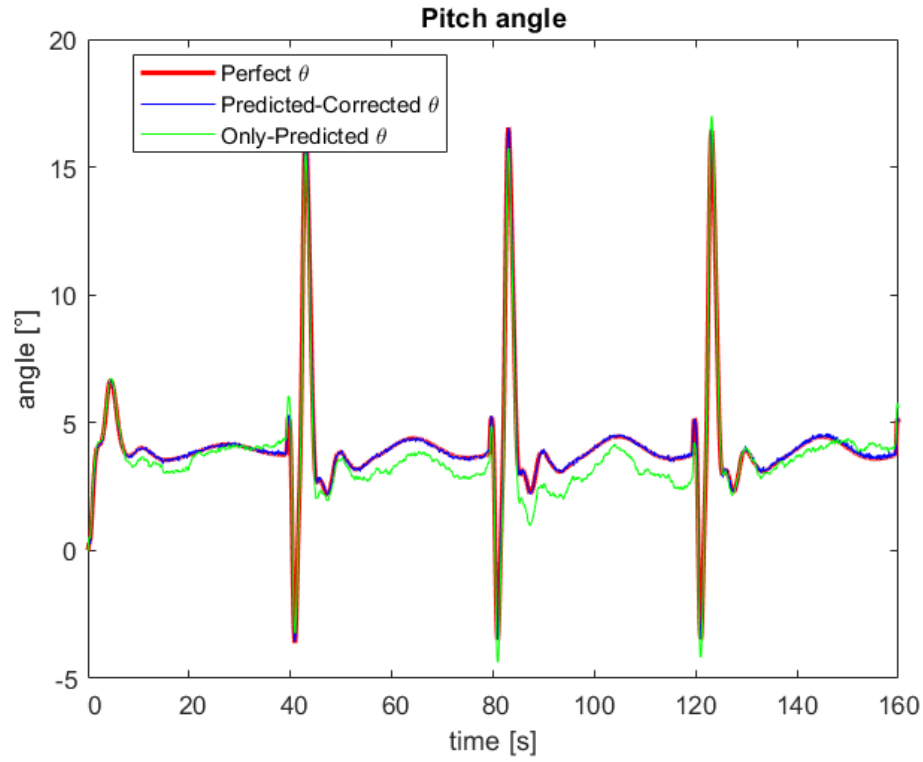


(c) Down

Figure 5.5: NED coordinates (Continued)

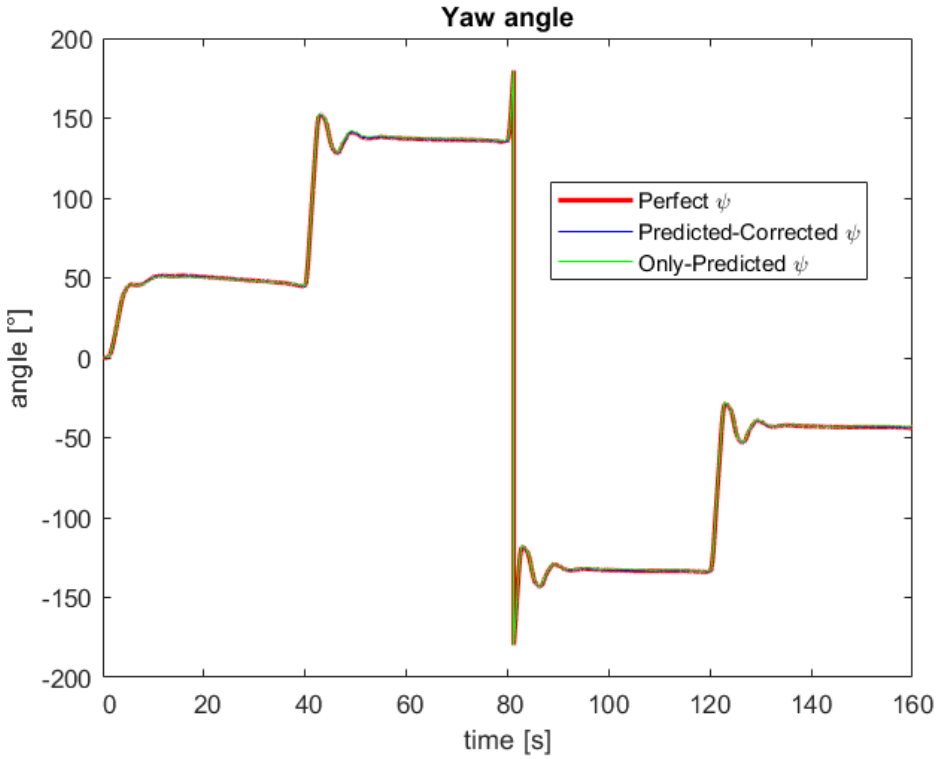


(a) Roll



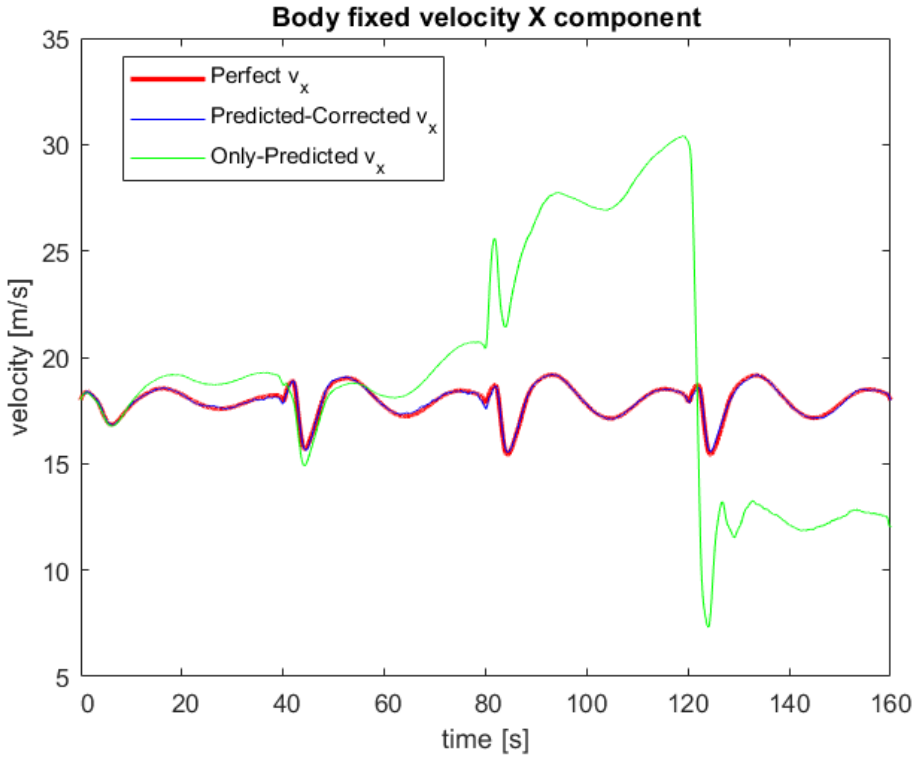
(b) Pitch

Figure 5.6: Euler angles



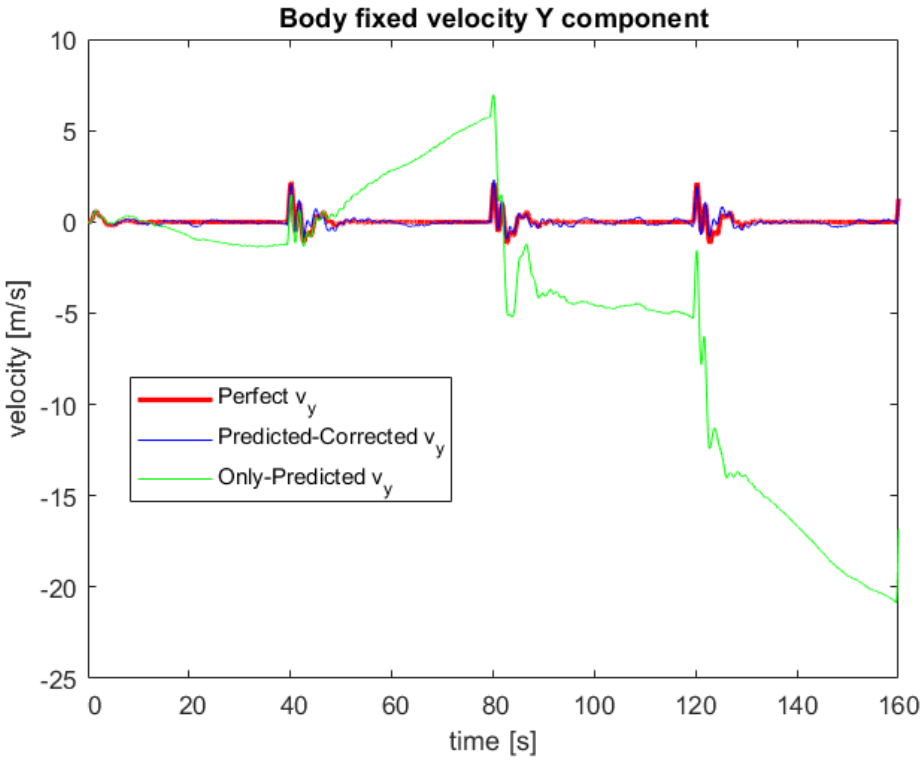
(c) Yaw

Figure 5.6: Euler angles (Continued)

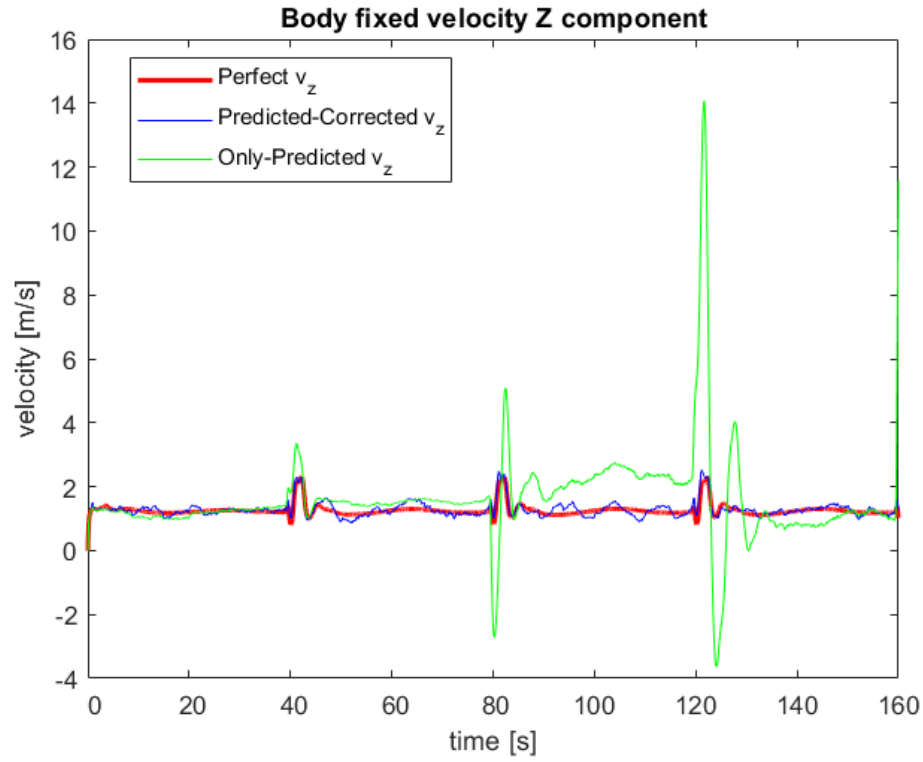


(a) X component

Figure 5.7: Body-fixed velocity



(b) Y component



(c) Z component

Figure 5.7: Body-fixed velocity (Continued)

5.3.1 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 $\hat{a}$ and observed a values. Mathematically, it can be expressed as:

$$RMSE(a) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{a} - a)^2} \tag{5.1}$$

As a final point of the performance analysis, the RMSE values are presented in Table 5.2.

Variable	Predicted-Corrected RMSE	Only-Predicted RMSE
X	0.16 m	388.82 m
Y	0.2 m	204.81 m
Z	0.15 m	34.49 m
ϕ	5.61°	49.63°
θ	3.19°	34.44°
ψ	4.72°	232.21°
v_x	0.07 m/s	5.59 m/s
v_y	0.16 m/s	8.77 m/s
v_z	0.13 m/s	1.44 m/s

Table 5.2: RMSE values

The corrected values have far smaller RMSE, than without correction. Furthermore, I would like to emphasize that, the only-predicted values are much worse for those variables which change a lot during the simulation for example X, Y, and yaw but better for variables that have smaller dynamics. In the case of predicted-corrected values, there is no visible correlation like that.

CHAPTER 6

CONCLUSION

In summary, the first semester can be regarded as successful, as the first working version of the filter is ready. However, further advancements are necessary to ensure its practical viability in real-world scenarios. These enhancements encompass the following:

1. The assimilation of sensor biases into the simulation.
2. The provision of position estimates within the "jumping" node frame.
3. The implementation of a least squares optimization algorithm capable of deriving the Earth coordinates of feature points.
4. The integration of genuine image processing algorithms into the ESKF framework, leading to:
   (a) The incorporation of camera imperfections, such as lens distortion errors into the simulation.
   (b) The enhancement of the measurement update process to rely not solely on a single camera image but also on track information pertaining to feature points.
5. The development of a back-end component for the filter.

APPENDIX A

QUATERNION OPERATIONS

All of the following formulas can be found in [18].

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

A.1 Rotation vector to quaternion formula

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

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

A.2 The time derivative of quaternion

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

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

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

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

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

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

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

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

A.3 Jacobian with respect to the vector

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

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

A.4 Jacobian with respect to the quaternion

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

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

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

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

Using these results the Jacobian with respect to quaternion is:

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

APPENDIX B

ESKF-RELATED EQUATIONS

All of the following formulas with little difference can be found in [18].

B.1 True rotation matrix

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

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

Considering the time derivative of the above yields:

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

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

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

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

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

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

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

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

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

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

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

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

B.2 Error-state equations

The error-state can be expressed easily as the composition of the true-state (3.5) and the nominal-state (3.7). As it was detailed in Chapter 3 the error state remains small therefore second-order errors are negligible.

B.2.1 Position error

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

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

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

B.2.2 Angular error

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

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

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

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

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

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

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

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

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

B.2.3 Velocity error

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

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

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

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

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

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

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

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

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

B.4 Jacobian of reset function (g) by the error rotation vector

The goal is to determine the derivative of (4.17b) by the error rotation vector $\delta\boldsymbol{\theta}$. Firstly, I aim to formalize that equation with rotation vector parameters, neglecting the constant terms:

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

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

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

The above equation splits into a scalar- and a vector- part, where the scalar part is formed by second-order infinitesimals, therefore it is not very useful, additionally the constant member in the original (4.17b) equation is responsible for exactly this neglect. Only considering the vector part, the Jacobian:

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

ABBREVIATIONS

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

List of Figures

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

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

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

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

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

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

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

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

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

[17] 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.

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

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