TÉCNICO LISBOA

Cooperative Position and Orientation Estimation for Multi-Vehicle Systems

Francisco José Machado e Moura

Thesis to obtain the Master of Science Degree in

Aerospace Engineering

Supervisor(s): Professor Rita Maria Mendes de Almeida Correia da Cunha
Professor João Pedro Castilho Pereira Santos Gomes

Examination Committee

Chairperson: Prof. Afzal Suleman
Supervisor: Prof. Rita Maria Mendes de Almeida Correia da Cunha
Member of the Committee: Prof. Bruno João Nogueira Guerreiro

October 2021

Dedicated to someone special...

Acknowledgments

I would like to begin by thanking my supervisors, Professor Rita Cunha and Professor João Pedro Gomes. Even though I had contact with control and signal processing during the course, my experience in this field at the start of this thesis was very limited. Their guidance and insights were invaluable in formulating the research, methodology and implementation.

I want to thank my family. You have always been available to listen and encourage me. Last but not least, I want to give a warm thank you to the friends I made along this journey at Técnico and at Residência de Estudantes Eng.º Duarte Pacheco, specially Beatriz Alves, Diogo Oliveira, João Ferreira, João Moita, Luís Alves, Pedro Pinheiro, Rita Costa and Tiago Alves.

Resumo

Os UAVs [1] têm sido utilizados há muito tempo tanto em aplicações militares como comerciais. A sua procura crescente levou a uma diminuição do seu preço e, por conseguinte, desencadeou um aumento da acessibilidade e diversidade. Esta mudança no setor impulsionou a implementação dos UAVs em alguns dos trabalhos mais perigosos dentro do setor comercial.

Estas implementações economicamente vantajosas vão desde a recolha de dados à gestão de resíduos. Definidos por dimensões compactas, conveniência, capacidade de descolagem vertical e agilidade, os multi-rotors são a variação mais comum.

A estimativa e controlo cooperativos são tópicos de investigação prósperos no campo dos sistemas multiagentes. A cooperação pode permitir ou melhorar o desempenho na execução de tarefas, através da deteção local e troca de informação.

Com o mapeamento local a ser demasiado complexo e com elevado consumo de energia, num ambiente sem GPS [2] e sem conhecimento prévio da localização, as tarefas de piloto automático não podem ser executadas, tornando a estimação cooperativa a opção mais adequada. Esta tese centra-se na investigação de diferentes topologias de interação e na seleção adequada de medições locais para permitir a estimação coletiva da posição e orientação de todos os veículos.

Esta tese concentra-se nos conceitos fundamentais de um algoritmo de estimação da posição e do rumo de um UAV para um conjunto muito específico de condições, tais como, a falta de sinal GPS sob o tabuleiro de uma ponte.

A estimativa de rumo implica conceitos de referenciais locais e inerciais, ângulos de Euler, matrizes de rotação e sobre os sensores necessários, tais como giroscópio, acelerómetro e magnetómetro.

Os filtros são fundamentais para obter a melhor estimativa possível com os sensores e modelos disponíveis. Devido às suas ótimas capacidades de estimação, o Filtro de Kalman, e as suas diferentes variantes têm sido utilizados na navegação autónoma há vários anos. Por conseguinte, 4 dos algoritmos de filtragem utilizados são o Filtro de Kalman, o Extended Kalman Filter e um filtro complementar como Filtro de Kalman, bem como um filtro não linear, de modo a obter resultandos diferentes em função das necessidades e restrições do meio.

Na teoria de controlo, a filtragem de Kalman, também conhecida como estimativa quadrática linear, é um algoritmo que utiliza uma série de medições observadas ao longo do tempo, contendo ruído estatístico e outras imprecisões, e produz estimativas de variáveis desconhecidas que tendem a ser mais precisas do que aquelas baseadas apenas numa única medição, estimando uma distribuição conjunta de probabilidade sobre as variáveis para cada período de tempo.

Os principais resultados foram obtidos utilizando o Extended Kalman Filter e no filtro não-linear. A introdução do filtro não-linear não pode ser avaliada apenas com base no desvio RMS. A necessidade de menos computing power e baterias devido ao menor número de sensores pode ser uma vantagem em algumas situações.

Palavras-chave: UAV's, Estimação Cooperativa, Filtros, Filtro de Kalman, Posição, Orientação.

[1] Unmanned Aerial Vehicles
[2] Global Positioning System

Abstract

UAVs have long been used in both military and commercial applications. Its current increasing demand has led to a decrease in one's price and, therefore, triggered a rising accessibility and diversity. This shift in the sector boosted the UAVs' implementation in some of the most dangerous jobs within the commercial sector.

These cost-effective implementations range from data collection to waste management. Defined by compact dimensions, convenience, vertical take-off capability and agility, multi-rotors are the most common variation.

Cooperative estimation and control are thriving topics of research within the field of multiagent systems. Cooperation may enable or improve performance in task execution, through local sensing and exchange of information.

In a GPS denied environment without previous knowledge of the surrounding area, autopilot tasks cannot be executed, making cooperative estimation a better suited option. This thesis is focused on investigating different interaction topologies and the adequate selection of local measurements to enable the collective estimation of position and orientation of all vehicles.

This thesis provides an overall view on the fundamental concepts of an UAV's position and heading estimation algorithm for a very specific set of conditions, such as, the lack of GPS signal under a bridge deck.

The orientation estimation entails concepts of local and inertial referential, Euler angles, rotation matrices and about the sensors needed, such as the rate gyroscope, the accelerometer, and the magnetometer.

Filters are fundamental to obtain the best estimation possible with the sensors and models available. Due to its optimal estimation capabilities, the Kalman Filter, and its different variants have been used in autonomous navigation for several years. Therefore, 4 of the filtering algorithms used are Kalman Filter, Extended Kalman Filter and a complementary filter as Kalman Filter, as well as a non-linear filter, in order to obtain different results depending on the needs and constraints of the environment.

In control theory, Kalman filtering, also known as linear quadratic estimation, is an algorithm that uses a series of measurements observed over time, containing statistical noise and other inaccuracies, and produces estimates of unknown variables that tend to be more accurate than those based on a single measurement alone, by estimating a joint probability distribution over the variables for each time frame.

The implementation of this thesis is based on two major parts, the block diagrams, and the modelling of the state space. In the quest for the optimal solution fifteen configurations were implemented and tested with the results being discussed further on.

The main results were based on the Extended Kalman Filter and the non-linear filter. The introduction of the non-linear filter cannot be evaluated only based on the RMS deviation though. The need for less computing power and batteries due to the fewer sensors can be a plus in some situations.

Keywords: UAV's, Cooperative Estimation, Filtering, Kalman Filter, Position, Orientation.

Contents

List of Tables

List of Figures

Nomenclature

Greek symbols

β	Bias Term.
η	Noise Term.
μ	Mean.
ψ, θ, ϕ	Euler Angles.
σ	Variance.

Roman symbols

A, B, C, D	State-Space Matrices.
E, F, G, H	State-Space Matrices.
h	Output Function.
J	Jacobian Matrix.
P, K	Kalman Filter Matrices.
u	Control Vector.
x	State.
y	Output.
a	Linar Acceleration.
l	Gain.
p	Position.
S	Skew Matrix.
d	Relative Position.
p, q, r	Angular Velocity.
R	Rotation Matrix.

u, v, w Linear Velocity.

V Local Referential.

Subscripts

i, j Drone indexes.

x, y, z Cartesian components.

Superscripts

. Time Derivative.

ˆ Estimate.

˜ Error.

O Orientation.

P Position.

T Transpose.

Chapter 1

Introduction

1.1 Motivation

UAVs have long been used in both military and commercial applications, first and foremost by defence organisations and briefly after by civil early adopters. However, this technology still has plenty of room to grow. Its current increasing demand has led to a decrease in one's price and, therefore, triggered a rising accessibility and diversity. This shift in the sector boosted the UAVs' implementation in some of the most dangerous jobs within the commercial sector.

These cost-effective implementations range from data collection to waste management. With continuous research and development in position and orientation estimation as well as image based technologies, drones' implementation may cover more and more complex tasks in the future.

The chart represents the state of the drone's industry according to the law of innovation diffusion, Figure 1.1. The blue curve represents the diffusion of innovation in a predictable path with five distinct phases. The yellow line is a cumulative depiction of total market share.

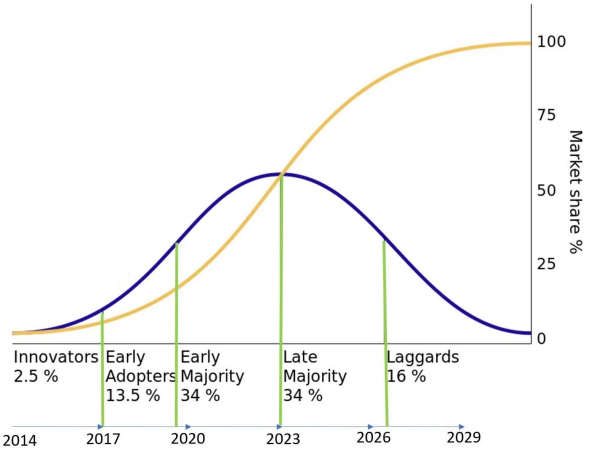

Figure 1.1: Drone industry according to the law of innovation diffusion [1].

According to PwC, drones' applications in the commercial sector is valued at over $127B and still growing [2].

Defined by compact dimensions, convenience, vertical takeoff capability and agility, multi-rotors are the most common variation.

A quadrotor is a structure with 4 arms, each with a rotor in the end. Cooperative estimation and control are thriving topics of research within the field of multiagent systems. Cooperation may enable or improve performance in task execution, through local sensing and exchange of information.

1.2 Topic Overview

In recent years UAVs have been capturing the attention and interest in a broad range of applications, such as search and coverage missions, exploration, cooperative manipulation and control. In each of these cases, the problem of cooperative localisation in a group of robots has to be solved. Cooperative localisation consists in improving the positioning capacity of each robot through the exchanges of information with other robots in the group [3].

Relative localisation between UAVs is a requisite for cooperative localisation and orientation estimation in the case when absolute localisation information such as information from global positioning system (GPS) is unavailable or to inaccurate to be used, which can happen indoors, in urban environments and even forest and remote locations. Taking this in consideration, relative localisation is a key parameter in cooperative UAV systems [4].

A convenient solution for relative location is to obtain the distance and bearing information using cameras and AI methods. Nevertheless, there is the limitation of cameras only being able of operate within a limited range and are prone to suffer from occlusion and lighting conditions. In opposition, distance measurements can be obtained using different sensors such as ultra-wideband (UWB), radars, and lidars, which can operate over a much larger range [4].

There are, already, several methods used for solving cooperative localisation problems, such as the Extended Kalman Filter (EKF) for a centralised system, or, if the computation is decentralised and the communication is unreliable, other techniques like Covariance Intersection or Interleaved Update. Approaches that assume bounded errors using polytopes and linear programming algorithms have also been proposed [3].

Related Work

In the article Attitude and Heading Reference System Based on3D Complementary Filter, the relationship between the measured signals, on each axis in the sensor coordinates, and the inertial coordinates were defined by the Euler angles, respectively by roll (ϕ), pitch (θ) and yaw (ψ) angles, as a specific sequence of rotation with respect to x, y and z axes of a referred coordinate system. The axes of inertial coordinate system were defined as a right handed Cartesian coordinate system with North, West and Upward (NWU) directions [5].

In the article Development of attitude and heading reference system, the Euler angles were utilised to find the transformation matrix which relates the North-East-Down (NED) frame and the body frame. Furthermore, the Euler angles were determined from the gyro- scope, magnetometer and accelerometer to implement the Complementary Filter [6].

In the article A Simple Attitude Unscented Kalman Filter: Theory and Evaluation in a Magnetometer-Only Spacecraft Scenario a quaternion-based attitude unscented Kalman filter was formulated with quaternion errors parametrized by small angle approximations considering a magnetometer only spacecraft scenario. The method was applied to a filter with a state vector consisting of the attitude quaternion and the gyro bias vector [7].

A generalised complementary filter (GCF) was proposed in the article Generalized complementary filter for attitude estimation based on vector observations and cross products, for attitude estimation. The GCF was based on the vector observation and its cross product. The results from the GCF had better numerical stability and much higher computational efficiency than the multiplicative extended Kalman filter (MEKF) [8].

1.3 Problem Statement

Scenario Description

The problem of cooperative position and orientation estimation in multi-agent systems is formulated in this thesis considering a flight in formation with three drones labelled $1, 2, 3$, as shown in Figure 1.2. The formation is composed by a leader aircraft, 3, at unknown location at time t, followed by two drones, 1 and 2, each equipped with a GPS receiver.

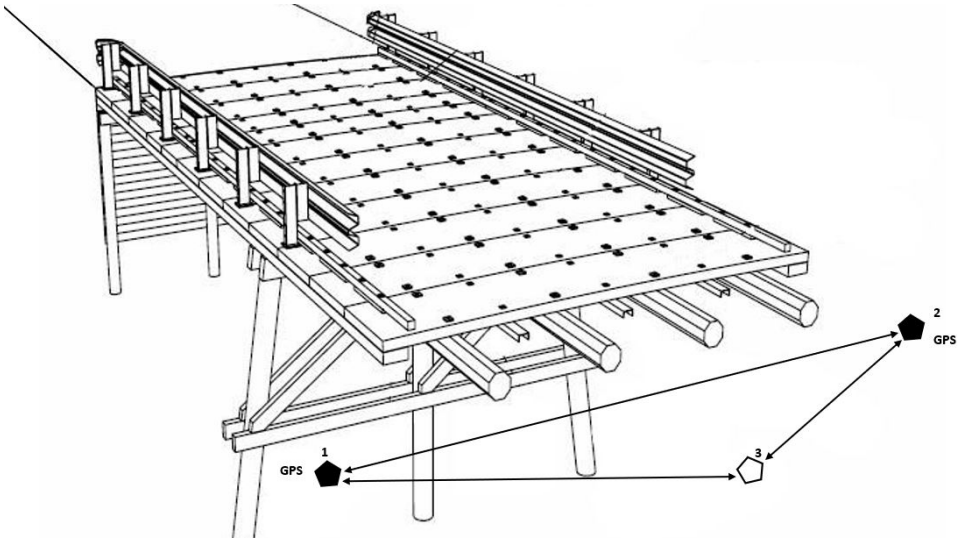

Figure 1.2: Context illustration. [9]

In a GPS denied environment without previous knowledge of the surrounding area, Autopilot tasks cannot be executed, making cooperative estimation a better suited option.

Approach Overview

The work is focused on investigating the correlation between number of drones used and the results in the cooperative estimation of the position. The different interaction topologies between drones and the adequate selection of local measurements. As well as the better suited filtering techniques to enable the collective estimation of position and orientation of all vehicles taking in consideration the problem constraints.

1.4 Thesis Outline

This thesis provides an overall view on the fundamental concepts of an UAV's position and heading estimation algorithm for a very specific set of conditions, such as, the lack of GPS signal under a bridge deck. The workflow is described bellow. The first chapter provides a short introduction to the problem being solved and an insight to the proposed methodology. The second chapter, the background, provides important concepts about attitude and position of an UAV, the sensors used, and filtering techniques that are crucial for proper understanding of the following chapters. In chapter 3, the reader can find the description of the implementation explaining how the problem was solved. From block diagrams of the simulink implementation, to their modeling. The results achieved by the proposed configurations are exposed on the fourth chapter, which exposes the performance of the number of drones, the filter techniques chosen and the sets of sensors. The conclusions drawn from this work and ideas of future work can be found on chapter 5.

Chapter 2

Background

As previously mentioned, the problem of cooperative position and orientation estimation in multi-agent systems can be formulated, for instance, considering a flight in formation with three drones labelled $1, 2, 3$, as shown in Figure 1.2. The formation is composed by a leader aircraft, 3, at unknown location at time t, followed by two drones, 1 and 2, each equipped with a GPS receiver.

This chapter provides the fundamental concepts for the implementation of an UAV's position and heading estimation algorithm for a very specific set of conditions, such as, the lack of GPS signal under a bridge deck.

The first concept introduced is orientation. In order to understand orientation, first of all, two concepts must be introduced, the local and inertial coordinate frames. These two concepts are fundamental to the use of Euler angles, the orientation parameters, and for a proper sensor's output usage. In this chapter, the mathematics behind Euler angles and the rotation matrix are key to the understanding of the orientation's representation. The sensors used are the rate gyroscope, the accelerometer and the magnetometer.

Following the orientation overview, the positioning sensors must be described as well. Once the correct estimation of the third drone position lies on it. The sensors used are the GPS and cameras supported by two vision based algorithms whose output is the relative position and relative orientation between drones.

The last concept mentioned in this section is filtering. Filters are fundamental to obtain the best estimation possible with the sensors and models available. The majority of the filters presented are variations of the Kalman Filter. These variations can be due to the non-linear nature of the model, Extended Kalman Filter, or a Kalman Filter interpretation of Complementary Filtering.

2.1 Orientation

The attitude of an UAV is a crucial aspect in autonomous flight. Not only accuracy but low complexity and low cost parts are fundamental requisites.

In the representation of attitude, several approaches were developed over the years: Euler angles, quaternions, direct cosine matrix and the rotational matrix. In this thesis only Euler angles and the rotational matrix are examined in more detail.

2.1.1 Local and Inertial Coordinate Frames

Each vehicle has its own local referential, $\{V_i\}$, $i \in [1,2,3]$, sympathetic with the drone's kinematic, that is defined by its position, ${}^{I}\vec{p_i}$, $i \in [1,2,3]$, and orientation, $[\phi_i, \theta_i, \psi_i], i \in [1,2,3]$, in an inertial referential, $\{I\}$.

The local referential, $\{V_i\}$, is sympathetic with the drone's kinematic and has its origin on the drone's centre of gravity. It is defined by the x-axis being aligned with the arm of the rotor 1, the y-axis aligned with the arm of the rotor 2 and the z axis perpendicular the xy plane aiming down, as described in Figure 2.1.

The inertial referential, $\{I\}$, has its origin on Earth's surface, and moves sympathetic with Earth, which means it is fixed in those coordinates throughout time. The the x-axis aims North, the y-axis aims East and the z-axis points to Earth's core.

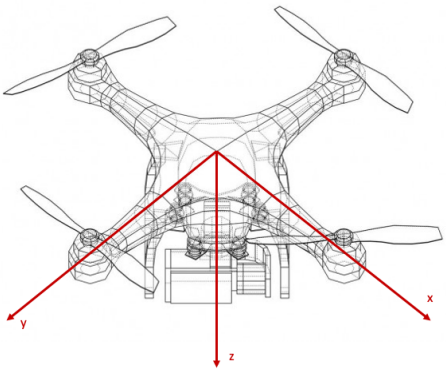

Figure 2.1: Drone's Local Referential, in Red.

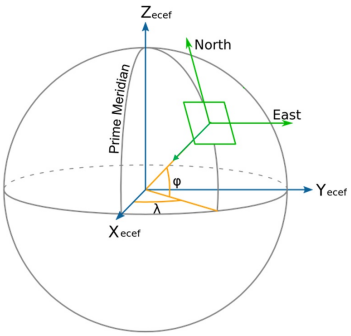

Figure 2.2: Inertial Referential, in Green.

Each aircraft has its position expressed in the inertial referential as:

$$ {}^{I}\vec{p_i} = (x_i, y_i, z_i) \tag{2.1} $$

where, (x_i, y_i, z_i), $i \in [1,2,3]$, represents the location of the ith aircraft in a coordinate system external to all vehicles.

2.1.2 Euler Angles

Although the position of the local referential has been defined regarding the inertial referential, the orientation has not. Intuitively the orientation is defined by three angles, the Euler angles, known as roll, pitch and yaw.

Roll is angle of rotation related to the x-axis, whose notation is ϕ. Pitch is the angle of rotation related to the y-axis, whose notation is θ. Yaw is the angle of rotation related to the z-axis, whose notation is ψ.

This representation of Euler angles are shown in Figure 2.3.

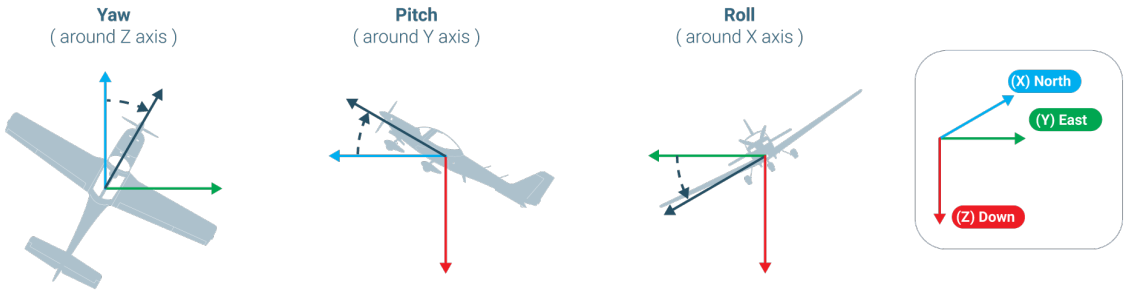

Figure 2.3: Euler angles: sequence of standard rotations.

Although the order of the representation is not compulsory, from now on, the Euler angles are represented by order, as $[\phi, \theta, \psi]$. This representation will be used in the simulations and filters introduced further ahead.

To make a comparison, all translations can be represented as three numbers, x, y and z, as the succession of three consecutive linear movements along three perpendicular axes X, Y and Z axes. The same applies for rotations, all the rotations can be described using three numbers, $[\phi, \theta, \psi]$, as the succession of three rotational movements around three axes that are perpendicular one to the next. This similarity between linear coordinates and angular coordinates makes Euler angles very intuitive, but unfortunately they have a huge disadvantage, the possible loss of a degree of freedom (Gimbal Lock).

A gimbal lock occurs because the map from Euler angles to rotations is not a local homeomorphism at every point, and thus at some points the number of degrees of freedom must drop below 3, at which point gimbal lock occurs. Euler angles provide a means for giving a numerical description of any rotation in three-dimensional space using three numbers, but not only is this description not unique, but there are some points where not every change in rotations can be realised by a change in the Euler angles. This is a topological constraint. [2]

A potential solution to the gimbal lock is to represent the orientation in other way that not with Euler angles. Alternative representations for orientation that can be adopted are rotation matrices, quaternions, or direct cosine matrices (DCM).

[2]Citation from `https://en.wikipedia.org/wiki/Gimbal_lock`

2.1.3 Rotation Matrix

In order to avoid singularities, rotation matrices are used in this thesis instead of Euler angles [10]. For a better understanding of the concept, the first rotation matrix to be defined is in two dimensions (x,y). That means that the only angle in the orientation is yaw, ψ, the rotation related to the z-axis. Figure 2.4 represents the rotation from frame $\{B\}$ to frame $\{A\}$. That is, from a local referential $\{B\}$ to an inertial one $\{A\}$. Coordinates expressed in $\{B\}$ are transformed to coordinates expressed in $\{A\}$, by equation (2.2).

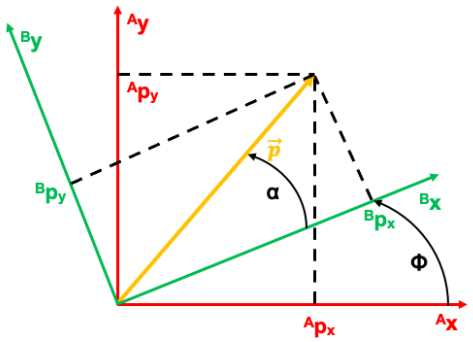

Figure 2.4: Rotation from frame $\{B\}$ to $\{A\}$.

$$ {}^{A}\vec{p} = {}^{A}_{B}R(\psi)\,{}^{B}\vec{p} \tag{2.2} $$

Equation (2.2) in a matrix representation:

$$ \begin{bmatrix} {}^{A}p_x \\ {}^{A}p_y \\ {}^{A}p_z \end{bmatrix} = {}^{A}_{B}R(\psi) \begin{bmatrix} {}^{B}p_x \\ {}^{B}p_y \\ {}^{B}p_z \end{bmatrix} \tag{2.3} $$

If the vector $\vec{p}$ is described in the form $\vec{p} = p\ (cos(\alpha), sin(\alpha), 0)$ instead of $\vec{p} = (x, y, z)$:

$$ \begin{bmatrix} p\ cos(\psi+\alpha) \\ p\ sin(\psi+\alpha) \\ 0 \end{bmatrix} = {}^{A}_{B}R(\psi) \begin{bmatrix} p\ cos(\alpha) \\ p\ sin(\alpha) \\ 0 \end{bmatrix} \tag{2.4} $$

Taking in consideration equation (2.4) and $cos(a+b) = cos(a)cos(b) - sin(a)sin(b)$, $sin(a+b) = sin(a)cos(b) + cos(a)sin(b)$, the Yaw rotation matrix is:

$$ {}^{A}_{B}R(\psi) = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.5} $$

As seen above, a vector defined at any local referential, $\{V_i\}$, can be transformed into the inertial referential, $\{I\}$, employing sequentially the three rotations, each one given by its rotation matrix. The roll and pitch rotation matrices are obtained analogous as equation (2.5).

The Roll rotation matrix of the ith aircraft is defined as:

$$ {}^{I}_{i}R(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi) & cos(\phi) \end{bmatrix} \tag{2.6} $$

The Pitch rotation matrix of the ith aircraft is defined as:

$$ {}^{I}_{i}R(\theta) = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix} \tag{2.7} $$

And the Yaw rotation matrix of the ith aircraft is defined as:

$$ {}^{I}_{i}R(\psi) = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.8} $$

Therefore, the Rotation Matrix of the ith aircraft is defined as:

$$ {}^{I}_{i}R = {}^{I}_{i}\ R(\psi){}^{I}_{i}R(\theta){}^{I}_{i}R(\phi) = \begin{bmatrix} c_\psi c_\theta & c_\psi s_\theta s_\phi - s_\psi c_\phi & c_\psi s_\theta c_\phi + s_\psi s_\phi \\ s_\psi c_\theta & s_\psi s_\theta s_\phi + c_\psi c_\theta & s_\psi s_\theta c_\phi - c_\psi s_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix} \tag{2.9} $$

The notation $c_\xi = cos(\xi)$ and $s_\xi = sin(\xi)$ was used in equation (2.8).

2.1.4 The Rate Gyroscope

In order to obtain orientation information of each drone, some sensors must be used. A gyroscope is a spinning device used for measuring or maintaining orientation and angular velocity.

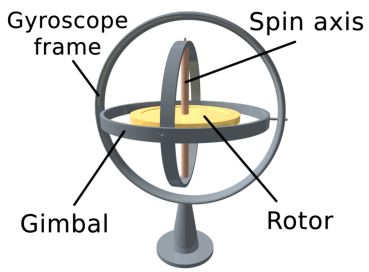

Figure 2.5: 3D Gyroscope component.

It is composed by a rotor in which the spin axis is free to assume any orientation by itself. When

rotating, the orientation of this axis is unaffected by tilting or rotation of the mounting, according to the conservation of angular momentum. A rate gyro is a type of gyroscope that indicates angular rates, that is, $\omega = [p, q, r]$ without a fixed point of reference. Although there is no direct correspondence between the angular rates $[p, q, r]$ and the derivatives of Euler angles, the following approximation for small roll and pitch angles can be used: $[p, q] \approx [\dot{\phi}, \dot{\theta}]$. The advantage of rate gyros over other types of gyros is the fast response rate and relatively low cost. Rate gyros are defined by a drift term and zero mean Gaussian noise, resulting from the manufacturing process. A rate gyro is defined by the three following equations:

$$\begin{cases} y_{gyro,x} = p + \beta_{gyro,z} + \eta_{gyro,x}, & \eta_{gyro,x} \sim N(0, 0.0169) \\ y_{gyro,y} = q + \beta_{gyro,z} + \eta_{gyro,y}, & \eta_{gyro,y} \sim N(0, 0.0169) \\ y_{gyro,z} = r + \beta_{gyro,z} + \eta_{gyro,z}, & \eta_{gyro,z} \sim N(0, 0.0169) \end{cases} \tag{2.10}$$

$[p, q, r]$, represents the angular rates, β_{gyro} the Bias and η_{gyro} the sensor's noise in the readings. The gyro noise, η_{gyro}, is driven by noise of ARW (Angular Random Walk), and a Gaussian can be defined as being zero mean, $\mu \approx 0^{o}$, $\sigma_{gyro} = 0.0169^{o}s^{-1}$ and $\eta_{gyro} \sim N(0, 0.0169)$. This material was withdrawal from the book Small Unmanned Aircraft: Theory and Practice by Timothy W. McLain and Randal Beard [11]. The bias is a drift term driven by noise of RRW (Rate Random Walk) and is defined as a function of time:

$$\beta_{gyro} = f(t); \tag{2.11}$$

In this thesis the bias is defined as not being a function of time, equation (2.12), based on [12].

$$f(t) = \begin{cases} \beta_{gyro,x} = -0.0113 \\ \beta_{gyro,y} = 0.0322 \\ \beta_{gyro,z} = 0.0115 \end{cases} \tag{2.12}$$

2.1.5 The Accelerometer

As seen in 2.1.4, the rate gyro's outputs are angular rates, but the goal is to obtain the angles, not their rate of change, therefore, a couple extra sensors must be introduced.

An accelerometer is a sensor that measures proper acceleration. Proper acceleration is the acceleration of a body in its own instantaneous rest frame. This is different from coordinate acceleration, which is acceleration in a fixed coordinate system. For example, an accelerometer at rest on the surface of the Earth will measure an acceleration due to Earth's gravity, straight upwards of $g = 9.81m/s^2$, which means, that by the definition in Figure 2.2, is opposed to the z-axis. By contrast, accelerometers in free fall (falling toward the centre of the Earth at a rate of about $9.81m/s^2$) will measure zero. The accelerometer can be modelled as the following:

$$\begin{cases} y_{acc,x} = \dot{u} + qw - rv + gsen(\theta) + \eta_{acc,x}, & \eta_{acc,x} \sim N(0, 0.005112) \\ y_{acc,y} = \dot{v} + ru - pw - gcos(\theta)sen(\phi) + \eta_{acc,y}, & \eta_{acc,y} \sim N(0, 0.007291) \\ y_{acc,z} = \dot{w} + pv - qu - gcos(\theta)cos(\phi) + \eta_{acc,z}, & \eta_{acc,z} \sim N(0, 0.1614) \end{cases} \tag{2.13}$$

Once the dynamic accelerations can be neglected in comparison to the static accelerations we got:

$$\begin{cases} y_{acc,x} = gsen(\theta) + \eta_{acc,x}, & \eta_{acc,x} \sim N(0, 0.005112) \\ y_{acc,y} = -gcos(\theta)sen(\phi) + \eta_{acc,y}, & \eta_{acc,y} \sim N(0, 0.007291) \\ y_{acc,z} = -gcos(\theta)cos(\phi) + \eta_{acc,z}, & \eta_{acc,z} \sim N(0, 0.1614) \end{cases} \tag{2.14}$$

The data is based on [13]. This sensor is used to obtain measurements of $[\phi, \theta]$, thus an extra sensor to obtain the measurement of ψ must be incorporated.

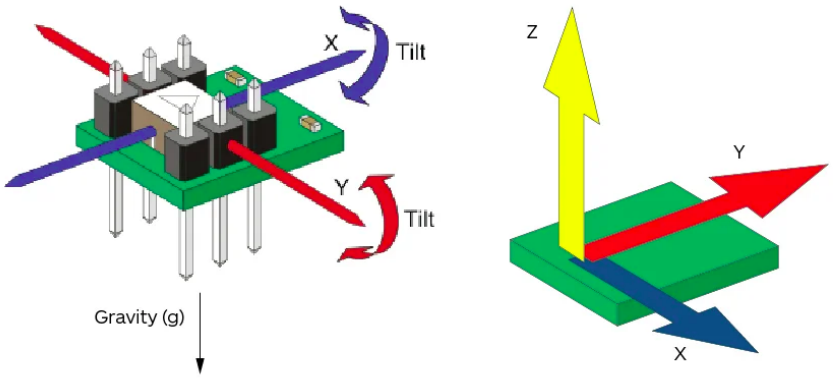

Figure 2.6: Accelerometer sketch from `https://bit.ly/3oU5e3d`

2.1.6 The Magnetometer

The magnetic field sensors can be used as a compass and therefore determine the orientation of the sensor relative to the magnetic north pole. The output is, intuitively, the yaw value, ψ, plus white noise, η_{mag}.

$$y_{mag} = \psi + \eta_{mag}, \qquad \eta_{mag} \sim N(0, 12.399) \tag{2.15}$$

The data was gathered from the article Magnetic Field Sensor Calibration for Attitude Determination [14].

2.2 Position

So that a three dimensional estimation of the position could be obtained, the orientation estimation had to be defined previously, as the sensor's outputs are represented in the local referential and the position of each drone is expressed in the inertial referential. In the scenarios where $[\phi, \theta, \psi] = [0^o, 0^o, 0^o]$ the local and inertial referential change, only, in their positioning.

Two aircraft are equipped with GPS receivers and all three with vision based relative position and relative orientation sensors, all of which are described hereupon. This thesis aims to explore which measurements, or combination of measurements, are better suited to obtain the estimated position of the GPS denied aircraft, 3, with the least possible error as well as the other two aircraft. The position estimates can be obtained using three sets of measurements: GPS coordinates of two aircraft, inter-vehicle distance and inter-vehicle angle. Each aircraft has its position regarding the inertial referential, $\{I\}$, defined as:

$$^I\vec{p_i} = (x_i, y_i, z_i), i \in [1, 2, 3] \tag{2.16}$$

The GPS coordinates are defined as $^I\vec{p_i}$, $i \in [1, 2]$. Each drone, j, also has its relative position regarding the other drones, i, both in local, $\{V_i\}$, and inertial referential, $\{I\}$, defined as:

$$^{V_i}\vec{d_{i,j}}, i \neq j \in [1, 2, 3] \tag{2.17}$$

$$^I\vec{d_{i,j}}, i \neq j \in [1, 2, 3] \tag{2.18}$$

Hence, the distance between drones is defined as:

$$||\vec{d_{i,j}}||, i \neq j \in [1, 2, 3] \tag{2.19}$$

Each Drone has its relative angular position regarding the other drones defined as the normalised relative position vector, that can be expressed both in local, $\{V_i\}$, and inertial referential, $\{I\}$, defined as:

$$\frac{^{V_i}\vec{d_{i,j}}}{||\vec{d_{i,j}}||_{i,j}}, i \neq j \in [1, 2, 3] \tag{2.20}$$

$$\frac{^I\vec{d_{i,j}}}{||\vec{d_{i,j}}||_{i,j}}, i \neq j \in [1, 2, 3] \tag{2.21}$$

Each set of measurements, like distance and angle has its own reliability affected by measurement errors as well as the GPS positioning is affected by data gathering frequency and noise. Where the best set of measurements combinations will be determined through computational simulations.

2.2.1 The GPS

GPS, or Global Positioning System, is a global navigation system that uses satellites, a receiver and algorithms to synchronise location, velocity and time data for air, sea and land travel. The GPS output is

modelled as:

$$\begin{cases} y_{GPS,x} =^{I} p_x + \eta_{GPS,x}, & \eta_{GPS,x} \sim N(\mu_{GPS,x}, \sigma^2_{GPS,x}) \\ y_{GPS,y} =^{I} p_y + \eta_{GPS,y}, & \eta_{GPS,y} \sim N(\mu_{GPS,y}, \sigma^2_{GPS,y}) \\ y_{GPS,z} =^{I} p_z + \eta_{GPS,z}, & \eta_{GPS,z} \sim N(\mu_{GPS,z}, \sigma^2_{GPS,z}) \end{cases} \tag{2.22}$$

The GPS sensor's gathered data, sourced in the Global Positioning System (GPS) Standard Positioning Service (SPS) Performance Analysis Report [15], can be plotted being the number of samples a function of the range error. From the plot analyses the Gaussian can be estimated and the values are in table

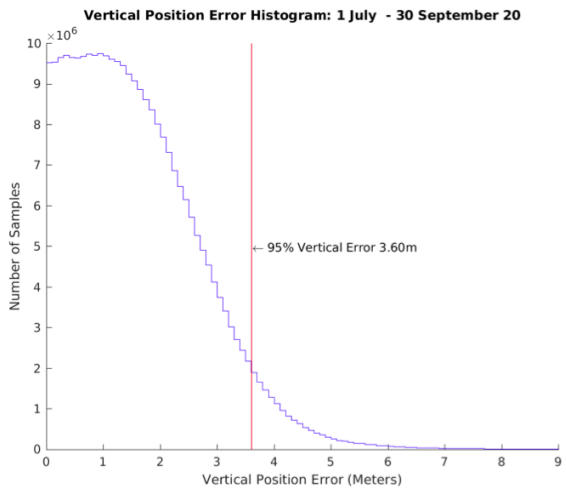

Figure 2.7: GPS - Global Vertical Error Histogram

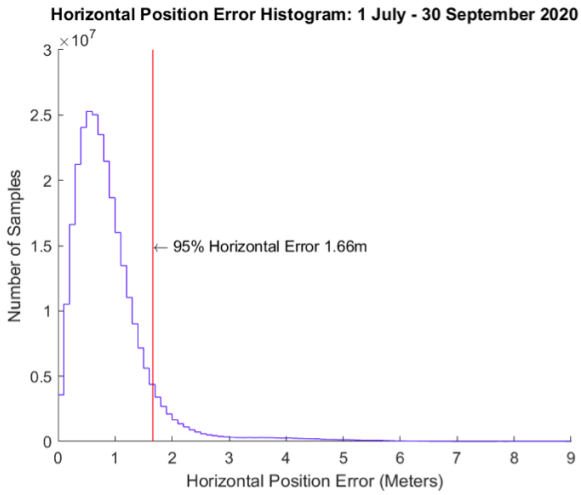

Figure 2.8: GPS - Global Horizontal Error Histogram

2.1. Although the mean is not null, it is considered to be in this approximation. The variance comes from equation 2.23:

$$\mu + 2\sigma = 95\% error \tag{2.23}$$

Which results in the following table:

Axis	$\sigma_{GPS}[m]$	$\mu_{GPS}[m]$	**Gaussian:** $\eta_{GPS,y} \sim N(\mu_{GPS}, \sigma^2_{GPS})$
x:	1.8	≈ 0	$\eta_{GPS,x} \sim N(0, 3.24)$
y:	1.8	≈ 0	$\eta_{GPS,y} \sim N(0, 3.24)$
z:	0.85	≈ 0	$\eta_{GPS,z} \sim N(0, 0.6881)$

Table 2.1: GPS Gaussian parameters table.

2.2.2 The Distance Sensor

The distance sensor is an AI vision based sensor. As seen in the beginning of this section, the distance sensor is based on the relative positioning of the drones. In particular the module of the vectors. And its output is:

$$y_{distance} = ||\vec{d_{i,j}}|| + \eta_{distance}, [i \neq j] \in [1, 2, 3] \qquad \eta_{distance} \sim N(\mu_{distance}, \sigma^2_{distance}) \tag{2.24}$$

The plots shown in Figure 2.9 are from the article Error Evaluation in a Stereo-vision-Based 3D Reconstruction System [16]. The plot above shows the error in the reconstruction of contour. The error

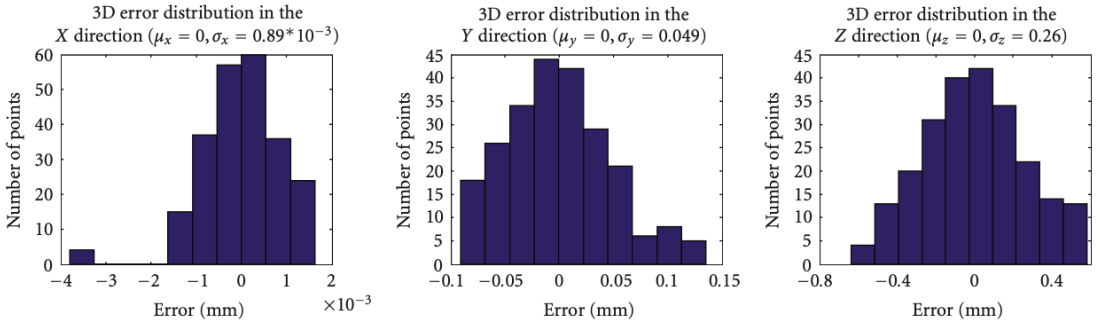

Figure 2.9: 3D error distribution in the x, y, and z-directions of a typical reconstructed contour.

in the distance measurement is proportional to the error of the reconstruction of contour and the distance between drones. Taking in account an expected distance bellow 10 meters and the average of the variances in the plot, table 2.2 was arbitrated.

2.2.3 The Angle Sensor

The angle sensor is also an AI vision based sensor. And as seen in the beginning of this section, the angle sensor is also based on the relative positioning of the drones. In particular the normalised vectors. And its output is:

$$y_{angle} = \frac{\vec{d_{i,j}}}{||\vec{d_{i,j}}||}, [i \neq j] \in [1,2,3] + \eta_{angle} \qquad \eta_{angle} \sim N(\mu_{angle}, \sigma^2_{angle}) \tag{2.25}$$

The similarities between the distance sensor and the angle sensor continue, and the values that define the noise, were, once again, set arbitrarily based on Figure 2.9.

Sensor	σ	μ	**Gaussian:** $\eta \sim N(\mu, \sigma^2)$
Distance	1×10^{-3}	0	$\eta_{distance} \sim N(0, 1 \times 10^{-6})$
Angle	7×10^{-4}	0	$\eta_{angle} \sim N(0, 5 \times 10^{-7})$

Table 2.2: Angle sensor Gaussian parameters table.

2.3 Filtering

Due to its optimal estimation capabilities, the Kalman Filter, and its different variants have been used in autonomous navigation for several years. Therefore, 3 of the following filtering algorithms are Kalman Filter based.

2.3.1 Kalman Filter

In control theory, Kalman filtering, also known as linear quadratic estimation, is an algorithm that uses a series of measurements observed over time, containing statistical noise and other inaccuracies, and produces estimates of unknown variables that tend to be more accurate than those based on a single measurement alone, by estimating a joint probability distribution over the variables for each time frame. [3]

The kalman filter is an algorithm based on prediction, the first step, where it produces an estimate of the current state as well as its uncertainty and then a subsequent update, the second step.

The current state is calculated using the known system model, input vector as well as the previous state. This step does not include the system's process noise and nonlinearities. After observing the output measurements, which are corrupted with error, both the state and its uncertainty are updated, given more weight to the more certain estimates, this means that the algorithm is recursive and its behaviour is demonstrated, schematically in Figure 2.10.

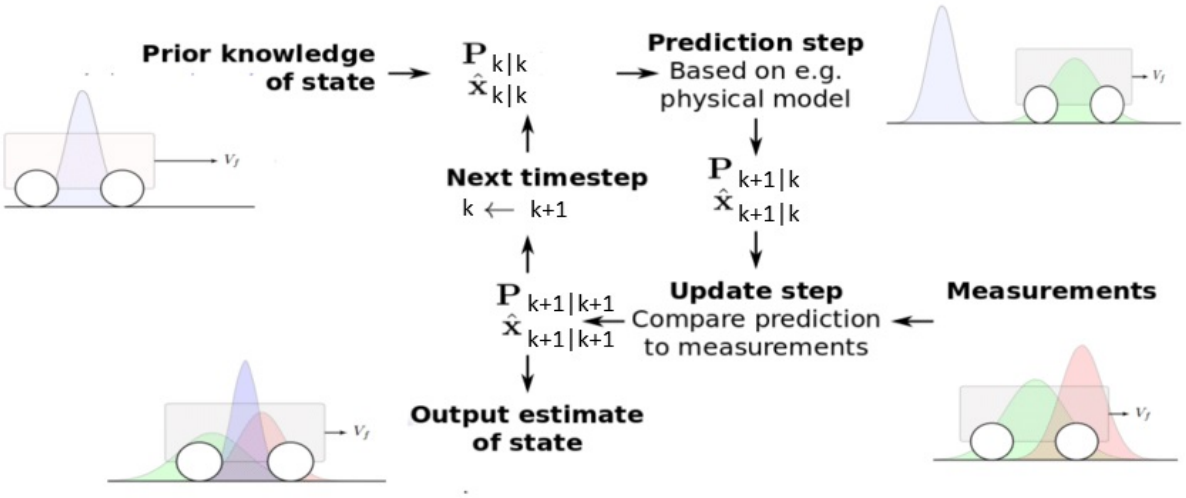

Figure 2.10: Kalman Filter recursive algorithm behaviour [17].

It is assumed that the process noise and sensor's noise, η_x and η_y, are Gaussian with zero mean and its variance can be obtained from experimental data. Also, it's assumed that the dynamic systems

[3] Citation from `https://en.wikipedia.org/wiki/Kalman_filter`

are linear and can be described in a matrix state space representation as follows:

$$\begin{cases} x(k+1) = Ax(k) + Bu(k) + \eta_x(k), & \eta_x(k) \sim N(0, R_w) \\ y(k) = Cx(k) + Du(k) + \eta_y(k), & \eta_y(k) \sim N(0, R_v) \end{cases} \tag{2.26}$$

The variables above are, k, discrete time, x, state vector, y, output vector, u, input (or control) vector, A, state matrix, B, input matrix, C, output matrix, D, the feedforward matrix, η_x and η_y, the process noise and sensor's error, are Gaussian with zero mean and σ_x and σ_y as variance. The system's optimal state

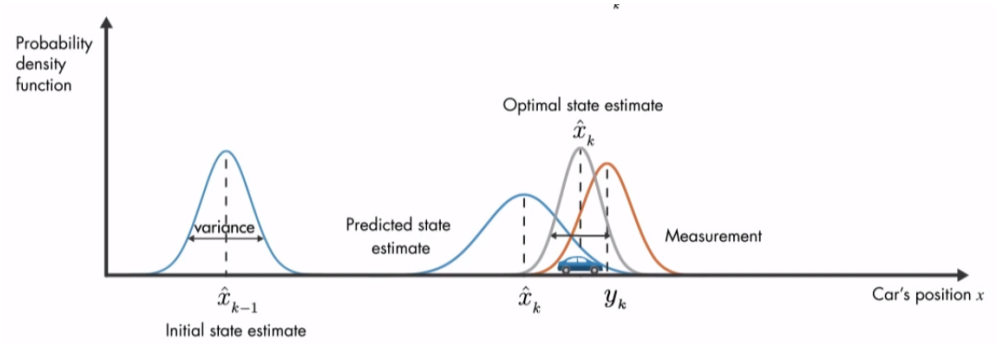

Figure 2.11: Kalman Filter probability density functions schematic [17].

estimate is obtained using the system's predicted state estimate and the measurement with a weighted average. The kalman gain, evaluates which measures have smaller estimated uncertainty and therefore are trusted more. The Measurement Update is defined as:

$$\begin{cases} \hat{x}(k+1|k+1) = \hat{x}(k+1|k) + K(k+1)[y(k+1) - C\hat{x}(k+1|k) - Du(k+1)] \\ P(k+1|k+1) = P(k+1|k) - K(k+1)CP(k+1|k) \end{cases} \tag{2.27}$$

This process is repeated at every time step, with the new predicted state estimate and its covariance being dependent on the previous optimal state estimate. As seen in the following equation:

$$\begin{cases} \hat{x}(k+1|k) = A\hat{x}(k|k) + Bu(k) \\ P(k+1|k) = AP(k|k)A^T + R_w \end{cases} \tag{2.28}$$

The uncertainty of the measurements and of the current state estimate are fundamental to obtain the Kalman Gain, $K(k+1)$. The Kalman Gain is then used to calculate the optimal state estimate, equation 2.27. With a high gain, the filter places more weight on the most recent measurements, and thus follows them more responsively. With a low gain, the filter follows the model predictions more closely.

$$K(k+1) = P(k+1|k)C^T[R_v + CP(k+1|k)C^T]^{-1} \tag{2.29}$$

The state space model shown in each subsequent section is in continuous time, therefore, to implement

the Kalman Filter it must be converted to discrete time. In continuous time the system is defined as:

$$\begin{cases} \dot{x} = Ex(t) + Fu(t) \\ y(t) = Gx(t) + Hu(t) \end{cases} \tag{2.30}$$

The variables above are, t, continuous time, x, state vector, y, output vector, u, input (or control) vector, E, state matrix, F, input matrix, G, output matrix, and H, the feedforward matrix.

To obtain the discrete state space matrices, three MATLAB functions were used. First the sampling time is defined, h, then the $\tau = ss(E, F, G, H)$ function creates a state space model in continuous time, then the $\xi = c2d(\tau, h)$ function converts the previous model to discrete time and the function $[A, B, C, D, Ts] = ssdata(\xi)$ obtains the state space discrete matrices.

2.3.2 Extended Kalman Filter

Previously, the Kalman Filter was described as a linear estimation algorithm. Once the first implementations were a non-linear one, an Extended Kalman Filter (EKF) must be implemented. In the EKF the dynamic systems don't need to be linear but must be differentiable functions. The system is now represented as follows:

$$\begin{cases} x(k+1) = Ax(k) + Bu(k) + \eta_x(k), & \eta_x(k) \sim N(0, R_w) \\ y(k) = h(x_k, u_k) + \eta_y(k), & \eta_y(k) \sim N(0, R_v) \end{cases} \tag{2.31}$$

Must be noted that the Extended Kalman Filter doesn't need a linear function as state function, nevertheless in this thesis all the state functions are linear, and therefore, the EKF is described with state matrices. The variables above are, k, discrete time, x, state vector, y, output vector, u, input (or control) vector, A, state matrix, B, input matrix, $h(x_k, u_k)$, measurement function, η_x and η_y, the process noise and sensor's error, are Gaussian with zero mean and σ_x and σ_y as variance.

The predicted state estimate, $\hat{x}(k+1|k)$, is computed from the previous optimal state estimate, $\hat{x}(k|k)$, as in the Kalman Filter, and $h(x_k, u_k)$ is used to compute the predicted output, $\hat{y}(k+1|k)$, estimate using the predicted state estimate. Once $h(x_k, u_k)$ cannot be in a matrix form, the system must be linearized in each time step so that the Kalman Filter equations can be applied. The linearization around the estimation is done computing, at each time step, the matrix of partial derivatives. The Jacobian is computed with the current predicted state estimation, $J(x(k+1|k), u(k+1)) \equiv J$.

$$J(x_k, u_k) = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \dots & \frac{\partial h_1}{\partial x_i} \\ \dots & \dots & \dots \\ \frac{\partial h_j}{\partial x_1} & \dots & \frac{\partial h_j}{\partial x_i} \end{bmatrix} \tag{2.32}$$

The Measurement Update is defined as:

$$\begin{cases} \hat{x}(k+1|k+1) = \hat{x}(k+1|k) + K(k+1)[y(k+1) - h(\hat{x}(k+1|k), u(k+1))] \\ P(k+1|k+1) = P(k+1|k) - K(k+1)JP(k+1|k) \end{cases} \tag{2.33}$$

This process is repeated at every time step, with the new optimal estimate state and its covariance influencing the prediction of the next state estimation, as follows:

$$\begin{cases} \hat{x}(k+1|k) = A\hat{x}(k|k) + Bu(k) \\ P(k+1|k) = AP(k|k)A^T + R_w \end{cases} \tag{2.34}$$

Once again, the Kalman Gain affects the predicted state estimation. With a high gain, the filter places more weight on the most recent measurements, and thus follows them more responsively. With a low gain, the filter follows the model predictions more closely.

$$K(k+1) = P(k+1|k)J^T[R_v + JP(k+1|k)J^T]^{-1} \tag{2.35}$$

2.3.3 Kalman Filter as Complementary Filtering

In order to understand the usage of a Kalman Filter or Extended Kalman Filter as complementary filtering, first, a brief introduction to complementary filtering must be done.

Sensor fusion can be achieved by several algorithms, one of the less complex to implement is the complementary filter. The filter is defined by two gains that act as high and low pass filter to the sensors' output. This filter is specially important in the orientation estimation as the accelerometer is susceptible to vibrations that need to be filtered. In order to obtain a moving average, that is, a filtered acceleration, a low pass filter is the answer. On the opposite end of the spectrum, the rate gyro is accurate in the short term but due to the bias term the long term results lack accuracy. With this in mind a high pass filter is desired. That way the short-term gyroscope data is used while eliminating long term errors. The

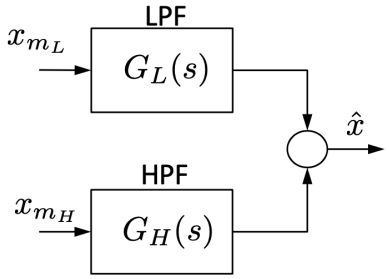

Figure 2.12: Block Diagram of Complementary Filtering [19].

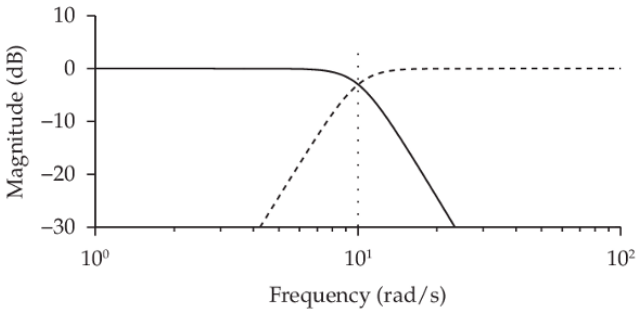

Figure 2.13: Bode Diagram of Complementary Filtering [19].

low pass filter is defined as:

$$G_L(s) = \frac{l}{s+l} \tag{2.36}$$

The high pass filter is defined as:

$$G_H(s) = \frac{s}{s+l} \tag{2.37}$$

And the combination of both, as shown in Figure 2.12:

$$G_H(s) + G_L(s) = 1 \tag{2.38}$$

Kalman Filters can be implemented in order to function as complementary filtering. Figure 2.14 shows a scenario where the inputs are the rate gyro, $\dot{x}_{mH}$, and the magnetometer, x_{mL}. In a complementary filtering scenario or a Kalman Filtering one, the integral of the rate gyro must be calculated before entering the filter. This integration reduces the accuracy of the heading due to the bias.

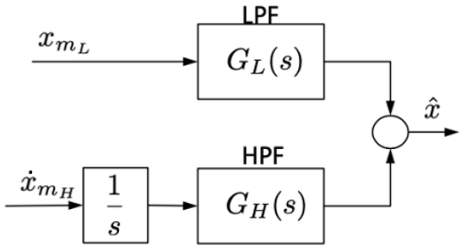

Figure 2.14: Block Diagram of Complementary Filtering with integration [20].

The output of the complementary filtering shown in Figure 2.14 is:

$$\hat{X}(s) = \frac{l}{s+l} X_{mL}(s) + \frac{s}{s+l} \frac{\dot{X}_{mH}(s)}{s} \tag{2.39}$$

The same filtering can be achieved using a Kalman Filter, but for that a few alterations to the implementation in 2.3.1 must be done. Recalling equation (2.30).

$$\begin{cases} \dot{x} = Ex(t) + Fu(t) \\ y(t) = Gx(t) + Hu(t) \\ \dot{\hat{x}} = E\hat{x}(t) + Fu(t) + L(y(t) - G\hat{x}) \end{cases} \tag{2.40}$$

The difference in this application is that instead of having null input and the magnetometer and the integral of the rate gyro as output, the input is the rate gyro and the output is the magnetometer. That is:

$$\begin{cases} u = \dot{x}_{mH} = y_{gyro,z} \\ y = x_{mL} = y_{mag} \end{cases} \tag{2.41}$$

If equation (2.40) is rearranged and the Laplace transform is calculated, something similar to equation (2.39) appears:

$$\hat{Y}(s) = [G(sI - E + LG)^{-1}L]Y(s) + [G(sI - E + LG)^{-1}F]U(s) \tag{2.42}$$

Taking in consideration equation (2.41):

$$\hat{X}(s) = \frac{GL}{sI - E + LG} X_{mL}(s) + \frac{GFs}{sI - E + LG} \frac{\dot{X}_{mH}(s)}{s} \tag{2.43}$$

Equation (2.43) has the same structure as equation (2.39).

2.3.4 Non-linear Filtering

The non-linear filter was designed to obtain the position estimation of drone 3. This filter has the objective to use less sensors than the previous filters, using only the angle sensor and not the combination of the angle sensor and distance sensor. The state space model shown in the previous filters is in continuous time and is defined as:

$$\begin{cases} \dot{x} = Ex(t) + Fu(t) \\ y(t) = h(x,t) \end{cases} \tag{2.44}$$

The variables above are, t, continuous time, x, state vector, y, output vector, u, input (or control) vector, E, state matrix, F and $h(x,t)$ the output function.

From equation (2.40), the Kalman Filter's estimation is:

$$\dot{\hat{x}} = E\hat{x}(t) + Fu(t) + L(y(t) - G\hat{x}) \tag{2.45}$$

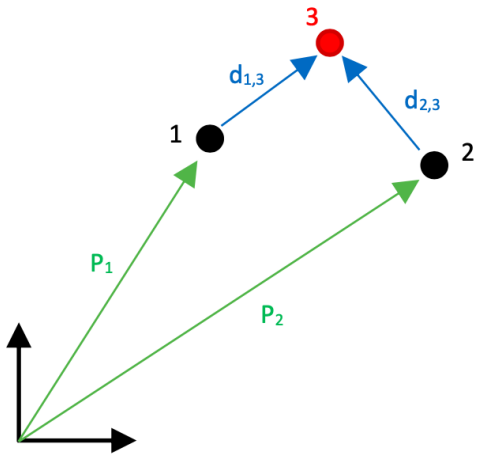

Figure 2.15: Schematics of drones positioning and sensors for non-linear filtering.

For the stability proof of the non-linear filter the states are the position of drone 3, $\vec{p_3}$:

$$x = \vec{p_3} \tag{2.46}$$

The control vector is the velocity of drone 3, $\dot{\vec{p_3}}$:

$$u = \dot{\vec{p_3}} \tag{2.47}$$

The inputs of the filter are the GPS positions of Drone 1 and Drone 2, $\vec{p_1}$ e $\vec{p_2}$, as well as the angle sensor, $\frac{\vec{d_{1,3}}}{||\vec{d_{1,3}}||}$ and $\frac{\vec{d_{2,3}}}{||\vec{d_{2,3}}||}$:

$$h^T = \begin{bmatrix} \vec{p_1} & \vec{p_2} & \frac{\vec{d_{1,3}}}{||\vec{d_{1,3}}||} & \frac{\vec{d_{2,3}}}{||\vec{d_{2,3}}||} \end{bmatrix} \tag{2.48}$$

Must be noted that:

$$\vec{d_{1,3}} = \vec{p_3} - \vec{p_1} \tag{2.49}$$

$$\frac{\vec{d_{1,3}}}{||\vec{d_{1,3}}||} \times (\vec{p_3} - \vec{p_1}) = 0 \tag{2.50}$$

$$\tilde{\vec{p_3}} = \vec{p_3} - \hat{\vec{p_3}} \tag{2.51}$$

Taking in consideration the skew matrix:

$$a \times b = S(a)b \tag{2.52}$$

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \times \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} a_2b_3 - b_2a_3 \\ b_1a_3 - a_1b_3 \\ a_1b_3 - b_1a_3 \end{bmatrix} = S(a) \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \tag{2.53}$$

Using equations (2.49) to (2.52):

$$S(\frac{\vec{d_{1,3}}}{||\vec{d_{1,3}}||})(\hat{\vec{p_3}} - \vec{p_1}) = S(\frac{\vec{d_{1,3}}}{||\vec{d_{1,3}}||})(\hat{\vec{p_3}} - \vec{p_3} + \vec{p_3} - \vec{p_1}) \tag{2.54}$$

$$S(\frac{\vec{d_{1,3}}}{||\vec{d_{1,3}}||})(-\tilde{\vec{p_3}} + \vec{d_{1,3}}) = -S(\frac{\vec{d_{1,3}}}{||\vec{d_{1,3}}||})\tilde{\vec{p_3}} \tag{2.55}$$

$$\hat{\vec{p_3}} - \vec{p_1} = -\tilde{\vec{p_3}} + \vec{d_{1,3}} \tag{2.56}$$

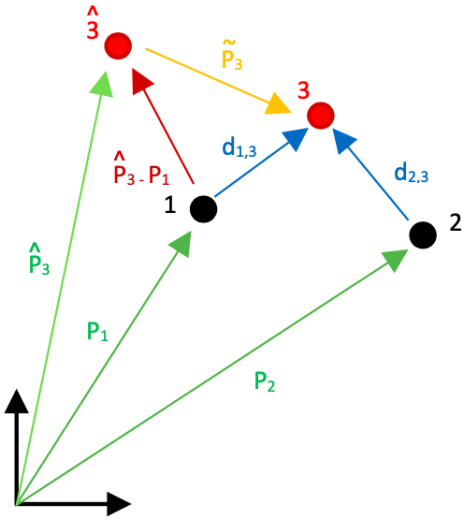

Figure 2.16: Schematics for equations (2.49)-(2.59).

Bearing in mind equations (2.46) and (2.47):

$$\dot{\vec{p_3}} = u \tag{2.57}$$

$$\dot{\hat{\vec{p_3}}} = u + l(\hat{\vec{p_3}} - \vec{p_1}) \tag{2.58}$$

$$\dot{\hat{\vec{p_3}}} = u + lS(\frac{\vec{d_{1,3}}}{||\vec{d_{1,3}}||})^2(\hat{\vec{p_3}} - \vec{p_1}) = u - lS(\frac{\vec{d_{1,3}}}{||\vec{d_{1,3}}||})^2\tilde{\vec{p_3}} \tag{2.59}$$

To check whether or not $\tilde{\vec{p_3}}$ converges asymptotically to zero, consider the error system:

$$\dot{\tilde{\vec{p_3}}} = \dot{\vec{p_3}} - \dot{\hat{\vec{p_3}}} \tag{2.60}$$

$$\dot{\tilde{\vec{p_3}}} = u - u + lS(\frac{\vec{d_{1,3}}}{||\vec{d_{1,3}}||})^2\tilde{\vec{p_3}} \tag{2.61}$$

$$\dot{\tilde{\vec{p_3}}} = lS(\frac{\vec{d_{1,3}}}{||\vec{d_{1,3}}||})^2\tilde{\vec{p_3}} \tag{2.62}$$

If $x = \tilde{\vec{p_3}}$, then equation (2.62) can be rewritten as:

$$\dot{x} = f(x,t) \tag{2.63}$$

To prove that the error tends to zero Lyapunov stability arguments can be invoked, starting with a candidate Lyapunov function that is positive definite:

$$V(x) > 0 \tag{2.64}$$

$$V(x) = \begin{cases} 0, x = 0 \\ > 0, x \neq 0 \end{cases} \tag{2.65}$$

$$V(x) = \frac{1}{2}x^T x \tag{2.66}$$

The positive definiteness of $V(x)$ has been proven. If one can show that the time derivative of $V(x)$ is negative definite along the solutions of the system, then one guarantees that $x = 0$ is asymptotically stable.

$$\dot{V}(x) = \begin{cases} 0, x = 0 \\ < 0, x \neq 0 \end{cases} \tag{2.67}$$

$$\dot{V}(x) = x^T \dot{x} = x^T f(x,t) \tag{2.68}$$

Considering the present case, $\dot{V}(\tilde{\vec{p_3}}) = l\tilde{\vec{p_3}}^T S(\frac{\vec{d_{1,3}}}{||\vec{d_{1,3}}||})^2 \tilde{\vec{p_3}} \leq 0$ which is only negative semi-definite, meaning that convergence to the origin is not guaranteed, the filter needs some adjustments. Therefore:

$$\dot{\tilde{\vec{p_3}}} = u - u + lS(\frac{\vec{d_{1,3}}}{||\vec{d_{1,3}}||})^2 \tilde{\vec{p_3}} + lS(\frac{\vec{d_{2,3}}}{||\vec{d_{2,3}}||})^2 \tilde{\vec{p_3}} \tag{2.69}$$

Considering once again:

$$V(x) = \frac{1}{2} x^T x = \frac{1}{2} \tilde{\vec{p_3}}^2 \tag{2.70}$$

One obtains:

$$\dot{V}(x) = \begin{cases} 0, x = 0 \\ < 0, x \neq 0 \end{cases} \tag{2.71}$$

$$\dot{V}(x) = x^T \dot{x} = x^T f(x,t) = \tilde{\vec{p_3}}^T f(x,t) \tag{2.72}$$

$$\dot{V}(x) = \tilde{\vec{p_3}}^T l \sum_{i=1}^{2} S(\frac{\vec{d_{i,3}}}{||\vec{d_{i,3}}||})^2 \tilde{\vec{p_3}} \tag{2.73}$$

$$= -l\tilde{\vec{p_3}}^T S^T(\frac{\vec{d_{1,3}}}{||\vec{d_{1,3}}||}) S(\frac{\vec{d_{1,3}}}{||\vec{d_{1,3}}||}) \tilde{\vec{p_3}} - l\tilde{\vec{p_3}}^T S^T(\frac{\vec{d_{2,3}}}{||\vec{d_{2,3}}||}) S(\frac{\vec{d_{2,3}}}{||\vec{d_{2,3}}||}) \tilde{\vec{p_3}} \tag{2.74}$$

Note that $S(a)^T = -S(a)$. Therefore, $\dot{V} = -l\tilde{\vec{p_3}}^T \sum_{i=1}^{2} S^T(\frac{\vec{d_{i,3}}}{||\vec{d_{i,3}}||}) S(\frac{\vec{d_{i,3}}}{||\vec{d_{i,3}}||}) \tilde{\vec{p_3}}$ which is negative definite provided that $\vec{d_{1,3}}$ and $\vec{d_{2,3}}$ are not collinear.

When $\dot{V(x)} = 0$ when $x = 0$ and $x \to 0$ when $t \to \infty$, therefore the filter guarantees convergence of $\hat{\vec{p_3}}$ to $\vec{p_3}$ and takes the form:

$$\dot{\hat{\vec{p_3}}} = u + l \sum_{i=1}^{2} S(\frac{\vec{d_{i,3}}}{||\vec{d_{i,3}}||})^2 (\hat{\vec{p_3}} - \vec{p_i}) \tag{2.75}$$

Chapter 3

Implementation

The implementation of this thesis is based on two major parts, the block diagrams and the modeling of the state space.

The implementation has two different approaches. The first one is a decentralised approach. That means that each drone is treated as separated identity and, therefore, has its own block diagram and model. The second approach is a centralised model.

3.1 Block Diagrams

The implementation lays on a few blocks, regardless of being centralised or decentralised. These fundamental blocks are the simulation block, the filtering block and the RMS block.

Must be noted that although these blocks stay the same throughout this thesis, the model changes, and, therefore, the blocks' behaviour changes.

Root Mean Square

The Root Mean Square block is the data gathering block. The root mean square calculation, in estimation, is a measure of the imperfection of the fit of the estimator to the data, and is given by equation (3.1).

$$RMS = \sqrt{\frac{1}{n}[(x_1 - \hat{x_1})^2 + (x_2 - \hat{x_2})^2 + ... + (x_n - \hat{x_n})^2]} \tag{3.1}$$

It has two inputs, the estimation, the Filter's output and the real value, the simulation's output.

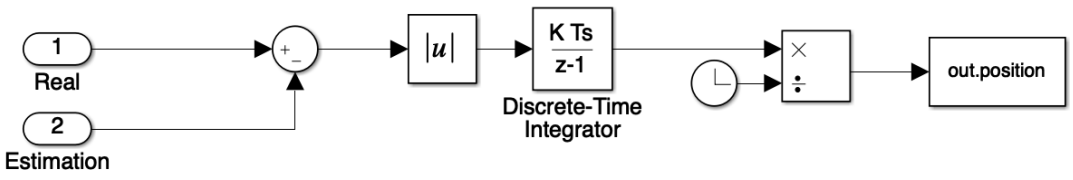

Figure 3.1: Data Gathering block diagram.

Simulation

The simulation subsystem is a block diagram that can be described in state space representation, equation (2.26). This state space representation can describe a centralised or decentralised approach, with linear or non-linear output.

A linear output is represented by equation (3.2).

$$\begin{cases} x(k+1) = Ax(k) + Bu(k) + \eta_x(k), & \eta_x(k) \sim N(0, R_w) \\ y(k) = Cx(k) + Du(k) + \eta_y(k), & \eta_y(k) \sim N(0, R_v) \end{cases} \tag{3.2}$$

If the output is not linear it can be defined as in equation (2.31):

$$\begin{cases} x(k+1) = Ax(k) + Bu(k) + \eta_x(k), & \eta_x(k) \sim N(0, R_w) \\ y(k) = h(x_k, u_k) + \eta_y(k), & \eta_y(k) \sim N(0, R_v) \end{cases} \tag{3.3}$$

The variables above are, k, discrete time, x, state vector, y, output vector, u, input (or control) vector, A, state matrix, B, input matrix, C, output matrix, D, the feedforward matrix, $h(x_k, u_k)$, the measurement function, η_x and η_y, the process noise and sensor's error.

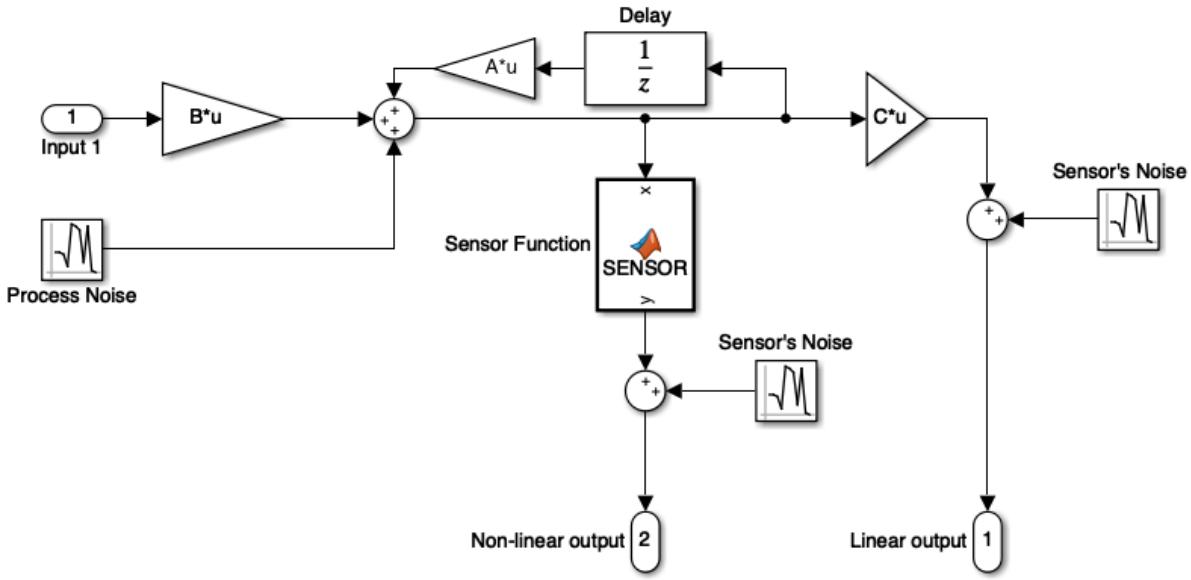

Figure 3.2: Simulation subsystem block diagram.

The subsystem has one input, the control vector, Input 1, and two outputs, the linear sensor, output 1, and the non-linear sensor, output 2.

Should be noted that the simulation can have only linear sensors, only non-linear sensors or a combination of both, depending on the implementation.

The initial conditions of this and all block diagrams are set in the modeling section, 3.2.

Kalman Filter

The block diagram of the Kalman Filter is defined by its equations, shown in section 2.3.1. Retrieving equations (2.27) to (2.29).

The Measurement Update, equation (2.27).

$$\hat{x}(k+1|k+1) = \hat{x}(k+1|k) + K(k+1)[y(k+1) - C\hat{x}(k+1|k) - Du(k+1)] \tag{3.4}$$

$$P(k+1|k+1) = P(k+1|k) - K(k+1)CP(k+1|k) \tag{3.5}$$

The new state prediction and its covariance, equation (2.28).

$$\hat{x}(k+1|k) = A\hat{x}(k|k) + Bu(k) \tag{3.6}$$

$$P(k+1|k) = AP(k|k)A^T + R_w \tag{3.7}$$

The Kalman Gain calculation, equation (2.29).

$$K(k+1) = P(k+1|k)C^T[R_v + CP(k+1|k)C^T]^{-1} \tag{3.8}$$

The Kalman Filter's block diagram, shown below, has Input 1, the control vector, Input 2, the sensors, and the Output 1, the estimation.

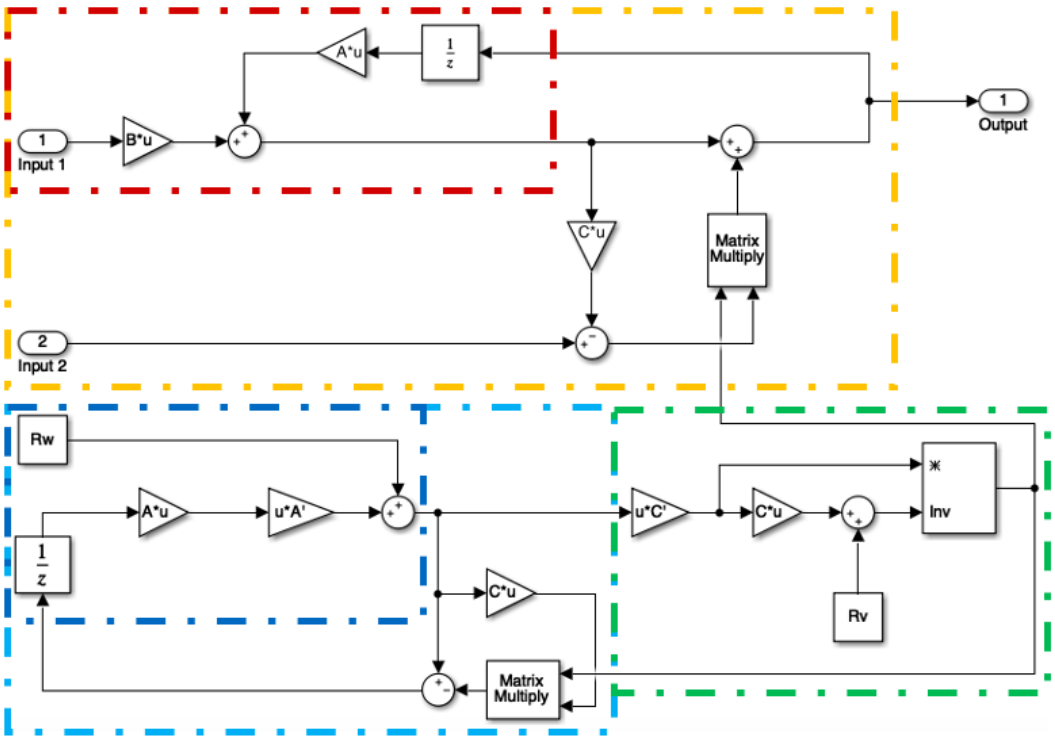

Figure 3.3: Kalman Filter block diagram.

The block diagram represents equations (3.4) to (3.8). Equation (3.4) is in yellow; Equation (3.5) is in light blue; Equation (3.6) is in red; Equation (3.7) is in dark blue; Equation (3.8) is in green;

Extended Kalman Filter

The Extended Kalman Filter's block diagram has a very similar structure to the Kalman Filter's. Retrieving equations (2.33) to (2.35) and bearing in mind that J is the Jacobian, equation (2.32).

The Measurement Update, equation (2.33):

$$\hat{x}(k+1|k+1) = \hat{x}(k+1|k) + K(k+1)[y(k+1) - h(\hat{x}(k+1|k), u(k+1))] \tag{3.9}$$

$$P(k+1|k+1) = P(k+1|k) - K(k+1)JP(k+1|k) \tag{3.10}$$

The new state prediction and its covariance, equation (2.34):

$$\hat{x}(k+1|k) = A\hat{x}(k|k) + Bu(k) \tag{3.11}$$

$$P(k+1|k) = AP(k|k)A^T + R_w \tag{3.12}$$

And, once again, the Kalman Gain, equation (2.35):

$$K(k+1) = P(k+1|k)J^T[R_v + JP(k+1|k)J^T]^{-1} \tag{3.13}$$

The Extended Kalman Filter's block diagram has Input 1, the control vector, Input 2, the sensors' measurements and the Output 1, the state estimation.

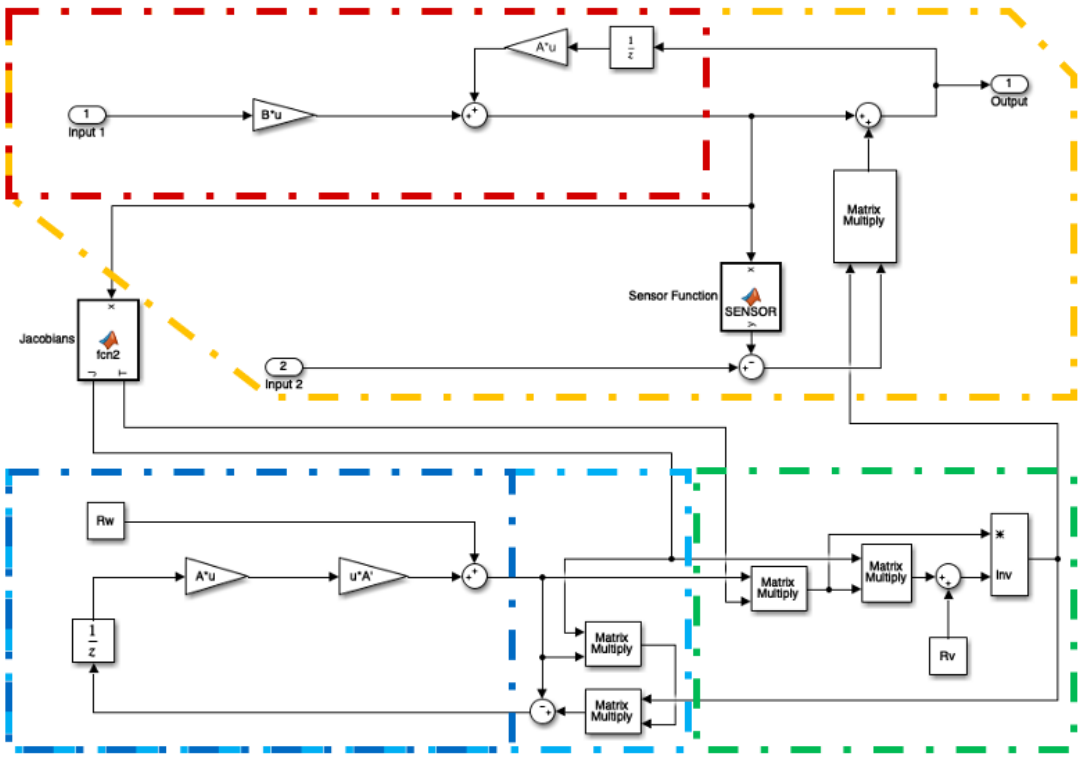

Figure 3.4: Extended kalman Filter block diagram. Equation (3.9) is in yellow; Equation (3.10) is in light blue; Equation (3.11) is in red; Equation (3.12) is in dark blue; Equation (3.13) is in green;

Non-linear Filter

The non-linear filter's block diagram has a very different structure to the Kalman Filter's. The state space model shown in the previous filters is in continuous time and the is defined as:

$$\begin{cases} \dot{x} = Ex(t) + Fu(t) \\ y(t) = h(x,t) \end{cases} \tag{3.14}$$

Retrieving the equation (2.75) and bearing in mind that the states of the filter are now the position and velocity of drone 3, $\vec{p_3}$ and $\dot{\vec{p_3}}$, and the control vector is its acceleration, $u = \vec{a_3}$:

$$x^T = \begin{bmatrix} \vec{p_3} & \dot{\vec{p_3}} \end{bmatrix}, \qquad u^T = \begin{bmatrix} \vec{a_3} \end{bmatrix} \tag{3.15}$$

We got:

$$\dot{\hat{x}} = Ex + Fu + l\sum_{i=1}^{2} S(\frac{\vec{d_{i,3}}}{||\vec{d_{i,3}}||})^2(\hat{\vec{p_3}} - \vec{p_i}) \tag{3.16}$$

Where $\vec{p_i}$ is used the GPS sensor's output.

Remembering equations (2.52) and (2.53):

$$a \times b = S(a)b \tag{3.17}$$

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \times \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} a_2b_3 - b_2a_3 \\ b_1a_3 - a_1b_3 \\ a_1b_3 - b_1a_3 \end{bmatrix} = S(a) \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \tag{3.18}$$

This filter can be implemented by the following block:

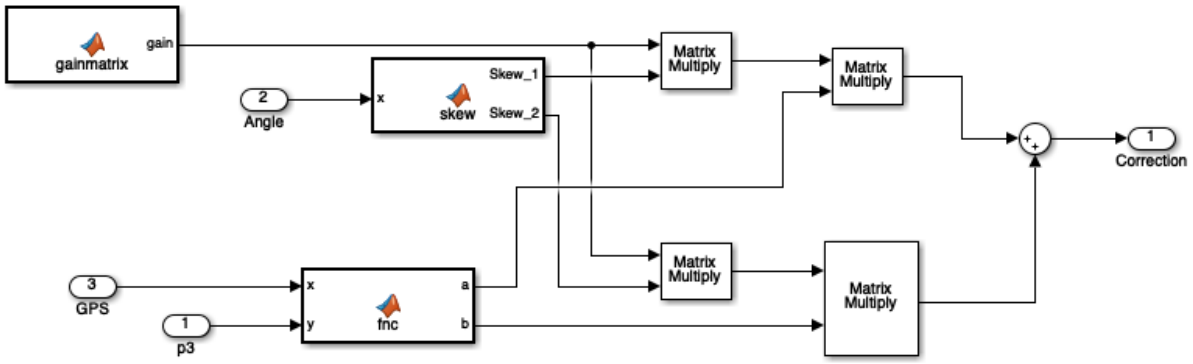

Figure 3.5: Non-linear Filter correction-block diagram, $l\sum_{i=1}^{2} S(\frac{\vec{d_{i,3}}}{||\vec{d_{i,3}}||})^2(\hat{\vec{p_3}} - \vec{p_i})$.

3.1.1 Decentralised

The decentralised approach was the first to be implemented and simulated. This approach considers each drone as an independent identity, and the communication between drones provides each other with sensor information of each drone. This implementation and consequent simulation has seven different model configurations but only three block diagrams. The three block diagrams differ from each other in the number of drones.

The Drone

Throughout the different inter-drone interactions the drone's block diagram is constant, changing its matrices, inputs and outputs.

Each drone subsystem is composed by a drone simulation, a filter and a data gathering block. The subsystem can be represented by the following block diagram:

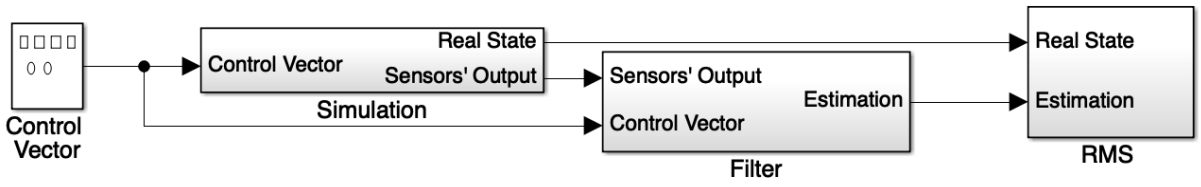

Figure 3.6: Drone's subsystem block diagram.

This subsystem has one input, the the control vector and four outputs. The Sensors' Output, the real state, the state estimation and the error of the estimation from the Data Gathering Block.

One Drone

The first Kalman filter implementation was the fusion of the data gathered from the GPS and the predicted state from the model.

The block diagram implementation of one drone with a GPS receiver is:

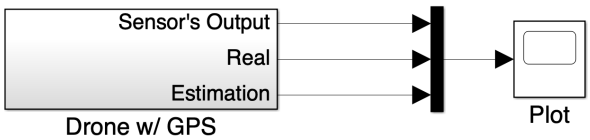

Figure 3.7: Block diagram of one drone simulation.

Must be noted that the control vector is part of the Drone's subsystem, therefore, there is no Input in this block. The outputs are the Sensors' output, the real states and their optimal estimation. The outputs are connected to a scope to obtain a visual representation of the filter's error.

Two Drones

The second Kalman Filter implementation was designed to obtain the position estimation of the Drone without a GPS receiver. For that two drones are needed, the Drone with a GPS receiver, as previously used plus Drone 3.

The block diagram for this simulation is composed as follows:

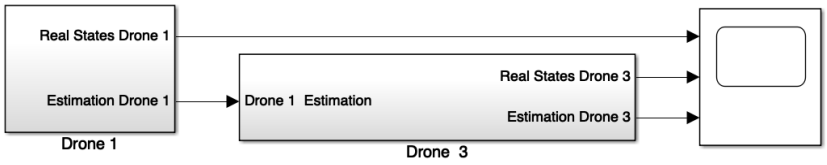

Figure 3.8: Block diagram implementation of Drone 1 and Drone 3.

The subsystem of Drone 1 is the same as previously described. Drone 3, on the other hand, has one input, the position estimation of Drone 1. The interaction between drones will be specified in section 3.2.

Three Drones

The third implementation had as goal the improvement of Drone 3 position estimation. Bearing that in mind, another drone with a GPS receiver was added, Drone 2.

This simulation block diagram is represented in Figure 3.8.

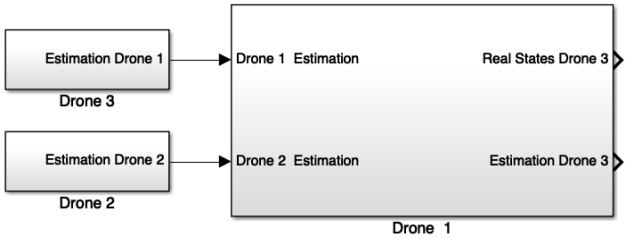

Figure 3.9: Block diagram implementation of Drone 1, Drone 2 and Drone 3.

The subsystem of Drone 1 and Drone 2 are the same and are already described. Drone 3, on the other hand, has one extra input, the position estimation of Drone 1 and Drone 2. The interaction between drones will also be specified in section 3.2.

3.1.2 Centralised

As seen in the previous subsection, the decentralised approach considered each drone as independent identities. As a result of that the optimal state estimation deteriorates, once each drone position estimate is calculated in separate.

From this section on, there is only one simulation subsystem with states containing information on the three drones.

Position

The first centralised simulation has the same goal of the last implementation, to obtain the best estimation of each drone positioning, as well as, choosing the best set of sensors for that goal.

The Simulink block diagram is very similar to the Drone subsystem. It is composed by a simulation, an estimator and a Data Gathering Block, as seen in Figure 3.10.

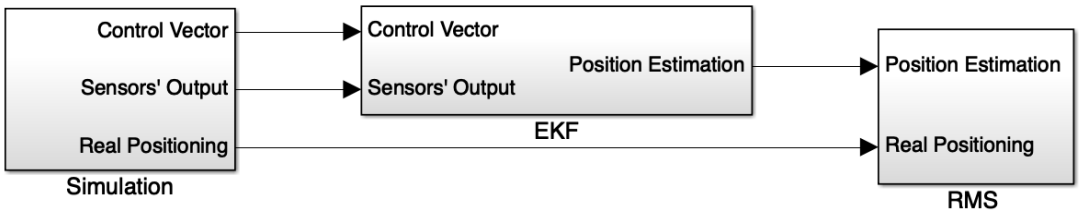

Figure 3.10: Block diagram implementation of centralised positioning estimation.

There are no changes in the subsystems, they are composed as seen in section 3.1.

This system has not an input, the control vector is in the simulation subsystem. It has two outputs, the real position and the estimation of the three drones.

Position and Orientation

Once the positioning estimation of all three drones is developed, the next goal is to implement orientation in the simulation as well. For that, not only the model had to be changed, but the block diagram needed a few adjustments as well.

The Simulink block diagram is very similar to the previous one, Figure 3.10, and is composed by a simulation, an estimator block and RMS as seen below.

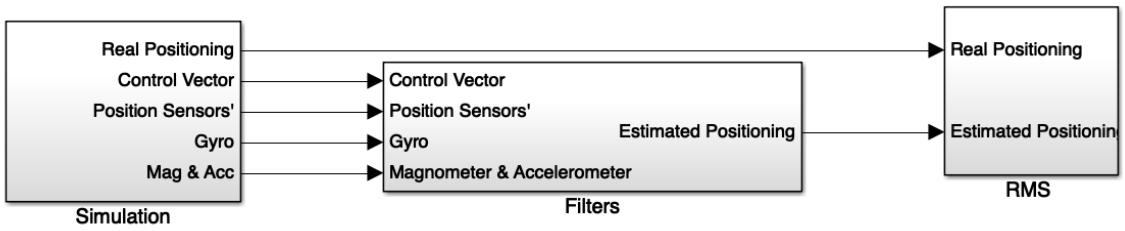

Figure 3.11: Block diagram implementation of centralised positioning and orientation estimation.

The simulation has as output the real positioning of all drones, the control vector (acceleration) and three sets of sensors. The magnetometer, the accelerometer and the rate gyroscope are orientation sensors. The position sensors are the same set as previously used, but now, expressed in the local referential instead of the inertial referential.

In the filters' block, there are two filters, one for the position estimate (Extended Kalman Filter or Nonlinear Filter) and an Extended Kalman Filter used to obtain the estimation of orientation. That estimation is then used to transform the vectors expressed in the local referential to the inertial referential, using the rotational matrix, ${}^{i}R(\psi)$, so that they can be used in the positioning estimation.

The Filters' block is composition is shown in Figure 3.12.

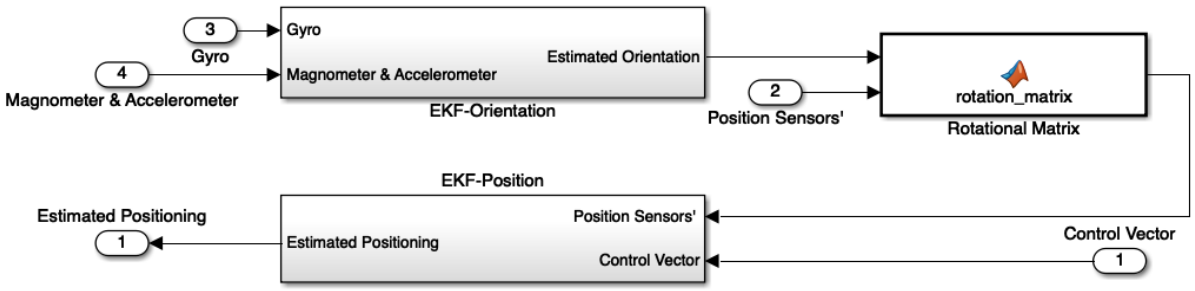

Figure 3.12: Filter's Block Diagram.

The EKF subsystem has already been described as well as its block diagram. Bearing in mind that its inputs are the control vector and the measurements given by the sensors. The output is the drones' position estimation.

3.2 Modeling

As seen in section 3.1, the implementation lays on a few blocks, regardless of being centralised or decentralised. The behaviour of these blocks changes depending on the model implemented. Throughout this section there are fourteen different models.

Each subsystem is defined by its equations, matrices and functions. Hence, each model is defined by its subsystems.

The simulation subsystem can be described in state space representation in discrete time, as in equation (3.2) or equation (3.3).

A linear output is represented by equation (3.14).

$$\begin{cases} x(k+1) = Ax(k) + Bu(k) + \eta_x(k), & \eta_x(k) \sim N(0, R_w) \\ y(k) = Cx(k) + Du(k) + \eta_y(k), & \eta_y(k) \sim N(0, R_v) \end{cases} \tag{3.19}$$

If the output is not linear it can be defined as in equation (3.15):

$$\begin{cases} x(k+1) = Ax(k) + Bu(k) + \eta_w(k), & \eta_w(k) \sim N(0, R_w) \\ y(k) = h(x_k, u_k) + \eta_v(k), & \eta_v(k) \sim N(0, R_v) \end{cases} \tag{3.20}$$

But, for a better intuitive understanding of the model, continuous time is used instead of discrete time.

$$\begin{cases} \dot{x} = Ex + Fu + \eta_w, & \eta_w \sim N(0, R_w) \\ y = Gx + Hu + \eta_v, & \eta_v \sim N(0, R_v) \end{cases} \tag{3.21}$$

As the block diagrams are expressed in discrete time, three MATLAB functions were used to transform the state space representation from continuous to discrete time. Being, h, the sampling time, function $\tau = ss(E, F, G, H)$ creates a state space model in continuous time; function $\xi = c2d(\tau, h)$ converts the previous model to discrete time; and $[A, B, C, D, Ts] = ssdata(\xi)$ obtains the discrete matrices.

Some variables used in the implementation and not yet defined are shown in table 3.1.

Variable	Symbol	Value
Sampling Time	h	$0.005[s]$
Linear Velocity and Acceleration Process Noise	σ^2_{linear}	$2[m/s]\|[ms^{-2}]$
Relative Position Sensor Noise	σ^2_{vector}	$3[m]$
Angle Positioning Sensor Noise	σ^2_{α}	$3[^{o}]$
Angular Velocity Process Noise	σ^2_{ϕ}, σ^2_{θ}, σ^2_{ψ}	$0.001[^{o}s^{-2}]$

Table 3.1: Extra Variables table.

3.2.1 Decentralised

This subsection describes the different approaches tested in the drone's position.

As previously mentioned, the drone's simulation subsystem is a block diagram of the drone's linear dynamic system and can be described in a state space representation in continuous time which is, afterwards, converted to discrete time.

1st Configuration - One Drone

The first Kalman Filter implementation was the fusion of the data gathered from the GPS and the predicted state estimation from the model in 2D. Only one drone was considered, Drone 1.

In continuous time the simulation of the Drone 1 is defined as:

$$\begin{cases} \dot{x} = Ex + Fu + \eta_w, & \eta_w \sim N(0, R_w) \\ y = Gx + Hu + \eta_v, & \eta_v \sim N(0, R_v) \end{cases} \tag{3.22}$$

The states are the position of drone 1, $\vec{p_1}$, with coordinates (x, y) and its linear velocity, $\dot{\vec{p_1}}$. The input is the linear acceleration of drone 1.

$$x^T = \begin{bmatrix} \vec{p_1} & \dot{\vec{p_1}} \end{bmatrix}, \qquad u^T = \begin{bmatrix} \vec{a_1} \end{bmatrix} \tag{3.23}$$

The state space representation is:

$$\dot{x} = \begin{bmatrix} \mathbf{0}_2 & \mathbf{I}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 \end{bmatrix}_{4\times4} x + \begin{bmatrix} \mathbf{0}_2 \\ \mathbf{I}_2 \end{bmatrix}_{4\times2} u \tag{3.24}$$

This configuration output, the GPS, is defined by:

$$y = \begin{bmatrix} \mathbf{I}_2 & \mathbf{0}_2 \end{bmatrix}_{2\times4} x + \begin{bmatrix} \mathbf{0}_2 \end{bmatrix}_{2\times2} u \tag{3.25}$$

The process noise, $\eta_w \sim N(0, R_w)$ and the sensors' error, $\eta_v \sim N(0, R_v)$:

$$R_w = \sigma^2_{linear} \times \mathbf{I}_4, \qquad R_v = \sigma^2_{GPS;x} \times \mathbf{I}_2 \tag{3.26}$$

2nd Configuration - Two Drones

The second Kalman Filter implementation was designed to estimate the positioning of Drone 3, a drone without a GPS receiver. For that implementation at least two drones are needed. Drone 1, a drone with a GPS receiver as shown in the first configuration is the anchor and the sensor in Drone 3 allows it to obtain its relative position to Drone 1.

As seen in the block diagrams' section, Drone 1 and Drone 3 are independent. Therefore, only Drone 3 is defined here as Drone 1 stays the same.

In continuous time the simulation of the Drone 3 is defined as:

$$\begin{cases} \dot{x} = Ex + Fu + \eta_w, & \eta_w \sim N(0, R_w) \\ y = Gx + Hu + \eta_v, & \eta_v \sim N(0, R_v) \end{cases} \tag{3.27}$$

The states are the position of drone 1, $\vec{p_3}$, with coordinates (x, y) and its linear velocity, $\dot{\vec{p_3}}$. The input is the linear acceleration of Drone 3 and the optimal estimation of Drone's 1 positioning.

$$x^T = \left[\vec{p_3} \;\; \dot{\vec{p_3}}\right], \qquad u^T = \left[\vec{a_3} \;\; \hat{\vec{p_1}}\right] \tag{3.28}$$

The state space representation is:

$$\dot{x} = \begin{bmatrix} \mathbf{0}_2 & \mathbf{I}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 \end{bmatrix}_{4\times4} x + \begin{bmatrix} \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{I}_2 & \mathbf{0}_2 \end{bmatrix}_{4\times4} u \tag{3.29}$$

This configuration output is Drone's 3 relative position to Drone 1, ${}^I\vec{d_{1,3}}$, and it is defined by:

$$y = \begin{bmatrix} \mathbf{I}_2 & \mathbf{0}_2 \end{bmatrix}_{2\times4} x + \begin{bmatrix} \mathbf{0}_2 & -\mathbf{I}_2 \end{bmatrix}_{2\times4} u \tag{3.30}$$

The process noise, $\eta_w \sim N(0, R_w)$ and the sensors' error, $\eta_v \sim N(0, R_v)$:

$$R_w = \sigma^2_{linear} \times \mathbf{I}_4, \qquad R_v = \sigma^2_{vector} \times \mathbf{I}_2 \tag{3.31}$$

3rd Configuration - Three Drones

The third drone implementation was designed to test the position estimation improvement of Drone 3. For that a third drone was added. Drone two is a drone with a GPS receiver as well as Drone 1. The two GPS enabled drones were the same as defined previously. Therefore, only drone 3 was considered.

In continuous time the simulation of the Drone 3 is defined as:

$$\begin{cases} \dot{x} = Ex + Fu + \eta_w, & \eta_w \sim N(0, R_w) \\ y = Gx + Hu + \eta_v, & \eta_v \sim N(0, R_v) \end{cases} \tag{3.32}$$

The states are the position of drone 3, $\vec{p_3}$, with coordinates (x, y) and its linear velocity, $\dot{\vec{p_3}}$. The input is the linear acceleration of Drone 3 and the optimal estimation of Drone's 1 and Drone's 2 positioning.

$$x^T = \left[\vec{p_3}\ \dot{\vec{p_3}}\right], \qquad u^T = \left[\vec{a_3}\ \hat{\vec{p_1}}\ \hat{\vec{p_2}}\right] \tag{3.33}$$

The state space representation is:

$$\dot{x} = \begin{bmatrix} \mathbf{0}_2 & \mathbf{I}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 \end{bmatrix}_{4\times4} x + \begin{bmatrix} \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{I}_2 & \mathbf{0}_2 & \mathbf{0}_2 \end{bmatrix}_{4\times6} u \tag{3.34}$$

This configuration output is Drone's 3 relative position to Drone 1 and Drone 2, ${}^I\vec{d_{1,3}}$ and ${}^I\vec{d_{2,3}}$, and it is defined by:

$$y = \begin{bmatrix} \mathbf{I}_2 & \mathbf{0}_2 \\ \mathbf{I}_2 & \mathbf{0}_2 \end{bmatrix}_{4\times4} x + \begin{bmatrix} \mathbf{0}_2 & -\mathbf{I}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & -\mathbf{I}_2 \end{bmatrix}_{4\times6} u \tag{3.35}$$

The process noise, $\eta_w \sim N(0, R_w)$ and the sensors' error, $\eta_v \sim N(0, R_v)$:

$$R_w = \sigma^2_{linear} \times \mathbf{I}_4, \qquad R_v = \sigma^2_{vector} \times \mathbf{I}_4 \tag{3.36}$$

4th Configuration - Three Drones

With the intention of improving the position estimation of all drones, another sensor was introduced. This sensor is in Drone 1 and gives its relative position to Drone 2. Therefore, only one drone was considered, Drone 1.

In continuous time the simulation of the Drone 1 is defined as:

$$\begin{cases} \dot{x} = Ex + Fu + \eta_w, & \eta_w \sim N(0, R_w) \\ y = Gx + Hu + \eta_v, & \eta_v \sim N(0, R_v) \end{cases} \tag{3.37}$$

The states are the position of drone 1, $\vec{p_1}$, with coordinates (x, y) and its linear velocity, $\dot{\vec{p_1}}$. The input

is the linear acceleration of Drone 3 and the optimal estimation of Drone's 1 and Drone's 2 positioning.

$$x^T = \left[\vec{p_1}\ \dot{\vec{p_1}}\right], \qquad u^T = \left[\vec{a_1}\ \hat{\vec{p_2}}\right] \tag{3.38}$$

The state space representation is:

$$\dot{x} = \begin{bmatrix} \mathbf{0}_2 & \mathbf{I}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 \end{bmatrix}_{4\times4} x + \begin{bmatrix} \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{I}_2 & \mathbf{0}_2 \end{bmatrix}_{4\times4} u \tag{3.39}$$

This configuration output is Drone's 1 relative position to Drone 2, ${}^I\vec{d_{1,2}}$, and the GPS coordinates of Drone 1, $\vec{p_1}$.

$$y = \begin{bmatrix} \mathbf{I}_2 & \mathbf{0}_2 \\ \mathbf{I}_2 & \mathbf{0}_2 \end{bmatrix}_{4\times4} x + \begin{bmatrix} \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & -\mathbf{I}_2 \end{bmatrix}_{4\times4} u \tag{3.40}$$

The process noise, $\eta_w \sim N(0, R_w)$ and the sensors' error, $\eta_v \sim N(0, R_v)$:

$$R_w = \sigma^2_{linear} \times \mathbf{I}_4, \qquad R_v = \begin{bmatrix} \sigma^2_{GPS;x} & 0 & 0 & 0 \\ 0 & \sigma^2_{GPS;y} & 0 & 0 \\ 0 & 0 & \sigma^2_{vector} & \\ 0 & 0 & 0 & \sigma^2_{vector} \end{bmatrix} \tag{3.41}$$

5th, 6th and 7th Configurations - Three Drones

From the first to the fourth configuration only Kalman Filters were used, once the outputs were linear. However, that is a simplification. Bearing this in mind, the sensor's output had to be changed. The two alternatives are distance sensors and angular position sensors.

The state space is the same from the fifth to the seventh configuration, with the exception of the output function, $h_j, \in [5, 6, 7]$. So, the three configurations are specified together. Only the changes in Drone 3 are exhibited here.

In continuous time the simulation of the Drone 3 is defined, now, as:

$$\begin{cases} \dot{x} = Ex + Fu + \eta_w, & \eta_w \sim N(0, R_w) \\ y = h(x, u) + \eta_v, & \eta_v \sim N(0, R_v) \end{cases} \tag{3.42}$$

The states stay the same, and are the position of drone 3, $\vec{p_3}$, with coordinates (x, y) and its linear velocity, $\dot{\vec{p_3}}$. The input is the linear acceleration of Drone 3 and the optimal estimation of Drone's 1 and Drone's 2 positioning.

$$x^T = \left[\vec{p_3}\ \dot{\vec{p_3}}\right], \qquad u^T = \left[\vec{a_3}\ \hat{\vec{p_1}}\ \hat{\vec{p_2}}\right] \tag{3.43}$$

The state space representation is:

$$\dot{x} = \begin{bmatrix} \mathbf{0}_2 & \mathbf{I}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 \end{bmatrix}_{4\times4} x + \begin{bmatrix} \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{I}_2 & \mathbf{0}_2 \\ \mathbf{I}_2 & \mathbf{0}_2 \end{bmatrix}_{6\times4} u \tag{3.44}$$

The first output function is based on a distance sensor. The measurement function, h_5, is defined as:

$$h_5 = \begin{bmatrix} ||\vec{d_{1,3}}|| \\ ||\vec{d_{3,2}}|| \end{bmatrix}_{2\times1} = \begin{bmatrix} \sqrt{(p_{3x}-p_{1x})^2+(p_{3y}-p_{1y})^2} \\ \sqrt{(p_{3x}-p_{2x})^2+(p_{3y}-p_{2y})^2} \end{bmatrix}_{2\times1} \tag{3.45}$$

And its Jacobian is:

$$J_5 = \begin{bmatrix} \frac{p_{3x}-p_{1x}}{\sqrt{(p_{3x}-p_{1x})^2+(p_{3y}-p_{1y})^2}} & \frac{p_{3y}-p_{1y}}{\sqrt{(p_{3x}-p_{1x})^2+(p_{3y}-p_{1y})^2}} & \mathbf{0}_2 \\ \frac{p_{3x}-p_{2x}}{\sqrt{(p_{3x}-p_{2x})^2+(p_{3y}-p_{2y})^2}} & \frac{p_{3y}-p_{2y}}{\sqrt{(p_{3x}-p_{2x})^2+(p_{3y}-p_{2y})^2}} & \mathbf{0}_2 \end{bmatrix}_{2\times4} \tag{3.46}$$

The second output function is an angle based sensor. This sensor's output is the angle between Drone 3 and 1, $\alpha_{3,1}$ as well as between Drone 3 and 2, $\alpha_{3,2}$.

$$h_6 = \begin{bmatrix} \alpha_{3,1} \\ \alpha_{3,2} \end{bmatrix}_{2\times1} = \begin{bmatrix} cos^{-1}\frac{p_{1y}-p_{3y}}{\sqrt{(p_{3x}-p_{1x})^2+(p_{3y}-p_{1y})^2}} \\ cos^{-1}\frac{p_{2y}-p_{3y}}{\sqrt{(p_{3x}-p_{2x})^2+(p_{3y}-p_{2y})^2}} \end{bmatrix}_{2\times1} \tag{3.47}$$

Its Jacobian is:

$$J_6 = \begin{bmatrix} \frac{(p_{3x}-p_{1x})(p_{1y}-p_{3y})}{\sqrt{(p_{3x}-p_{1x})^2}[(p_{3x}-p_{1x})^2+(p_{3y}-p_{1y})^2]} & \frac{(p_{3x}-p_{1x})^2}{\sqrt{(p_{3x}-p_{1x})^2}[(p_{3x}-p_{1x})^2+(p_{3y}-p_{1y})^2]} & \mathbf{0}_2 \\ \frac{(p_{3x}-p_{2x})(p_{2y}-p_{3y})}{\sqrt{(p_{3x}-p_{2x})^2}[(p_{3x}-p_{2x})^2+(p_{3y}-p_{2y})^2]} & \frac{(p_{3x}-p_{2x})^2}{\sqrt{(p_{3x}-p_{2x})^2}[(p_{3x}-p_{2x})^2+(p_{3y}-p_{2y})^2]} & \mathbf{0}_2 \end{bmatrix}_{2\times4} \tag{3.48}$$

The third output function is an angle plus distance based sensor. This sensor's output is the angle between Drone 3 and 1, $\alpha_{3,1}$ as well as between Drone 3 and 2, $\alpha_{3,2}$ and the distance between Drone 1 and 3, $||\vec{d_{1,3}}||$, and Drone 2 and 3, $||\vec{d_{3,2}}||$.

$$h_7 = \begin{bmatrix} \alpha_{3,1} \\ \alpha_{3,2} \\ ||\vec{d_{1,3}}|| \\ ||\vec{d_{3,2}}|| \end{bmatrix}_{4\times1} = \begin{bmatrix} cos^{-1}\frac{p_{1y}-p_{3y}}{\sqrt{(p_{3x}-p_{1x})^2+(p_{3y}-p_{1y})^2}} \\ cos^{-1}\frac{p_{2y}-p_{3y}}{\sqrt{(p_{3x}-p_{2x})^2+(p_{3y}-p_{2y})^2}} \\ \sqrt{(p_{3x}-p_{1x})^2+(p_{3y}-p_{1y})^2} \\ \sqrt{(p_{3x}-p_{2x})^2+(p_{3y}-p_{2y})^2} \end{bmatrix}_{4\times1} \tag{3.49}$$

Its Jacobian is:

$$J_7 = \begin{bmatrix} \frac{(p_{3x}-p_{1x})(p_{1y}-p_{3y})}{\sqrt{(p_{3x}-p_{1x})^2}[(p_{3x}-p_{1x})^2+(p_{3y}-p_{1y})^2]} & \frac{(p_{3x}-p_{1x})^2}{\sqrt{(p_{3x}-p_{1x})^2}[(p_{3x}-p_{1x})^2+(p_{3y}-p_{1y})^2]} & \mathbf{0}_2 \\ \frac{(p_{3x}-p_{2x})(p_{2y}-p_{3y})}{\sqrt{(p_{3x}-p_{2x})^2}[(p_{3x}-p_{2x})^2+(p_{3y}-p_{2y})^2]} & \frac{(p_{3x}-p_{2x})^2}{\sqrt{(p_{3x}-p_{2x})^2}[(p_{3x}-p_{2x})^2+(p_{3y}-p_{2y})^2]} & \mathbf{0}_2 \\ \frac{p_{3x}-p_{1x}}{\sqrt{(p_{3x}-p_{1x})^2+(p_{3y}-p_{1y})^2}} & \frac{p_{3y}-p_{1y}}{\sqrt{(p_{3x}-p_{1x})^2+(p_{3y}-p_{1y})^2}} & \mathbf{0}_2 \\ \frac{p_{3x}-p_{2x}}{\sqrt{(p_{3x}-p_{2x})^2+(p_{3y}-p_{2y})^2}} & \frac{p_{3y}-p_{2y}}{\sqrt{(p_{3x}-p_{2x})^2+(p_{3y}-p_{2y})^2}} & \mathbf{0}_2 \end{bmatrix}_{2\times 4} \tag{3.50}$$

The process noise is the same in all configurations, $\eta_w \sim N(0, R_w)$ and the sensors' error is a function of the configuration, $\eta_v \sim N(0, R_v)$:

$$R_w = \sigma^2_{linear} \times \mathbf{I}_4, \qquad {}^5R_v = \sigma^2_{distance} \times \mathbf{I}_4 \tag{3.51}$$

$${}^6R_v = \sigma^2_{\alpha} \times \mathbf{I}_4, \qquad {}^7R_v = \begin{bmatrix} \sigma^2_{\alpha} & 0 & 0 & 0 \\ 0 & \sigma^2_{\alpha} & 0 & 0 \\ 0 & 0 & \sigma^2_{distance} & 0 \\ 0 & 0 & 0 & \sigma^2_{distance} \end{bmatrix} \tag{3.52}$$

3.2.2 Centralised

The decentralised approach considered each drone as independent identities, but from now on the three drones are in the same simulation.

Another major difference is that the decentralised section has no orientation and was defined only in two dimensions, (x, y). The eighth configuration is the last one being two dimensional and with no orientation. The eleventh configuration introduces 2-D orientation. The twelfth is the first three dimensional one the next one is a complete simulation.

8th, 9th and 10th Configurations

The drone's simulation subsystem is a block diagram of the drone's dynamic system and can be described as follows:

$$\begin{cases} \dot{x} = Ex + Fu + \eta_w, & \eta_w \sim N(0, R_w) \\ y = h(x, u) + \eta_v, & \eta_v \sim N(0, R_v) \end{cases} \tag{3.53}$$

In this iteration the states are the position of Drone 1, $\vec{p_1}$, the the relative position of drone 3 to Drone 1, $\vec{d_{1,3}}$, as well as Drone 2 to Drone 3, $\vec{d_{3,2}}$ and the respective velocities.

In continuous time the simulation is defined as follows:

$$x^T = \begin{bmatrix} \vec{p_1} & \vec{d_{1,3}} & \vec{d_{3,2}} & \dot{\vec{p_1}} & \dot{\vec{d_{1,3}}} & \dot{\vec{d_{3,2}}} \end{bmatrix} \tag{3.54}$$

The inputs are, as before, the linear accelerations.

$$u^T = \begin{bmatrix} \vec{a_1} & \vec{a_2} & \vec{a_3} \end{bmatrix} \tag{3.55}$$

The state space matrices are:

$$E = \begin{bmatrix} \mathbf{0}_6 & I_6 \\ \mathbf{0}_6 & \mathbf{0}_6 \end{bmatrix}_{12\times 12} \qquad F = \begin{bmatrix} & \mathbf{0}_6 & \\ I_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ -I_2 & \mathbf{0}_2 & I_2 \\ \mathbf{0}_2 & I_2 & -I_2 \end{bmatrix}_{12\times 6} \tag{3.56}$$

In this iteration, the output function is also based on GPS, distance and angle sensors. Must be taken in consideration that the angle sensor is now the normalised relative position vectors.

The correlation between $\vec{d_{1,2}}$ and the states is:

$$\begin{bmatrix} \vec{p_2} \\ \vec{d_{1,2}} \end{bmatrix} = \begin{bmatrix} \vec{p_1} + \vec{d_{1,3}} + \vec{d_{3,2}} \\ \vec{d_{1,3}} + \vec{d_{3,2}} \end{bmatrix} \tag{3.57}$$

The measurement functions are:

$$h_8^T = \begin{bmatrix} \vec{p_1} & \vec{p_2} & ||\vec{d_{1,3}}|| & ||\vec{d_{3,2}}|| & ||\vec{d_{1,2}}|| \end{bmatrix} \tag{3.58}$$

$$h_9^T = \begin{bmatrix} \vec{p_1} & \vec{p_2} & \frac{\vec{d_{1,3}}}{||\vec{d_{1,3}}||} & \frac{\vec{d_{3,2}}}{||\vec{d_{3,2}}||} & \frac{\vec{d_{1,2}}}{||\vec{d_{1,2}}||} \end{bmatrix} \tag{3.59}$$

$$h_{10}^T = \begin{bmatrix} \vec{p_1} & \vec{p_2} & ||\vec{d_{1,3}}|| & ||\vec{d_{3,2}}|| & ||\vec{d_{1,2}}|| & \frac{\vec{d_{1,3}}}{||\vec{d_{1,3}}||} & \frac{\vec{d_{3,2}}}{||\vec{d_{3,2}}||} & \frac{\vec{d_{1,2}}}{||\vec{d_{1,2}}||} \end{bmatrix} \tag{3.60}$$

The Jacobian of the GPS sensor is:

$$J_{GPS} = \begin{bmatrix} I_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_{2\times 6} \\ I_2 & I_2 & I_2 & \mathbf{0}_{2\times 6} \end{bmatrix}_{4\times 12} \tag{3.61}$$

The Jacobian of the normalised distance vectors is:

$$J_{\frac{\vec{d_{i,j}}}{||\vec{d_{i,j}}||}} = \begin{bmatrix} \mathbf{0}_2 & \begin{bmatrix} a & b \\ b & c \end{bmatrix} & \mathbf{0}_{2\times 8} \\ \mathbf{0}_{2\times 4} & \begin{bmatrix} d & e \\ e & f \end{bmatrix} & \mathbf{0}_{2\times 6} \\ \mathbf{0}_2 & \begin{bmatrix} g & h & g & h \\ h & k & h & k \end{bmatrix} & \mathbf{0}_{2\times 6} \end{bmatrix}_{6\times 12} \tag{3.62}$$

The Jacobian of the distance sensor is:

$$J_{\vec{d_{i,j}}} = \begin{bmatrix} \mathbf{0}_{3\times 2} & \begin{bmatrix} l & m & 0 & 0 \\ 0 & 0 & n & p \\ q & r & q & r \end{bmatrix} & \mathbf{0}_{3\times 6} \end{bmatrix}_{3\times 12} \tag{3.63}$$

Bearing in mind equation (3.52), that is: $\vec{d_{2,1}} = \vec{d_{1,3}} + \vec{d_{3,2}}$, we got:

$$a = \frac{d^2_{(1,3)y}}{||\vec{d_{1,3}}||^3},\ b = \frac{-d_{(1,3)x}d_{(1,3)y}}{||\vec{d_{1,3}}||^3},\ c = \frac{d^2_{(1,3)x}}{||\vec{d_{1,3}}||^3} \tag{3.64}$$

$$d = \frac{d^2_{(3,2)y}}{||\vec{d_{3,2}}||^3},\ e = \frac{-d_{(3,2)x}d_{(3,2)y}}{||\vec{d_{3,2}}||^3},\ f = \frac{d^2_{(3,2)x}}{||\vec{d_{3,2}}||^3} \tag{3.65}$$

$$g = \frac{d^2_{(1,2)y}}{||\vec{d_{1,2}}||^3},\ h = \frac{-d_{(1,2)x}d_{(1,2)y}}{||\vec{d_{1,2}}||^3},\ k = \frac{d^2_{(1,2)x}}{||\vec{d_{1,2}}||^3} \tag{3.66}$$

$$l = \frac{d_{(1,3)x}}{||\vec{d_{1,3}}||},\ m = \frac{d_{(1,3)y}}{||\vec{d_{1,3}}||},\ n = \frac{d_{(3,2)x}}{||\vec{d_{3,2}}||} \tag{3.67}$$

$$p = \frac{d_{(3,2)y}}{||\vec{d_{3,2}}||},\ q = \frac{d_{(1,2)x}}{||\vec{d_{1,2}}||},\ r = \frac{d_{(1,2)y}}{||\vec{d_{1,2}}||} \tag{3.68}$$

The process noise is the same in all configurations, $\eta_w \sim N(0, R_w)$ and the sensors' error is a function of the configuration, $\eta_v \sim N(0, R_v)$. Being,

$$R_w = \sigma^2_{linear} \times \mathbf{I}_{12}, \quad R_{GPS} = \sigma^2_{GPS,x} \times \mathbf{I}_4, \quad R_{distance} = \sigma^2_{distance} \times \mathbf{I}_3, \quad R_{angle} = \sigma^2_{angle} \times \mathbf{I}_6 \tag{3.69}$$

$${}^8R_v = \begin{bmatrix} R_{GPS} & \mathbf{0}_{4\times 3} \\ \mathbf{0}_{3\times 4} & R_{distance} \end{bmatrix}_{7\times 7} \quad {}^9R_v = \begin{bmatrix} R_{GPS} & \mathbf{0}_6 \\ \mathbf{0}_6 & R_{angle} \end{bmatrix}_{10\times 10} \quad {}^{10}R_v = \begin{bmatrix} R_{GPS} & \mathbf{0}_4 & \mathbf{0}_{4\times 5} \\ \mathbf{0}_{3\times 4} & R_{distance} & \mathbf{0}_{3\times 6} \\ \mathbf{0}_6 & \mathbf{0}_{6\times 1} & R_{angle} \end{bmatrix}_{13\times 13} \tag{3.70}$$

Kalman Filter as Complementary Filtering

The Kalman Filter has already been introduced in subsection 2.3. As well as the Extended Kalman Filter and Kalman Filter as complementary Filtering.

In order to choose which implementation shall be used in the orientation, a practical comparison between the two must be done.

The simulation is equal for both implementations and is defined by:

$$\begin{cases} \dot{x} = Ex + Fu + \eta_w, & \eta_w \sim N(0, R_w) \\ y = Gx + Hu + \eta_v, & \eta_v \sim N(0, R_v) \end{cases} \tag{3.71}$$

Where the states are the heading and its rate of change:

$$x^T = \begin{bmatrix} \psi & r \end{bmatrix} \tag{3.72}$$

Once the orientation is a result of process error, η_ψ, the only input is the Bias, $u = \beta_{gyro,z}$.

$$E = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} F = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \eta_w = \begin{bmatrix} 0 \\ \eta_\psi \end{bmatrix} \tag{3.73}$$

The output is the magnetometer and the rate gyro:

$$G = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} H = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \eta_v = \begin{bmatrix} \eta_{mag} \\ \eta_{gyro} \end{bmatrix} \tag{3.74}$$

The first implementation of the Kalman Filter is similar to the previously one used.

Using the previous representation, the Kalman Filter is defined by heading, bias and heading rate of change as states.

$$x^T = \begin{bmatrix} \psi & \beta_{gyro,z} & r \end{bmatrix} \tag{3.75}$$

Once the orientation is a result of process error, η_ψ, there is no input.

$$E = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} F = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \eta_w = \begin{bmatrix} 0 \\ 0 \\ \eta_\psi \end{bmatrix} \tag{3.76}$$

The output is the magnetometer and the rate gyro:

$$G = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix} H = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \eta_v = \begin{bmatrix} \eta_{mag} \\ \eta_{gyro} \end{bmatrix} \tag{3.77}$$

The second implementation of the Kalman Filter is as complementary filtering.

Using the previous representation, the filter is defined by the heading and bias as the only states.

$$x^T = \begin{bmatrix} \psi & \beta_{gyro,z} \end{bmatrix} \tag{3.78}$$

If equation (2.10), the gyro equation is rearranged and equation (2.12) is also used, results:

$$r = y_{gyro,z} - \beta_{gyro,z} - \eta_{gyro,z} \tag{3.79}$$

$$\dot{\beta}_{gyro,z} = 0 \tag{3.80}$$

The input of the filter is the rate gyro:

$$u = y_{gyro,z} \tag{3.81}$$

And, therefore, using equation (3.79), the filter's matrices are:

$$E = \begin{bmatrix} 0 & -1 \\ 0 & 0 \end{bmatrix} F = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \eta_x = \begin{bmatrix} -\eta_{gyro} \\ 0 \end{bmatrix} \tag{3.82}$$

The output is only the magnetometer:

$$G = \begin{bmatrix} 1 & 0 \end{bmatrix} H = \begin{bmatrix} 0 \end{bmatrix} \eta_y = \begin{bmatrix} \eta_{mag} \end{bmatrix} \tag{3.83}$$

11th Configuration - 2D Orientation

Until now, every simulation had, by default, null yaw. That means that the orientation of each drone would not change in time, but that is not a realistic approach, rather an intermediate step.

From this moment on, each local referential, $\{V_i\}$, $i \in [1, 2, 3]$, sympathetic with the drone's kinematic, is not only defined by its position, ${}^I p_i$, $i \in [1, 2, 3]$, but also, its heading, ψ_i, $i \in [1, 2, 3]$, in an inertial referential, $\{I\}$.

A vector defined at any local referential, $\{V_i\}$, can be transformed to the inertial referential, $\{I\}$, employing sequentially the three rotations, each one given by its rotation matrix, but for now, only the 2D yaw matrix is considered.

From equation (2.8), the Yaw rotation matrix of the ith in 2D aircraft is defined as:

$${}^i R(\psi) = \begin{bmatrix} cos(\psi) & -sin(\psi) \\ sin(\psi) & cos(\psi) \end{bmatrix} \tag{3.84}$$

And the inverse matrix in 2D:

$$^{I}R(\psi) = \begin{bmatrix} cos(\psi) & sin(\psi) \\ -sin(\psi) & cos(\psi) \end{bmatrix} \tag{3.85}$$

Any vector defined at any local referential can be transformed to the inertial referential, and is represented as:

$$^{i}R_{i}^{V_i} p_j =^{I} p_j, \qquad i,j \in [1,2,3] \tag{3.86}$$

A vector defined at the inertial referential can be transformed to any local referential, and is represented as:

$$^{I}R_{i}^{I} p_j =^{V_i} p_j, \qquad i,j \in [1,2,3] \tag{3.87}$$

As seen in the block diagrams of centralised position and orientation, in subsection 3.1.2, two filters are needed to obtain the optimal prediction of orientation and positioning. Therefore, for the first time, the simulation equations are not the same as the equations that define the filters.

Taking this in consideration, three state space representations are presented.

The first one is the drone's simulation subsystem and it is represented as follows:

$$SIMULATION = \begin{cases} \dot{x} = Ex + Fu + \eta_w, & \eta_w \sim N(0, R_w) \\ y = h(x,u) + \eta_v, & \eta_v \sim N(0, R_v) \end{cases} \tag{3.88}$$

In this iteration the simulation states are the position of drone 1, $\vec{p_1}$, the relative position of drone 1 and drone 3, $\vec{d_{1,3}}$, as well as drone 3 and drone 2, $\vec{d_{3,2}}$ and the respective velocities, as previously, plus the heading ψ_i, with $i \in [1,2,3]$

$$x^T = \begin{bmatrix} \vec{p_1} & \vec{d_{1,3}} & \vec{d_{3,2}} & \psi_1 & \psi_2 & \psi_3 & \dot{\vec{p_1}} & \dot{\vec{d_{1,3}}} & \dot{\vec{d_{3,2}}} & r_1 & r_2 & r_3 \end{bmatrix} \tag{3.89}$$

The simulation inputs are, as before, the linear accelerations.

$$u^T = \begin{bmatrix} \vec{a_1} & \vec{a_2} & \vec{a_3} \end{bmatrix} \tag{3.90}$$

The simulation state space matrices are:

$$E = \begin{bmatrix} \mathbf{0}_9 & I_9 \\ \mathbf{0}_9 & \mathbf{0}_9 \end{bmatrix}_{18\times 18} \qquad F = \begin{bmatrix} & \mathbf{0}_6 & \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ I_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ -I_2 & \mathbf{0}_2 & I_2 \\ \mathbf{0}_2 & I_2 & -I_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \end{bmatrix}_{18\times 6} \tag{3.91}$$

The measurement function of the simulation is based on GPS, distance and angle sensor, rate gyro and magnetometer. It should be noted that the output of the angle sensor is no longer in the inertial

referential but, instead, in the local referential, equation (3.87), as the states are all expressed in the inertial referential.

$$h_{11}^T = \left[\vec{p_1}\ \vec{p_2}\ ||\vec{d_{1,3}}||\ ||\vec{d_{3,2}}||\ ||\vec{d_{1,2}}||\ \frac{\vec{d_{1,3}}}{||\vec{d_{1,3}}||}\ \frac{\vec{d_{3,2}}}{||\vec{d_{3,2}}||}\ \frac{\vec{d_{1,2}}}{||\vec{d_{1,2}}||}\ r_1\ r_2\ r_3\ \psi_1\ \psi_2\ \psi_3\right] \tag{3.92}$$

The process noise, $\eta_w \sim N(0, R_w)$ and the sensors' error, $\eta_v \sim N(0, R_v)$ are defined as:

$$R_{linear} = \sigma_{linear}^2 \times \mathbf{I}_6, \qquad R_\psi = \sigma_\psi^2 \times \mathbf{I}_3 \tag{3.93}$$

$$R_w = \begin{bmatrix} R_{linear} & \mathbf{0}_6 & \mathbf{0}_6 \\ \mathbf{0}_{3\times6} & \mathbf{0}_{3\times6} & \mathbf{0}_{3\times6} \\ \mathbf{0}_{6\times9} & R_{linear} & \mathbf{0}_{6\times3} \\ \mathbf{0}_{3\times9} & \mathbf{0}_{3\times6} & R_\psi \end{bmatrix}_{18\times18} \tag{3.94}$$

$$R_{GPS} = \sigma_{GPS,x}^2 \times \mathbf{I}_4, \qquad R_{distance} = \sigma_{distance}^2 \times \mathbf{I}_3 \tag{3.95}$$

$$R_{angle} = \sigma_{angle}^2 \times \mathbf{I}_6, \qquad R_{gyro} = \sigma_{gyro}^2 \times \mathbf{I}_3, \qquad R_{mag} = \sigma_{mag}^2 \times \mathbf{I}_3 \tag{3.96}$$

$${}^{11}R_v = \begin{bmatrix} R_{GPS} & \mathbf{0}_4 & \mathbf{0}_4 & \mathbf{0}_4 & \mathbf{0}_{4\times3} \\ \mathbf{0}_{3\times4} & R_{distance} & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_{3\times6} \\ \mathbf{0}_6 & \mathbf{0}_{6\times1} & R_{angle} & \mathbf{0}_{6\times3} & \mathbf{0}_{6\times3} \\ \mathbf{0}_3 & \mathbf{0}_{3\times5} & \mathbf{0}_{3\times5} & R_{gyro} & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_{3\times5} & \mathbf{0}_{3\times5} & R_{mag} \end{bmatrix}_{19\times19} \tag{3.97}$$

The first filter to be used is the one destined to obtain the optimal orientation estimation. And, as always, can be represented by a state space.

$$ORIENTATION = \begin{cases} \dot{x} = Ex + Fu + \eta_w, & \eta_w \sim N(0, R_w) \\ y = Gx + \eta_v, & \eta_v \sim N(0, R_v) \end{cases} \tag{3.98}$$

This filter has as states the heading of the three drones, $\psi_i, i \in [1,2,3]$, and the bias from the rate gyroscope, $\beta_i, i \in [1,2,3]$

$$x^T = \left[\psi_1\ \psi_2\ \psi_3\ \beta_1\ \beta_2\ \beta_3\right] \tag{3.99}$$

Although the orientation is a result of process noise, the input of the filter is the rate gyro.

$$u^T = \left[r_1\ r_2\ r_3\right] \tag{3.100}$$

The state space matrices of the Kalman Filter are:

$$E = \begin{bmatrix} \mathbf{0}_3 & -I_3 \\ \mathbf{0}_3 & \mathbf{0}_3 \end{bmatrix}_{6\times 6} \qquad F = \begin{bmatrix} I_3 \\ \mathbf{0}_3 \end{bmatrix}_{6\times 3} \qquad G = \begin{bmatrix} I_3 & \mathbf{0}_3 \end{bmatrix}_{3\times 6} \tag{3.101}$$

The output function is the magnetometer, therefore the process noise, $\eta_w \sim N(0, R_w)$ and the sensors' error, $\eta_v \sim N(0, R_v)$ are defined as:

$$R_{gyro} = \sigma^2_{gyro} \times \mathbf{I}_3 \qquad R_w = \begin{bmatrix} R_{gyro} & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 \end{bmatrix}_{6\times 6} \qquad R_v = \sigma^2_{mag} \times \mathbf{I}_3 \tag{3.102}$$

With the orientation estimated, the values of the angle sensor must be expressed in the inertial referential before they are used in the positioning filter. For that equation (3.85) is used in the rotational matrix block's subsystem.

The implementation of the Extended Kalman Filter whose goal is to provide the optimal estimate of the positioning of all three drones is the same as shown in the 10th configuration, as nothing has changed in the position estimation component.

12th, 13th and 14th Configurations - 3D Positioning

The twelfth configuration is the first three-dimensional simulation. Once again, the simulation in state space representation.

$$\begin{cases} \dot{x} = Ex + Fu + \eta_x, & \eta_x \sim N(0, R_x) \\ y = h(x, u) + \eta_v, & \eta_v, \quad \eta_v \sim N(0, R_v) \end{cases} \tag{3.103}$$

In this iteration the states are the position of drone 1, $\vec{p_1}$, the distance between drone 3 and drone 1, $\vec{d_{1,3}}$, the distance between drone 3 and drone 2, $\vec{d_{2,3}}$ and the respective velocities, with coordinates (x, y, z).

In continuous time the simulation is defined as follows:

$$x^T = \begin{bmatrix} \vec{p_1} & \vec{d_{1,3}} & \vec{d_{3,2}} & \dot{\vec{p_1}} & \dot{\vec{d}}_{1,3} & \dot{\vec{d}}_{3,2} \end{bmatrix} \tag{3.104}$$

The inputs are, as before, the linear accelerations.

$$u^T = \begin{bmatrix} \vec{a_1} & \vec{a_2} & \vec{a_3} \end{bmatrix} \tag{3.105}$$

The state space representation is:

$$\dot{x} = \begin{bmatrix} & \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ \mathbf{0}_3 & \\ \mathbf{0}_3 & \mathbf{0}_3 \end{bmatrix}_{6\times 6} x + \begin{bmatrix} \mathbf{0}_3 \\ \begin{bmatrix} 1 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & -1 \end{bmatrix} \end{bmatrix}_{6\times 3} u \tag{3.106}$$

In this iteration, the output function is also based in GPS, distance and angle sensors.

$$\begin{bmatrix} \vec{p_2} \\ \vec{d_{1,2}} \end{bmatrix} = \begin{bmatrix} \vec{p_1} + \vec{d_{1,3}} + \vec{d_{3,2}} \\ \vec{d_{1,3}} + \vec{d_{3,2}} \end{bmatrix} \tag{3.107}$$

Taking in consideration the relation above, as seen previously, three sensors' configurations were tested, being h_{11}, h_{12} and h_{13}.

The firsts configuration, h_{11}, is composed by GPS in two drones and three distance sensors:

$$h_{12}^T = \begin{bmatrix} \vec{p_1} & \vec{p_2} & ||\vec{d_{1,3}}|| & ||\vec{d_{3,2}}|| & ||\vec{d_{1,2}}|| \end{bmatrix} \tag{3.108}$$

The second configuration, h_{12}, is composed by GPS in two drones and three normalised distance vectors:

$$h_{13}^T = \begin{bmatrix} \vec{p_1} & \vec{p_2} & \frac{\vec{d_{1,3}}}{||\vec{d_{1,3}}||} & \frac{\vec{d_{3,2}}}{||\vec{d_{3,2}}||} & \frac{\vec{d_{1,2}}}{||\vec{d_{1,2}}||} \end{bmatrix} \tag{3.109}$$

The final configuration, h_{13}, is a combination of the previous two.

$$h_{14}^T = \begin{bmatrix} \vec{p_1} & \vec{p_2} & ||\vec{d_{1,3}}|| & ||\vec{d_{3,2}}|| & ||\vec{d_{1,2}}|| & \frac{\vec{d_{1,3}}}{||\vec{d_{1,3}}||} & \frac{\vec{d_{3,2}}}{||\vec{d_{3,2}}||} & \frac{\vec{d_{1,2}}}{||\vec{d_{1,2}}||} \end{bmatrix} \tag{3.110}$$

In order to implement the Extended Kalman Filter the Jacobians must be defined. The three output functions are concatenations of three different sensors, GPS, distance and normalised distance vectors, therefore, the three Jacobians will also be a concatenation of three matrices. It should be noticed that the states are in the form (x, y, z) and not $\vec{v}$, therefore there are 18 states instead of 6.

The Jacobian of the GPS is:

$$J_{GPS} = \begin{bmatrix} I_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_{3\times 9} \\ I_3 & I_3 & I_3 & \mathbf{0}_{3\times 9} \end{bmatrix}_{6\times 18} \tag{3.111}$$

The Jacobian of the distance sensor is:

$$J_{\vec{d_{i,j}}} = \begin{bmatrix} \mathbf{0}_3 & \begin{bmatrix} a_1 & b_1 & c_1 \\ 0 & 0 & 0 \\ g_1 & k_1 & l_1 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 \\ d_1 & e_1 & f_1 \\ g_1 & k_1 & l_1 \end{bmatrix} & \mathbf{0}_{3\times 9} \end{bmatrix}_{3\times 18} \tag{3.112}$$

$$a_1 = \frac{d_{(1,3)x}}{||\vec{d_{1,3}}||}, \qquad b_1 = \frac{d_{(1,3)y}}{||\vec{d_{1,3}}||}, \qquad c_1 = \frac{d_{(1,3)z}}{||\vec{d_{1,3}}||} \tag{3.113}$$

$$d_1 = \frac{d_{(3,2)x}}{||\vec{d_{3,2}}||}, \qquad e_1 = \frac{d_{(3,2)y}}{||\vec{d_{3,2}}||}, \qquad f_1 = \frac{d_{(3,2)z}}{||\vec{d_{3,2}}||} \tag{3.114}$$

$$g_1 = \frac{d_{(1,2)x}}{||\vec{d_{1,2}}||}, \qquad k_1 = \frac{d_{(1,2)y}}{||\vec{d_{1,2}}||}, \qquad l_1 = \frac{d_{(1,2)z}}{||\vec{d_{1,2}}||} \tag{3.115}$$

The Jacobian of the normalised distance vectors is:

$$J_{\frac{\vec{d_{i,j}}}{||\vec{d_{i,j}}||}} = \begin{bmatrix} \mathbf{0}_3 & \begin{bmatrix} a_2 & b_2 & c_2 \\ b_2 & d_2 & e_2 \\ c_2 & e_2 & f_2 \end{bmatrix} & \mathbf{0}_{3\times 12} \\ \mathbf{0}_{3\times 6} & \begin{bmatrix} g_2 & k_2 & l_2 \\ k_2 & m_2 & n_2 \\ l_2 & n_2 & p_2 \end{bmatrix} & \mathbf{0}_{3\times 9} \\ \mathbf{0}_3 & \begin{bmatrix} q_2 & r_2 & s_2 & q_2 & r_2 & s_2 \\ r_2 & t_2 & u_2 & r_2 & t_2 & u_2 \\ s_2 & u_2 & v_2 & s_2 & u_2 & v_2 \end{bmatrix} & \mathbf{0}_{3\times 9} \end{bmatrix}_{9\times 18} \tag{3.116}$$

$$a_2 = \frac{d^2_{(1,3)y} + d^2_{(1,3)z}}{||\vec{d_{1,3}}||^3}, \quad b_2 = -\frac{d_{(1,3)x} \times d_{(1,3)y}}{||\vec{d_{1,3}}||^3}, \quad c_2 = -\frac{d_{(1,3)x} \times d_{(1,3)z}}{||\vec{d_{1,3}}||^3} \tag{3.117}$$

$$d_2 = \frac{d^2_{(1,3)x} + d^2_{(1,3)z}}{||\vec{d_{1,3}}||^3}, \quad e_2 = -\frac{d_{(1,3)y} \times d_{(1,3)z}}{||\vec{d_{1,3}}||^3}, \quad f_2 = \frac{d^2_{(1,3)x} + d^2_{(1,3)y}}{||\vec{d_{1,3}}||^3} \tag{3.118}$$

$$g_2 = \frac{d^2_{(3,2)y} + d^2_{(3,2)z}}{||\vec{d_{3,2}}||^3}, \quad k_2 = -\frac{d_{(3,2)x} \times d_{(3,2)y}}{||\vec{d_{3,2}}||^3}, \quad l_2 = -\frac{d_{(3,2)x} \times d_{(3,2)z}}{||\vec{d_{3,2}}||^3} \tag{3.119}$$

$$m_2 = \frac{d^2_{(3,2)x} + d^2_{(3,2)z}}{||\vec{d_{3,2}}||^3}, \quad n_2 = -\frac{d_{(3,2)y} \times d_{(3,2)z}}{||\vec{d_{3,2}}||^3}, \quad p_2 = \frac{d^2_{(3,2)x} + d^2_{(3,2)y}}{||\vec{d_{3,2}}||^3} \tag{3.120}$$

Bearing in mind equation (3.102), that is: $\vec{d_{2,1}} = \vec{d_{1,3}} + \vec{d_{3,2}}$, we got:

$$q_2 = \frac{d^2_{(1,2)y} + d^2_{(1,2)z}}{||\vec{d_{1,2}}||^3}, \quad r_2 = -\frac{d_{(1,2)x} \times d_{(1,2)y}}{||\vec{d_{1,2}}||^3}, \quad s_2 = -\frac{d_{(1,2)x} \times d_{(1,2)z}}{||\vec{d_{1,2}}||^3} \tag{3.121}$$

$$t_2 = \frac{d^2_{(1,2)x} + d^2_{(1,2)z}}{||\vec{d_{1,2}}||^3}, \quad u_2 = -\frac{d_{(1,2)y} \times d_{(1,2)z}}{||\vec{d_{1,2}}||^3}, \quad v_2 = \frac{d^2_{(1,2)x} + d^2_{(1,2)y}}{||\vec{d_{1,2}}||^3} \tag{3.122}$$

The process noise and the sensors' error is analogous to the ones detailed in the tenth configuration but now in three-dimensions.

EKF - 3D Drone Positioning and Orientation

The final configuration is a three-dimensional with orientation implementation. It is a fusion between the last two subsections.

As in the 2D configuration with orientation the transformation between the inertial and local referential

must be taken in account.

From this moment on, each local referential, $\{V_i\}$, $i \in [1,2,3]$, sympathetic with the drone's kinematic, is not only defined by its position, ${}^{I}p_i$, $i \in [1,2,3]$, but also, its orientation, $[\phi_i, \theta_i, \psi_i]$, $i \in [1,2,3]$, in an inertial referential, $\{I\}$.

A vector defined at any local referential, $\{V_i\}$, can be transformed to the inertial referential, $\{I\}$, employing sequentially the three rotations, each one given by its rotation matrix.

From section 2.1.3, we got:

The Rotation Matrix of the ith aircraft is defined as:

$$ {}^{i}R = {}^{i}\ R(\psi)^{i}R(\theta)^{i}R(\phi) = \begin{bmatrix} c_\psi c_\theta & c_\psi s_\theta s_\phi - s_\psi c_\phi & c_\psi s_\theta c_\phi + s_\psi s_\phi \\ s_\psi c_\theta & s_\psi s_\theta s_\phi + c_\psi c_\theta & s_\psi s_\theta c_\phi - c_\psi s_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix} \tag{3.123} $$

The inverse Rotation Matrix of the ith aircraft is defined as:

$$ {}^{I}R = {}^{I}\ R(\phi)^{I}R(\theta)^{I}R(\psi) = \begin{bmatrix} c_\psi c_\theta & s_\psi c_\theta & -s_\theta \\ c_\psi s_\theta s_\phi - s_\psi c_\phi & s_\psi s_\theta s_\phi + c_\psi c_\theta & c_\theta s_\phi \\ c_\psi s_\theta c_\phi + s_\psi s_\phi & s_\psi s_\theta c_\phi - c_\psi s_\phi & c_\theta c_\phi \end{bmatrix} \tag{3.124} $$

The notation $c_\xi = cos(\xi)$ and $s_\xi = sin(\xi)$ was used. Must bear in mind that the product order must be the same as the transformation rotations between referential.

Any vector defined at any local referential is transformed to the inertial referential by:

$$ {}^{i}R_i^{V_i} p_j = {}^{I}\ p_j, \qquad i,j \in [1,2,3] \tag{3.125} $$

A vector defined at the inertial referential is transformed to any local referential by:

$$ {}^{I}R_i^{I} p_j = {}^{V_i}\ p_j, \qquad i,j \in [1,2,3] \tag{3.126} $$

As seen in the block diagrams of centralised position and orientation, in subsection 3.1.2, two filters are needed to obtain the optimal prediction of orientation and positioning. Therefore, for the first time, the simulation equations are not the same as the equations that define the filters.

Bearing this in mind, two state space representations are presented.

The first one is the drone's simulation subsystem and it is represented as follows:

$$ SIMULATION = \begin{cases} \dot{x} = Ex + Fu + \eta_w, & \eta_w \sim N(0, R_w) \\ y = h(x,u) + \eta_v, & \eta_v \sim N(0, R_v) \end{cases} \tag{3.127} $$

In this iteration the simulation states are the position of drone 1, $\vec{p_1}$, the relative position of drone 1 and drone 3, $\vec{d_{1,3}}$, as well as drone 3 and drone 2, $\vec{d_{3,2}}$ and the respective velocities, as previously, plus

the orientation $[\phi_i, \theta_i, \psi_i]$, with $i \in [1, 2, 3]$. All coordinates are now (x, y, z).

$$x^T = \left[\vec{p_1}\ \vec{d_{1,3}}\ \vec{d_{3,2}}\ \phi_i\ \theta_i\ \psi_i\ \dot{\vec{p_1}}\ \dot{\vec{d_{1,3}}}\ \dot{\vec{d_{3,2}}}\ p_i\ q_i\ r_i\right] \tag{3.128}$$

The simulation inputs are, as before, the linear accelerations.

$$u^T = \left[\vec{a_1}\ \vec{a_2}\ \vec{a_3}\right] \tag{3.129}$$

The simulation state space matrices are:

$$E = \begin{bmatrix} \mathbf{0}_{18} & I_{18} \\ \mathbf{0}_{18} & \mathbf{0}_{18} \end{bmatrix}_{36\times 36} \qquad F = \begin{bmatrix} & \mathbf{0}_9 & \\ & \mathbf{0}_9 & \\ I_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ -I_3 & \mathbf{0}_3 & I_3 \\ \mathbf{0}_3 & I_3 & -I_3 \\ & \mathbf{0}_9 & \end{bmatrix}_{36\times 6} \tag{3.130}$$

The measurement function of the simulation is based on GPS, distance and angle sensor, rate gyro, accelerometer and magnetometer. It should be noted that the output of the angle sensor is expressed in the local referential, that is given by equation (3.126), as the states are all expressed in the inertial referential. The output will be shown in two different measurement functions, ${}^{O}h_{15}$, referent to the orientation and, ${}^{P}h_{15}$, referring to the positioning output function.

$${}^{P}h_{15}^T = \left[\vec{p_1}\ \vec{p_2}\ ||\vec{d_{1,3}}||\ ||\vec{d_{3,2}}||\ ||\vec{d_{1,2}}||\ \frac{\vec{d_{1,3}}}{||\vec{d_{1,3}}||}\ \frac{\vec{d_{3,2}}}{||\vec{d_{3,2}}||}\ \frac{\vec{d_{1,2}}}{||\vec{d_{1,2}}||}\right] \tag{3.131}$$

$${}^{O}h_{15}^T = \left[\psi_i\ gyro_i\ acc_i\right], i \in [1, 2, 3] \tag{3.132}$$

The process noise, $\eta_w \sim N(0, R_w)$ and the sensors' error, $\eta_v \sim N(0, R_v)$ are defined as:

$$R_{linear} = \sigma_{linear}^2 \times \mathbf{I}_9, \qquad R_{orientation} = \begin{bmatrix} \sigma_\phi^2 & 0 & 0 \\ 0 & \sigma_\theta^2 & 0 \\ 0 & 0 & \sigma_\psi^2 \end{bmatrix} \tag{3.133}$$

$$R_w = \begin{bmatrix} R_{linear} & \mathbf{0}_{9\times 6} & \mathbf{0}_{9\times 6} & \mathbf{0}_{9\times 6} & \mathbf{0}_9 \\ \mathbf{0}_9 & \mathbf{0}_{9\times 6} & \mathbf{0}_{9\times 6} & \mathbf{0}_{9\times 6} & \mathbf{0}_9 \\ \mathbf{0}_{9\times 18} & R_{linear} & \mathbf{0}_{9\times 3} & \mathbf{0}_{9\times 3} & \mathbf{0}_{9\times 3} \\ \mathbf{0}_3 & \mathbf{0}_{3\times 24} & R_{orientation} & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_{3\times 24} & R_{orientation} & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_{3\times 24} & R_{orientation} \end{bmatrix}_{36\times 36} \tag{3.134}$$

$$R_{GPS} = \sigma^2_{GPS,x} \times \mathbf{I}_6, \qquad R_{distance} = \sigma^2_{distance} \times \mathbf{I}_3, \qquad R_{angle} = \sigma^2_{angle} \times \mathbf{I}_9 \tag{3.135}$$

$$R_{acc} = \sigma^2_{acc} \times \mathbf{I}_9, \qquad R_{gyro} = \sigma^2_{gyro} \times \mathbf{I}_9, \qquad R_{mag} = \sigma^2_{mag} \times \mathbf{I}_3 \tag{3.136}$$

$$R_v = \begin{bmatrix} R_{GPS} & \mathbf{0}_6 & \mathbf{0}_6 & \mathbf{0}_6 & \mathbf{0}_6 & \mathbf{0}_{6\times 9} \\ \mathbf{0}_{3\times 6} & R_{distance} & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_{3\times 21} \\ \mathbf{0}_{9\times 3} & \mathbf{0}_{9\times 6} & R_{angle} & \mathbf{0}_{9\times 7} & \mathbf{0}_{9\times 7} & \mathbf{0}_{9\times 7} \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_{3\times 12} & R_{mag} & \mathbf{0}_{3\times 9} & \mathbf{0}_{3\times 9} \\ \mathbf{0}_{9\times 5} & \mathbf{0}_{9\times 5} & \mathbf{0}_{9\times 5} & \mathbf{0}_{9\times 6} & R_{gyro} & \mathbf{0}_9 \\ \mathbf{0}_{9\times 6} & \mathbf{0}_{9\times 6} & \mathbf{0}_{9\times 6} & \mathbf{0}_{9\times 6} & \mathbf{0}_{9\times 6} & R_{acc} \end{bmatrix}_{39\times 39} \tag{3.137}$$

The first filter to be used is the one destined to obtain the optimal orientation estimation. And, as always, can be represented by a state space.

$$ORIENTATION = \begin{cases} \dot{x} = Ex + Fu + \eta_w, & \eta_w \sim N(0, R_w) \\ y = h(x, u) + \eta_v, & \eta_v \sim N(0, R_v) \end{cases} \tag{3.138}$$

This filter has as states the orientation of the three drones, $[\phi_i, \theta_i, \psi_i], i \in [1, 2, 3]$, and the bias from the rate gyroscope, $\beta_i, i \in [1, 2, 3]$

$$x^T = \begin{bmatrix} \phi_1 & \theta_1 & \psi_1 & \beta_{1x} & \beta_{1y} & \beta_{1z} & \phi_2 & \theta_2 & \psi_2 & \beta_{2x} & \beta_{2y} & \beta_{2z} & \phi_3 & \theta_3 & \psi_3 & \beta_{3x} & \beta_{3y} & \beta_{3z} \end{bmatrix} \tag{3.139}$$

Even though the orientation is a result of process noise, the system has an input, the rate gyro.

$$u^T = \begin{bmatrix} p_1 & q_1 & r_1 & p_2 & q_2 & r_2 & p_3 & q_3 & r_3 \end{bmatrix} \tag{3.140}$$

The state space matrices of the Extended Kalman Filter are:

$$E = \begin{bmatrix} \mathbf{0}_3 & -I_3 & \mathbf{0}_{3\times 12} \\ & \mathbf{0}_{3\times 18} & \\ \mathbf{0}_{3\times 9} & -I_3 & \mathbf{0}_{3\times 6} \\ & \mathbf{0}_{3\times 18} & \\ \mathbf{0}_{3\times 9} & \mathbf{0}_{3\times 6} & -I_3 \\ & \mathbf{0}_{3\times 18} & \end{bmatrix}_{18\times 18} \qquad F = \begin{bmatrix} I_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & I_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & I_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \end{bmatrix}_{18\times 9} \tag{3.141}$$

The output function is based on the magnetometer and the accelerometer. Recalling from section 2.1 equations (2.14) and (2.15):

$${}^{O}h^T = \begin{bmatrix} gsen(\theta_i) & -gcos(\theta_i)sen(\phi_i) & -gcos(\theta_i)cos(\phi_i) & \psi_i \end{bmatrix}, i \in [1, 2, 3] \tag{3.142}$$

And its Jacobian of the orientation measurement function, ${}^{O}h$, is:

$$J_O = \begin{bmatrix} \begin{bmatrix} 0 & a & 0 \\ b & c & 0 \\ d & e & 0 \\ 0 & 0 & 1 \end{bmatrix} & & \mathbf{0}_{4\times15} & \\ \mathbf{0}_{4\times3} & \begin{bmatrix} 0 & f & 0 \\ h & k & 0 \\ l & m & 0 \\ 0 & 0 & 1 \end{bmatrix} & \mathbf{0}_{4\times12} & \\ & \mathbf{0}_{4\times6} & \begin{bmatrix} 0 & n & 0 \\ p & q & 0 \\ r & s & 0 \\ 0 & 0 & 1 \end{bmatrix} & \mathbf{0}_{4\times9} \end{bmatrix}_{12\times18} \tag{3.143}$$

$$a = gcos(\theta_1),\ b = -gcos(\theta_1)cos(\phi_1),\ c = gsin(\theta_1)sin(\phi_1) \tag{3.144}$$

$$d = gcos(\theta_1)sin(\phi_1),\ e = gsin(\theta_1)cos(\phi_1),\ f = gcos(\theta_2) \tag{3.145}$$

$$h = -gcos(\theta_2)cos(\phi_2),\ k = gsin(\theta_2)sin(\phi_2),\ l = gcos(\theta_2)sin(\phi_2) \tag{3.146}$$

$$m = gsin(\theta_2)cos(\phi_2),\ n = gcos(\theta_3),\ p = -gcos(\theta_3)cos(\phi_3) \tag{3.147}$$

$$q = gsin(\theta_3)sin(\phi_3),\ r = gcos(\theta_3)sin(\phi_3),\ s = gsin(\theta_3)cos(\phi_3) \tag{3.148}$$

The process noise, $\eta_w \sim N(0, R_w)$ and the sensors' error, $\eta_v \sim N(0, R_v)$ are defined as:

$$R_{gyro} = \sigma^2_{gyro} \times \mathbf{I}_3 \qquad R_w = \begin{bmatrix} R_{gyro} & \mathbf{0}_{3\times8} & \mathbf{0}_{3\times7} \\ & \mathbf{0}_{3\times18} & \\ \mathbf{0}_{3\times6} & R_{gyro} & \mathbf{0}_{3\times9} \\ & \mathbf{0}_{3\times18} & \\ \mathbf{0}_{3\times12} & R_{gyro} & \mathbf{0}_{3\times3} \\ & \mathbf{0}_{3\times18} & \end{bmatrix}_{18\times18} \tag{3.149}$$

$$R_{acc,mag} = \begin{bmatrix} \sigma^2_{acc,x} & 0 & 0 & 0 \\ 0 & \sigma^2_{acc,y} & 0 & 0 \\ 0 & 0 & \sigma^2_{acc,z} & 0 \\ 0 & 0 & 0 & \sigma^2_{mag} \end{bmatrix} \qquad R_v = \begin{bmatrix} R_{acc,mag} & \mathbf{0}_4 & \mathbf{0}_4 \\ \mathbf{0}_4 & R_{acc,mag} & \mathbf{0}_4 \\ \mathbf{0}_4 & \mathbf{0}_4 & R_{acc,mag} \end{bmatrix}_{12\times12} \tag{3.150}$$

With the orientation estimated, the values of the angle sensor must be expressed in the inertial referential before they are used in the positioning filter. For that equation (3.125) is used in the rotational matrix block's subsystem. The implementation of the Extended Kalman Filter for the optimal positioning estimation was defined in the fourteenth configuration - 3D Positioning.

3.2.3 Non-linear Filter

The non-linear filter has a centralised simulation as well and it is represented as follows:

$$SIMULATION = \begin{cases} \dot{x} = Ex + Fu + \eta_w, & \eta_w \sim N(0, R_w) \\ y = h(x, u) + \eta_v, & \eta_v \sim N(0, R_v) \end{cases} \tag{3.151}$$

In this iteration, the simulation states are the position of drone 1, $\vec{p_1}$, drone 2, $\vec{p_2}$, and drone 3, $\vec{p_3}$ plus the orientation $[\phi_i, \theta_i, \psi_i]$, with $i \in [1, 2, 3]$ and the respective velocities.

$$x^T = \begin{bmatrix} \vec{p_1} & \vec{p_2} & \vec{p_3} & \phi_i & \theta_i & \psi_i & \dot{\vec{p_1}} & \dot{\vec{p_2}} & \dot{\vec{p_3}} & p_i & q_i & r_i \end{bmatrix} \tag{3.152}$$

The simulation inputs are, as before, the linear accelerations.

$$u^T = \begin{bmatrix} \vec{a_1} & \vec{a_2} & \vec{a_3} \end{bmatrix} \tag{3.153}$$

The simulation state space matrices are:

$$E = \begin{bmatrix} \mathbf{0}_{18} & I_{18} \\ \mathbf{0}_{18} & \mathbf{0}_{18} \end{bmatrix}_{36\times 36} \qquad F = \begin{bmatrix} \mathbf{0}_9 \\ \mathbf{0}_9 \\ I_9 \\ \mathbf{0}_9 \end{bmatrix}_{36\times 6} \tag{3.154}$$

The measurement function of the simulation is based on GPS, angle sensor, rate gyro, accelerometer and magnetometer. It should be noted that the output of the angle sensor is expressed in the inertial referential, that is given by equation (3.121), as the states are all expressed in the inertial referential. The output will be shown in two different measurement functions, ${}^{O}h_{16}$, referent to the orientation and, ${}^{P}h_{16}$, referring to the positioning output function.

$${}^{P}h_{16}^T = \begin{bmatrix} \vec{p_1} & \vec{p_2} & \frac{\vec{d_{1,3}}}{||\vec{d_{1,3}}||} & \frac{\vec{d_{2,3}}}{||\vec{d_{2,3}}||} \end{bmatrix} \tag{3.155}$$

$${}^{O}h_{16}^T = \begin{bmatrix} \psi_i & gyro_i & acc_i \end{bmatrix}, i \in [1, 2, 3] \tag{3.156}$$

The first filter to be used is the one destined to obtain the optimal orientation estimation. And is represented by the following state space.

$$ORIENTATION = \begin{cases} \dot{x} = Ex + Fu + \eta_w, & \eta_w \sim N(0, R_w) \\ y = h(x, u) + \eta_v, & \eta_v \sim N(0, R_v) \end{cases} \tag{3.157}$$

This filter has as states the orientation of the three drones, $[\phi_i, \theta_i, \psi_i], i \in [1, 2, 3]$, and the bias from the

rate gyroscope, $\beta_i, i \in [1,2,3]$

$$x^T = \begin{bmatrix} \phi_1 & \theta_1 & \psi_1 & \beta_{1x} & \beta_{1y} & \beta_{1z} & \phi_2 & \theta_2 & \psi_2 & \beta_{2x} & \beta_{2y} & \beta_{2z} & \phi_3 & \theta_3 & \psi_3 & \beta_{3x} & \beta_{3y} & \beta_{3z} \end{bmatrix} \tag{3.158}$$

Even though the orientation is a result of process noise, the system has an input, the rate gyro.

$$u^T = \begin{bmatrix} p_1 & q_1 & r_1 & p_2 & q_2 & r_2 & p_3 & q_3 & r_3 \end{bmatrix} \tag{3.159}$$

The state space matrices of the Extended Kalman Filter are:

$$E = \begin{bmatrix} \mathbf{0}_3 & -I_3 & \mathbf{0}_{3\times12} \\ & \mathbf{0}_{3\times18} & \\ \mathbf{0}_{3\times9} & -I_3 & \mathbf{0}_{3\times6} \\ & \mathbf{0}_{3\times18} & \\ \mathbf{0}_{3\times9} & \mathbf{0}_{3\times6} & -I_3 \\ & \mathbf{0}_{3\times18} & \end{bmatrix}_{18\times18} \qquad F = \begin{bmatrix} I_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & I_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & I_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \end{bmatrix}_{18\times9} \tag{3.160}$$

The output function is based on the magnetometer and the accelerometer. Recalling from section 2.1 equations (2.14) and (2.15):

$${}^{O}h^T = \begin{bmatrix} gsen(\theta_i) & -gcos(\theta_i)sen(\phi_i) & -gcos(\theta_i)cos(\phi_i) & \psi_i \end{bmatrix}, i \in [1,2,3] \tag{3.161}$$

And its Jacobian of the orientation measurement function, ${}^{O}h$, is:

$$J_O = \begin{bmatrix} \begin{bmatrix} 0 & a & 0 \\ b & c & 0 \\ d & e & 0 \\ 0 & 0 & 1 \end{bmatrix} & & \mathbf{0}_{4\times15} & \\ \mathbf{0}_{4\times3} & \begin{bmatrix} 0 & f & 0 \\ h & k & 0 \\ l & m & 0 \\ 0 & 0 & 1 \end{bmatrix} & \mathbf{0}_{4\times12} & \\ & \mathbf{0}_{4\times6} & \begin{bmatrix} 0 & n & 0 \\ p & q & 0 \\ r & s & 0 \\ 0 & 0 & 1 \end{bmatrix} & \mathbf{0}_{4\times9} \end{bmatrix}_{12\times18} \tag{3.162}$$

$$a = gcos(\theta_1),\ b = -gcos(\theta_1)cos(\phi_1),\ c = gsin(\theta_1)sin(\phi_1) \tag{3.163}$$

$$d = gcos(\theta_1)sin(\phi_1),\ e = gsin(\theta_1)cos(\phi_1),\ f = gcos(\theta_2) \tag{3.164}$$

$$h = -gcos(\theta_2)cos(\phi_2),\ k = gsin(\theta_2)sin(\phi_2),\ l = gcos(\theta_2)sin(\phi_2) \tag{3.165}$$

$$m = gsin(\theta_2)cos(\phi_2),\ n = gcos(\theta_3),\ p = -gcos(\theta_3)cos(\phi_3) \tag{3.166}$$

$$q = gsin(\theta_3)sin(\phi_3),\ r = gcos(\theta_3)sin(\phi_3),\ s = gsin(\theta_3)cos(\phi_3) \tag{3.167}$$

The process noise, $\eta_w \sim N(0, R_w)$ and the sensors' error, $\eta_v \sim N(0, R_v)$ are defined as:

$$R_{gyro} = \sigma^2_{gyro} \times \mathbf{I}_3 \qquad R_w = \begin{bmatrix} R_{gyro} & \mathbf{0}_{3\times8} & \mathbf{0}_{3\times7} \\ & \mathbf{0}_{3\times18} & \\ \mathbf{0}_{3\times6} & R_{gyro} & \mathbf{0}_{3\times9} \\ & \mathbf{0}_{3\times18} & \\ \mathbf{0}_{3\times12} & R_{gyro} & \mathbf{0}_{3\times3} \\ & \mathbf{0}_{3\times18} & \end{bmatrix}_{18\times18} \tag{3.168}$$

$$R_{acc,mag} = \begin{bmatrix} \sigma^2_{acc,x} & 0 & 0 & 0 \\ 0 & \sigma^2_{acc,y} & 0 & 0 \\ 0 & 0 & \sigma^2_{acc,z} & 0 \\ 0 & 0 & 0 & \sigma^2_{mag} \end{bmatrix} \qquad R_v = \begin{bmatrix} R_{acc,mag} & \mathbf{0}_4 & \mathbf{0}_4 \\ \mathbf{0}_4 & R_{acc,mag} & \mathbf{0}_4 \\ \mathbf{0}_4 & \mathbf{0}_4 & R_{acc,mag} \end{bmatrix}_{12\times12} \tag{3.169}$$

With the orientation estimated, the values of the angle sensor must be expressed in the inertial referential before they are used in the positioning filter. For that equation (3.125) is used in the rotational matrix block's subsystem.

The next step is to obtain the estimation of Drone 3, using the non-linear filter. The position of drone 1 and 2, $\vec{p_1}$ and $\vec{p_2}$, is assumed to be the reading from the GPS sensors.

$$\dot{\hat{x}} = E\hat{x} + Fu + l\sum_{i=1}^{2} S(\frac{\vec{d_{i,3}}}{||\vec{d_{i,3}}||})^2(\hat{\vec{p_3}} - \vec{p_i}) \tag{3.170}$$

The state is the position of drone 3 and its velocity:

$$x^T = \begin{bmatrix} \vec{p_3} & \dot{\vec{p_3}} \end{bmatrix} \tag{3.171}$$

The control vector is the linear acceleration of drone 3.

$$u = \vec{a_3} \tag{3.172}$$

The matrices E and F are:

$$E = \begin{bmatrix} \mathbf{0}_3 & I_3 \\ \mathbf{0}_3 & \mathbf{0}_3 \end{bmatrix}_{6\times6} \qquad F = \begin{bmatrix} \mathbf{0}_3 \\ I_3 \end{bmatrix}_{6\times3} \tag{3.173}$$

The gain, l, is the following matrix, with $a = 0.5$ and $b = 0.4$:

$$l = \begin{bmatrix} aI_3 \\ bI_3 \end{bmatrix}_{6\times 3} \tag{3.174}$$

3.2.4 Summary of States, Inputs and Outputs

	SIMULATION		
	States	Inputs	Outputs
Config.:			
1	$\vec{p_1}\dot{\vec{p_1}}$	$\vec{a_1}$	$\vec{p_1}$
2	$\vec{p_3}\dot{\vec{p_3}}$	$\vec{a_3}\hat{\vec{p_1}}$	${}^I\vec{d_{1,3}}$
3	$\vec{p_3}\dot{\vec{p_3}}$	$\vec{a_3}\hat{\vec{p_1}}\hat{\vec{p_2}}$	${}^I\vec{d_{1,3}}\ {}^I\vec{d_{3,2}}$
4	$\vec{p_1}\dot{\vec{p_1}}$	$\vec{a_1}\hat{\vec{p_2}}$	$\vec{p_1}{}^I\vec{d_{1,2}}$
5	$\vec{p_3}\dot{\vec{p_3}}$	$\vec{a_3}\hat{\vec{p_1}}\hat{\vec{p_2}}$	$\|\|\vec{d_{1,3}}\|\|\ \|\|\vec{d_{3,2}}\|\|$
6	$\vec{p_3}\dot{\vec{p_3}}$	$\vec{a_3}\hat{\vec{p_1}}\hat{\vec{p_2}}$	$\alpha_{3,1}\ \alpha_{3,2}$
7	$\vec{p_3}\dot{\vec{p_3}}$	$\vec{a_3}\hat{\vec{p_1}}\hat{\vec{p_2}}$	$\alpha_{3,1}\ \alpha_{3,2}\ \|\|\vec{d_{1,3}}\|\|\ \|\|\vec{d_{3,2}}\|\|$
8	$\vec{p_1}\vec{d_{1,3}}\vec{d_{3,2}}\dot{\vec{p_1}}\dot{\vec{d_{1,3}}}\dot{\vec{d_{3,2}}}$	$\vec{a_1}\vec{a_2}\vec{a_3}$	$\vec{p_1}\vec{p_2}\|\|\vec{d_{1,3}}\|\|\|\|\vec{d_{3,2}}\|\|\|\|\vec{d_{1,2}}\|\|$
9	$\vec{p_1}\vec{d_{1,3}}\vec{d_{3,2}}\dot{\vec{p_1}}\dot{\vec{d_{1,3}}}\dot{\vec{d_{3,2}}}$	$\vec{a_1}\vec{a_2}\vec{a_3}$	$\vec{p_1}\vec{p_2}\frac{\vec{d_{1,3}}}{\|\|\vec{d_{1,3}}\|\|}\frac{\vec{d_{3,2}}}{\|\|\vec{d_{3,2}}\|\|}\frac{\vec{d_{1,2}}}{\|\|\vec{d_{1,2}}\|\|}$
10	$\vec{p_1}\vec{d_{1,3}}\vec{d_{3,2}}\dot{\vec{p_1}}\dot{\vec{d_{1,3}}}\dot{\vec{d_{3,2}}}$	$\vec{a_1}\vec{a_2}\vec{a_3}$	$\vec{p_1}\vec{p_2}\|\|\vec{d_{1,3}}\|\|\|\|\vec{d_{3,2}}\|\|\|\|\vec{d_{1,2}}\|\|\frac{\vec{d_{1,3}}}{\|\|\vec{d_{1,3}}\|\|}\frac{\vec{d_{3,2}}}{\|\|\vec{d_{3,2}}\|\|}\frac{\vec{d_{1,2}}}{\|\|\vec{d_{1,2}}\|\|}$
11	$\vec{p_1}\vec{d_{1,3}}\vec{d_{3,2}}\psi_1\psi_2\psi_3$ $\dot{\vec{p_1}}\dot{\vec{d_{1,3}}}\dot{\vec{d_{3,2}}}r_1r_2r_3$	$\vec{a_1}\vec{a_2}\vec{a_3}$	$\vec{p_1}\vec{p_2}\|\|\vec{d_{1,3}}\|\|\|\|\vec{d_{3,2}}\|\|\|\|\vec{d_{1,2}}\|\|\frac{\vec{d_{1,3}}}{\|\|\vec{d_{1,3}}\|\|}\frac{\vec{d_{3,2}}}{\|\|\vec{d_{3,2}}\|\|}\frac{\vec{d_{1,2}}}{\|\|\vec{d_{1,2}}\|\|}$ $\psi_1\psi_2\psi_3r_1r_2r_3$
12	$\vec{p_1}\vec{d_{1,3}}\vec{d_{3,2}}\dot{\vec{p_1}}\dot{\vec{d_{1,3}}}\dot{\vec{d_{3,2}}}$	$\vec{a_1}\vec{a_2}\vec{a_3}$	$\vec{p_1}\vec{p_2}\|\|\vec{d_{1,3}}\|\|\|\|\vec{d_{3,2}}\|\|\|\|\vec{d_{1,2}}\|\|$
13	$\vec{p_1}\vec{d_{1,3}}\vec{d_{3,2}}\dot{\vec{p_1}}\dot{\vec{d_{1,3}}}\dot{\vec{d_{3,2}}}$	$\vec{a_1}\vec{a_2}\vec{a_3}$	$\vec{p_1}\vec{p_2}\frac{\vec{d_{1,3}}}{\|\|\vec{d_{1,3}}\|\|}\frac{\vec{d_{3,2}}}{\|\|\vec{d_{3,2}}\|\|}\frac{\vec{d_{1,2}}}{\|\|\vec{d_{1,2}}\|\|}$
14	$\vec{p_1}\vec{d_{1,3}}\vec{d_{3,2}}\dot{\vec{p_1}}\dot{\vec{d_{1,3}}}\dot{\vec{d_{3,2}}}$	$\vec{a_1}\vec{a_2}\vec{a_3}$	$\vec{p_1}\vec{p_2}\|\|\vec{d_{1,3}}\|\|\|\|\vec{d_{3,2}}\|\|\|\|\vec{d_{1,2}}\|\|\frac{\vec{d_{1,3}}}{\|\|\vec{d_{1,3}}\|\|}\frac{\vec{d_{3,2}}}{\|\|\vec{d_{3,2}}\|\|}\frac{\vec{d_{1,2}}}{\|\|\vec{d_{1,2}}\|\|}$
15	$\vec{p_1}\vec{d_{1,3}}\vec{d_{3,2}}\phi_i\theta_i\psi_i$ $\dot{\vec{p_1}}\dot{\vec{d_{1,3}}}\dot{\vec{d_{3,2}}}p_iq_ir_i$	$\vec{a_1}\vec{a_2}\vec{a_3}$	$\vec{p_1}\vec{p_2}\|\|\vec{d_{1,3}}\|\|\|\|\vec{d_{3,2}}\|\|\|\|\vec{d_{1,2}}\|\|\frac{\vec{d_{1,3}}}{\|\|\vec{d_{1,3}}\|\|}\frac{\vec{d_{3,2}}}{\|\|\vec{d_{3,2}}\|\|}\frac{\vec{d_{1,2}}}{\|\|\vec{d_{1,2}}\|\|}\ \psi_igyro_iacc_i$
16	$\vec{p_1}\vec{p_2}\vec{p_3}\phi_i\theta_i\psi_i$ $\dot{\vec{p_1}}\dot{\vec{p_2}}\dot{\vec{p_3}}p_iq_ir_i$	$\vec{a_1}\vec{a_2}\vec{a_3}$	$\vec{p_1}\vec{p_2}\|\|\vec{d_{1,3}}\|\|\|\|\vec{d_{3,2}}\|\|\|\|\vec{d_{1,2}}\|\|\frac{\vec{d_{1,3}}}{\|\|\vec{d_{1,3}}\|\|}\frac{\vec{d_{3,2}}}{\|\|\vec{d_{3,2}}\|\|}\frac{\vec{d_{1,2}}}{\|\|\vec{d_{1,2}}\|\|}\ \psi_igyro_iacc_i$

Table 3.2: States, Inputs and Outputs in simulations.

The states of the position filter are the same states present in the simulation, the inputs of the position filter are the inputs plus the outputs (sensors) of the simulation and the outputs of the filter are the drones' position. The exceptions of are the 11th 15th and 16th configurations, which are described below:

	ORIENTATION FILTER			POSITION FILTER		
	States	Inputs	Outputs	States	Inputs	Outputs
Config.:						
11	$\psi_i \ \beta_i$	r_i	ψ_i	$\vec{p_1}\vec{d_{1,3}}\vec{d_{3,2}}$	$\vec{a_1}\vec{a_2}\vec{a_3} \ \vec{p_1}\vec{p_2}$	$\hat{\vec{p_1}}\hat{\vec{p_2}}\hat{\vec{p_3}}$
				$\dot{\vec{p_1}}\dot{\vec{d_{1,3}}}\dot{\vec{d_{3,2}}}$	$\|\|\vec{d_{1,3}}\|\|\|\|\vec{d_{3,2}}\|\|\|\|\vec{d_{1,2}}\|\|$	
					$\frac{\vec{d_{1,3}}}{\|\|\vec{d_{1,3}}\|\|} \frac{\vec{d_{3,2}}}{\|\|\vec{d_{3,2}}\|\|} \frac{\vec{d_{1,2}}}{\|\|\vec{d_{1,2}}\|\|}$	
15	$\phi_i \ \theta_i \ \psi_i \ \beta_{ix} \ \beta_{iy} \ \beta_{iz}$	$p_i \ q_i \ r_i$	$\phi_i \ \theta_i \ \psi_i$	$\vec{p_1}\vec{d_{1,3}}\vec{d_{3,2}}$	$\vec{a_1}\vec{a_2}\vec{a_3} \ \vec{p_1}\vec{p_2}$	$\hat{\vec{p_1}}\hat{\vec{p_2}}\hat{\vec{p_3}}$
				$\dot{\vec{p_1}}\dot{\vec{d_{1,3}}}\dot{\vec{d_{3,2}}}$	$\|\|\vec{d_{1,3}}\|\|\|\|\vec{d_{3,2}}\|\|\|\|\vec{d_{1,2}}\|\|$	
					$\frac{\vec{d_{1,3}}}{\|\|\vec{d_{1,3}}\|\|} \frac{\vec{d_{3,2}}}{\|\|\vec{d_{3,2}}\|\|} \frac{\vec{d_{1,2}}}{\|\|\vec{d_{1,2}}\|\|}$	
16	$\phi_i \ \theta_i \ \psi_i \ \beta_{ix} \ \beta_{iy} \ \beta_{iz}$	$p_i \ q_i \ r_i$	$\phi_i \ \theta_i \ \psi_i$	$\vec{p_3}\dot{\vec{p_3}}$	$\vec{a_3}\vec{p_1}\vec{p_2} \ \frac{\vec{d_{1,3}}}{\|\|\vec{d_{1,3}}\|\|} \frac{\vec{d_{2,3}}}{\|\|\vec{d_{2,3}}\|\|}$	$\vec{p_3}$

Table 3.3: States, Inputs and Outputs of the filters.

Chapter 4

Results

This section shows the quest for the optimal solution. In order to obtain the optimal solution fifteen configurations were implemented and tested. This allowed to understood the impact of the choice of filtering technique, of the sensors' used and the preponderance of the interactions between drones.

The results are organised in three parts, plots, where the real states and the estimations are shown. Tables, that show the simulation's error, the root mean square. And charts, for a better error visualisation and understanding of the path chosen to the optimal solution.

1st Configuration - One Drone

The first Kalman Filter implementation was the fusion of the GPS and the model. Only one drone was considered. This configuration is set in two dimensions, (x, y).

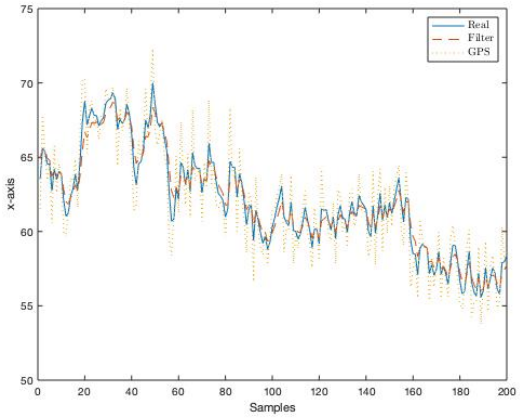

Figure 4.1: 1st Config.: GPS data fusion, x-axis.

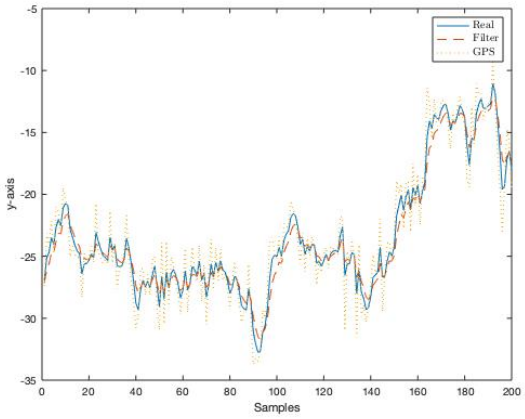

Figure 4.2: 1st Config.: GPS data fusion, y-axis.

The plots, Figure 4.1 and 4.2, show the real position of Drone 1, in blue, the GPS data in yellow and the filter's optimal position estimation in orange.

Table 4.1 displays the average deviation between the real position of Drone 1 and its estimation.

Drone:	$x-axis$	$[m]$	$y-axis$	$[m]$
1		0.6484		0.7594

Table 4.1: 1st Configuration - RMS Deviation Table.

2nd Configuration - Two Drones

The second Kalman Filter implementation was designed to estimate the positioning of Drone 3, a drone without a GPS receiver. For that implementation at least two drones are needed. Drone 1, a drone with a GPS receiver as shown in the first configuration is the anchor and the sensor in Drone 3 allows it to obtain its relative position to Drone 1, $\vec{d_{1,3}}$. This configuration is set in two dimensions, (x, y).

The plots, Figure 4.3 and 4.4 show the real position of Drone 3, in blue and the filter's optimal position estimation in orange. Table 4.2 displays the average deviation between the real position of Drones 1 and 3 and the estimations.

Drone:	$x-axis$	$[m]$	$y-axis$	$[m]$
1		0.6484		0.7594
3		0.5904		0.6903

Table 4.2: 2nd Configuration - RMS Deviation Table.

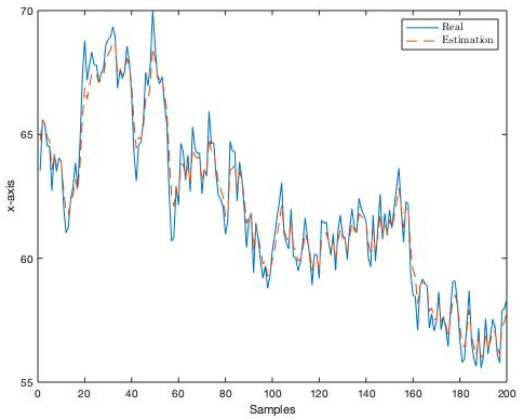

Figure 4.3: 2nd Config.: Drone 3 position estimation, x-axis.

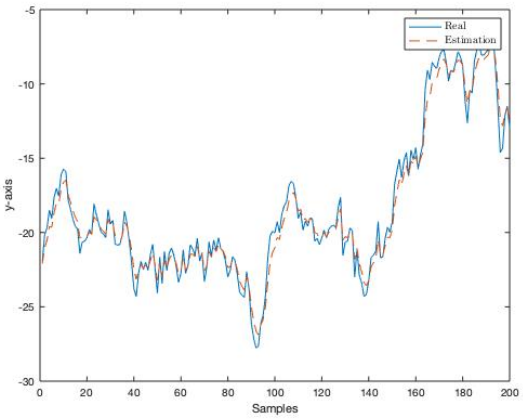

Figure 4.4: 2nd Config.: Drone 3 position estimation, y-axis.

3rd Configuration - Three Drones

The third drone implementation was designed to test the position estimation improvement of Drone 3. For that a third drone was added. Drone two is a drone with a GPS receiver as well as Drone 1.

The plot shows the real position of Drone 1, in blue and the filter's optimal position estimation in orange. The error in the optimal estimation of Drone 3, as seen in Table 4.3 is smaller than in the previous configuration, Table 4.2.

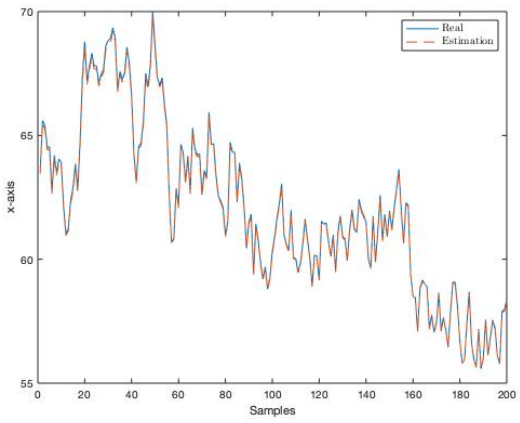

Figure 4.5: 3rd Config.: Drone 3 position estimation, x-axis.

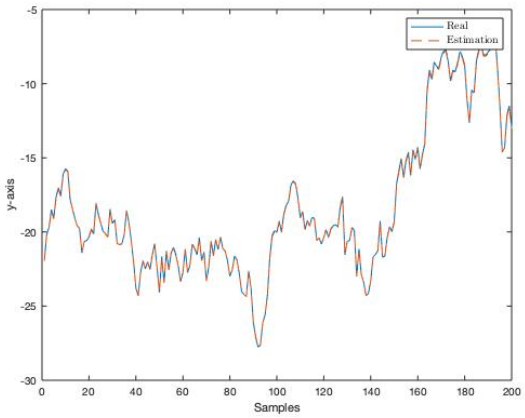

Figure 4.6: 3rd Config.:Drone 3 position estimation, y-axis.

Drone:	$x-axis$ $[m]$	$y-axis$ $[m]$
1	0.6484	0.7594
2	0.6484	0.7594
3	0.1156	0.1633

Table 4.3: 3rd Configuration - RMS Deviation Table.

4th Configuration - Three Drones

In this iteration another relative position sensor was introduced to Drone 1 to interact with Drone 2. The plot shows the real position of Drone 1, in blue and the filter's optimal position estimation in orange. When compared, the results of the fourth configuration, Table 4.4, are much better than the ones from the first configuration, Table 4.1.

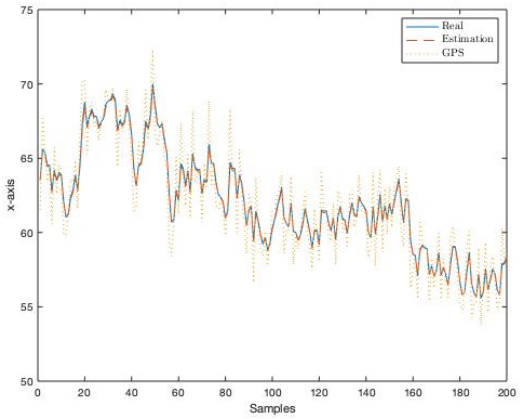

Figure 4.7: 4th Config.: Drone 1 position estimation, x-axis.

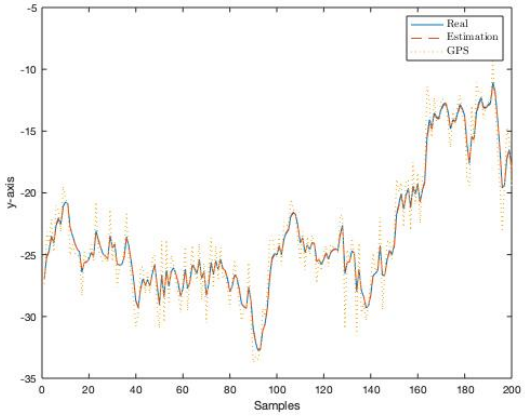

Figure 4.8: 4th Config.: Drone 1 position estimation, y-axis.

Drone:	$x-axis$ $[m]$	$y-axis$ $[m]$
1	0.1251	0.1787
2	0.1251	0.1787
3	0.1156	0.1633

Table 4.4: 4th Configuration - RMS Deviation Table.

5th, 6th and 7th Configurations - Three Drones

From the first to the fourth configuration only Kalman Filters were used, once the outputs were linear. However, that is a simplification. Bearing this in mind, the sensor's output had to be changed. The two alternatives are distance sensors, 5th configuration and angular position sensors, 6th configuration, or the concatenation of both, 7th configuration.

The plots show the real position of Drone 1, in blue and the filter's optimal position estimation in orange. Table 4.5 displays the average deviation between the real position of Drone 3 and its estimation.

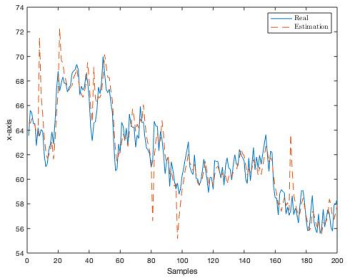

Figure 4.9: 5th Config.: Drone 3 position estimation, x-axis.

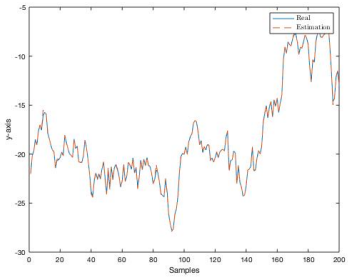

Figure 4.10: 5th Config.: Drone 3 position estimation, y-axis.

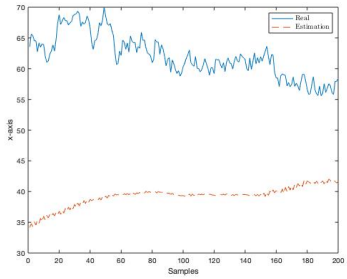

Figure 4.11: 6th Config.: Drone 3 position estimation, x-axis.

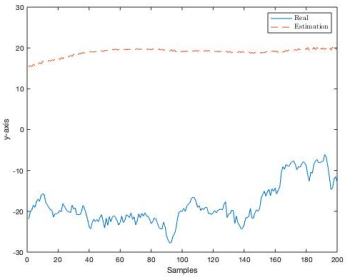

Figure 4.12: 6th Config.: Drone 3 position estimation, y-axis.

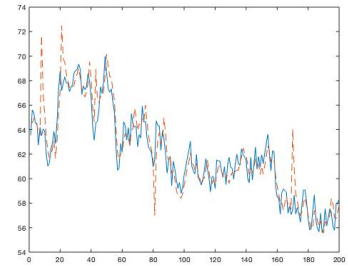

Figure 4.13: 7th Config.: Drone 3 position estimation, x-axis.

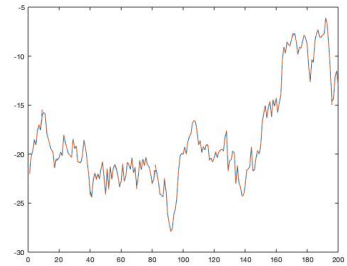

Figure 4.14: 7th Config.: Drone 3 position estimation, y-axis.

Configuration:	$x-axis$ $[m]$	$y-axis$ $[m]$
5	0.9453	0.0492
6	91.2630	40.2206
7	0.9444	0.0494

Table 4.5: 5th, 6th and 7th Configurations - RMS Deviation Table of Drone 3.

8th, 9th and 10th Configurations

The decentralised approach considered each drone as independent identities, but from now on the three drones are in the same simulation. These three configurations are a different approach to the 5th, 6th and 7th configurations.

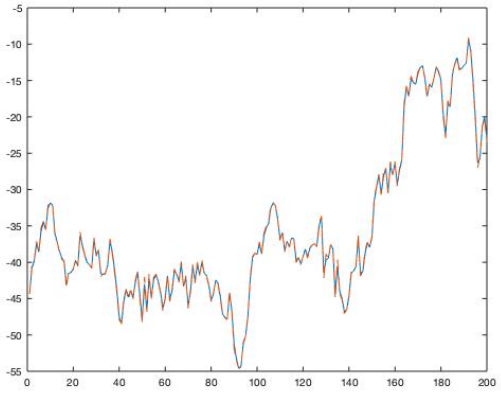

Figure 4.15: 8th Config.: Drone 3 position estimation, x-axis.

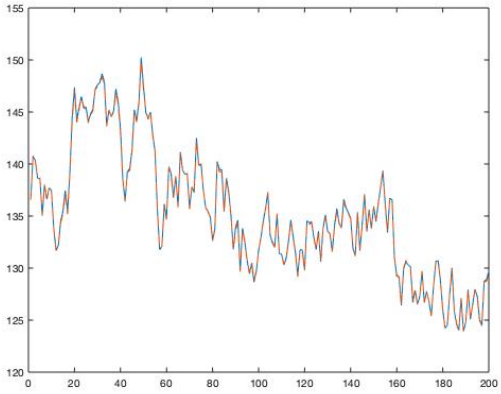

Figure 4.16: 8th Config.:Drone 3 position estimation, y-axis.

The plots show the real position in blue and the filter's optimal position estimation in orange.

Configuration number eight is based on distance sensors. Must be taken in consideration that with the state change, and with the process noise being the same, its influence is now bigger, making the same configurations, fifth and eighth have different deviations.

The ninth configuration uses normalised vectors as the output of the angular sensor.

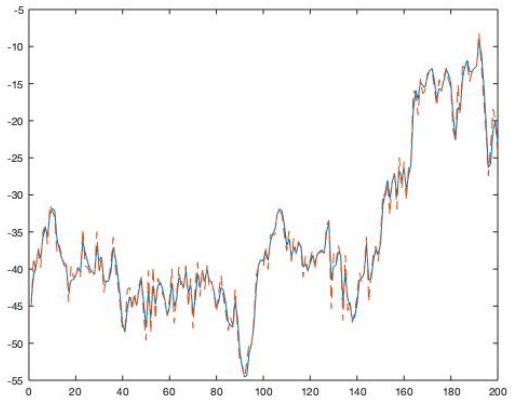

Figure 4.17: 9th Config.: Drone 3 position estimation, x-axis.

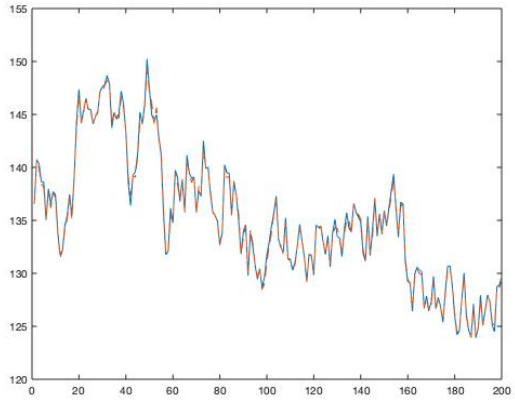

Figure 4.18: 9th Config.: Drone 3 position estimation, y-axis.

The tenth configuration uses normalised vectors as the output of the angular sensor plus the distance sensor.

Table 4.6 displays the average deviation between the real position of Drone 3 and its estimation.

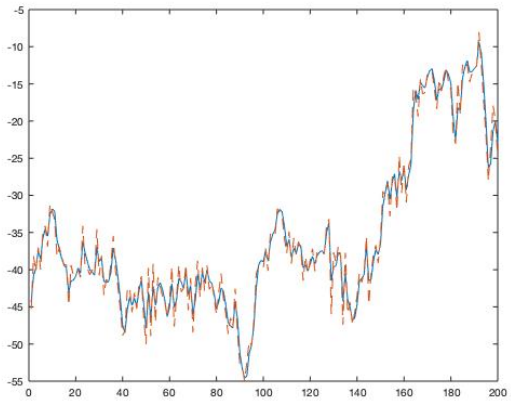

Figure 4.19: 10th Config.: Drone 3 position estimation, x-axis.

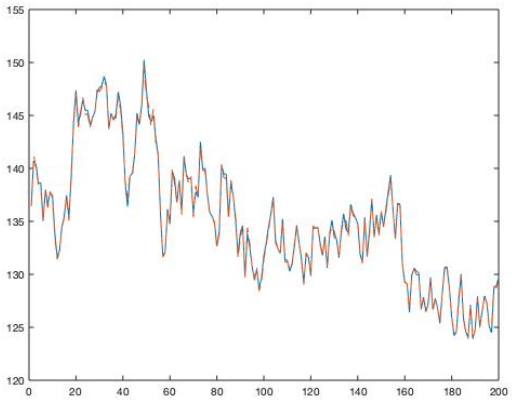

Figure 4.20: 10th Config.: Drone 3 position estimation, y-axis.

Configuration:	$x-axis \quad [m]$	$y-axis \quad [m]$
8	0.5163	1.0386
9	0.6596	0.9294
10	0.7306	0.9777

Table 4.6: 8th, 9th and 10th Configurations - RMS Deviation Table of Drone 3.

Kalman Filter as Complementary Filtering

The following plot shows the real heading, in black streak, from the simulation and the the two estimated headings, from implementation one, in blue, and two, in red.

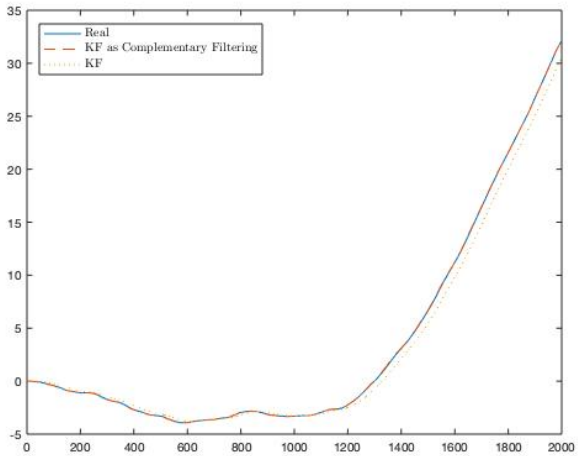

Figure 4.21: Simulation, (blue), KF implementation one, (orange), and implementation 2, (yellow).

The following table outlines the root mean square deviation in the two implementations.

Configuration	RMS Deviation [º]
Implementation 1	0.611
Implementation 2	0.022

Table 4.7: Kalman Filter as Complementary Filtering - RMS Deviation Table.

11th Configuration - 2D Orientation

Until now, every simulation had, by default, null yaw. That means that the orientation of each drone would not change with time, but that is not a realistic approach, rather an intermediate step.

From this moment on, each local referential sympathetic with the drone's kinematic, is not only defined by its position but also, its heading in an inertial referential.

The following plots show the estimation, in blue, and the real orientation, in orange, for all three drones as well as the position estimation of Drone 3, in orange, and its real position, in blue.

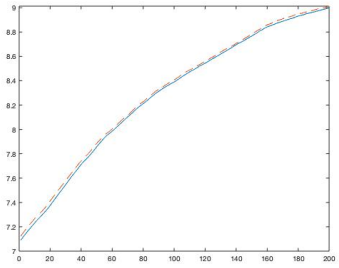

Figure 4.22: 11th Config.: Drone 1 heading estimation.

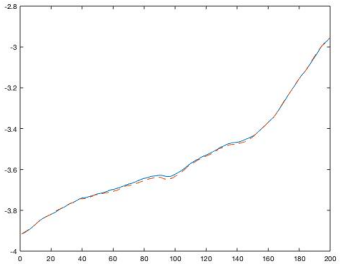

Figure 4.23: 11th Config.: Drone 2 heading estimation.

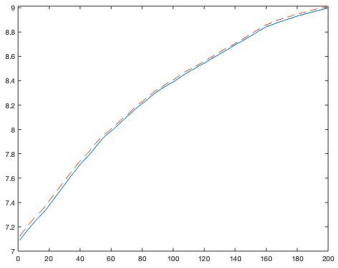

Figure 4.24: 11th Config.: Drone 3 heading estimation.

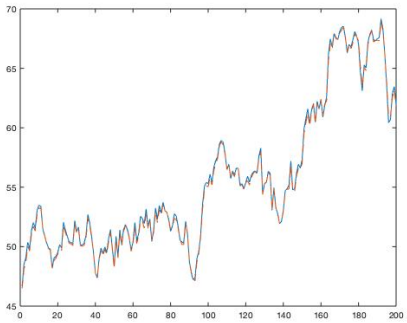

Figure 4.25: 11th Config.: Drone 3 position estimation, x-axis.

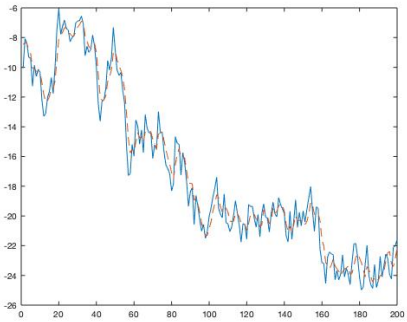

Figure 4.26: 11th Config.: Drone 3 position estimation, y-axis.

Drone:	$x-axis$ $[m]$	$y-axis$ $[m]$
1	0.5270	0.5309
2	1.3247	1.2038
3	0.6243	0.6504

Table 4.8: 11th Configuration - RMS Deviation Table of Drone 3.

12th, 13th and 14th Configurations - 3D Positioning

Once the 2D orientation is already implemented, the next step is to introduce another dimension. From now on the coordinates are (x, y, z).

The first results are from configuration number twelve, Figures 4.27, 4.28, 4.29. The measurement function is based on the distance sensor, $||\vec{d_{i,j}}||$.

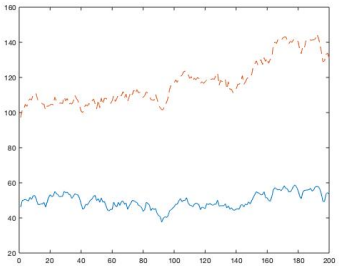

Figure 4.27: 12th Configuration: Drone 3 position estimation, x-axis.

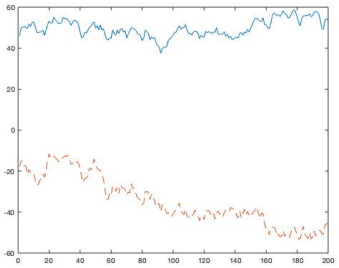

Figure 4.28: 12th Configuration: Drone 3 position estimation, y-axis.

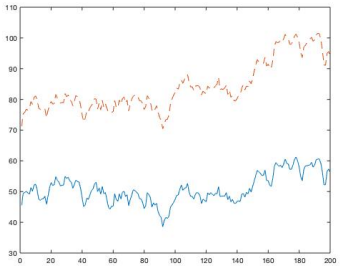

Figure 4.29: 12th Configuration: Drone 3 position estimation, z-axis.

The next configuration, 13th, Figures 4.30, 4.31, 4.32, is based on the normalised vector sensor, $\frac{\vec{d_{i,j}}}{||\vec{d_{i,j}}||}$.

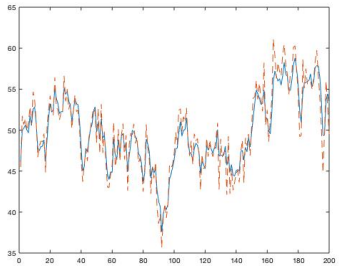

Figure 4.30: 13th Configuration: Drone 3 position estimation, x-axis.

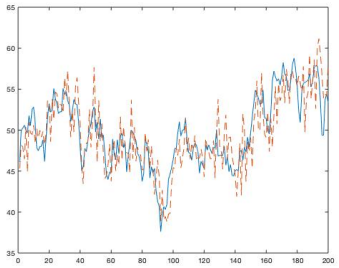

Figure 4.31: 13th Configuration: Drone 3 position estimation, y-axis.

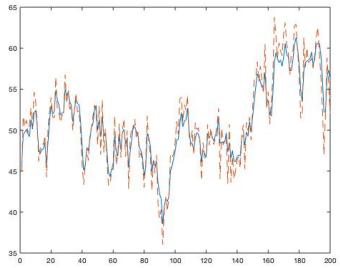

Figure 4.32: 13th Configuration: Drone 3 position estimation, z-axis.

The 14th configuration is based in the concatenation of the previous two, Figures 4.33, 4.34, 4.35.

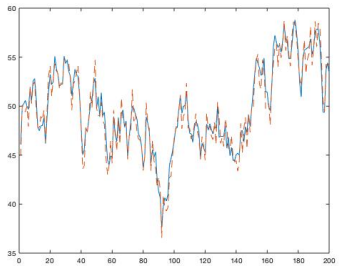

Figure 4.33: 14th Configuration: Drone 3 position estimation, x-axis.

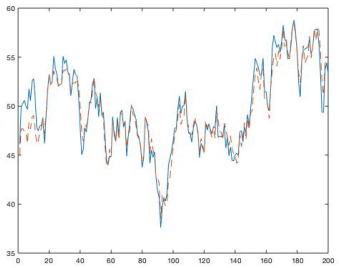

Figure 4.34: 14th Configuration: Drone 3 position estimation, y-axis.

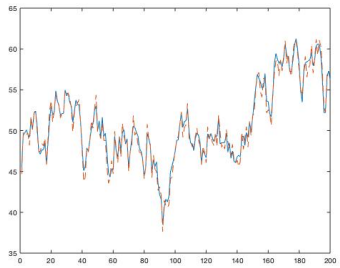

Figure 4.35: 14th Configuration: Drone 3 position estimation, z-axis.

Table 4.9 displays the root mean square deviation of Drone 3 in all three configurations.

Configuration:	$x-axis$ $[m]$	$y-axis$ $[m]$	$z-axis$ $[m]$
12	51.8973	46.7386	28.5462
13	7.2012	7.9761	7.9149
14	0.6731	0.8565	0.6207

Table 4.9: 12th, 13th and 14th Configuration - RMS Deviation Table of Drone 3.

EKF - 3D Positioning and Orientation

This configuration is a three-dimensional implementation with orientation. It is a fusion of the last two subsections.

As in the 2D configuration with orientation the transformation between the inertial and local referential must be taken in account.

From this moment on, each local referential, $\{V_i\}$, $i \in [1,2,3]$, sympathetic with the drone's kinematic, is not only defined by its position, ${}^{I}p_i$, $i \in [1,2,3]$, but also, its orientation, $[\phi_i, \theta_i, \psi_i]$, $i \in [1,2,3]$, in an inertial referential, $\{I\}$.

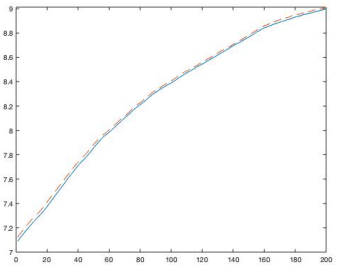

Figure 4.36: 15th Configuration: Drone 3 orientation estimation, ψ.

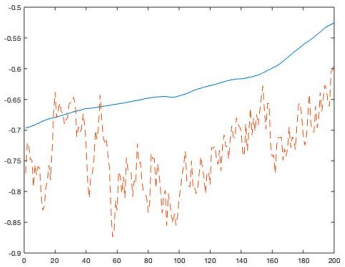

Figure 4.37: 15th Configuration: Drone 3 orientation estimation, θ.

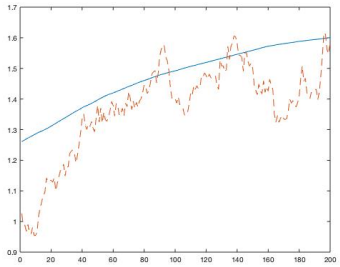

Figure 4.38: 15th Configuration: Drone 3 orientation estimation, ϕ.

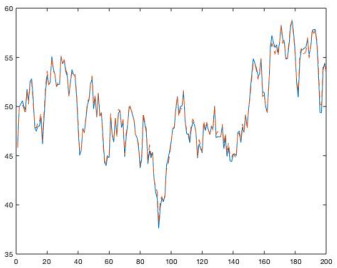

Figure 4.39: 15th Configuration: Drone 3 position estimation, x-axis.

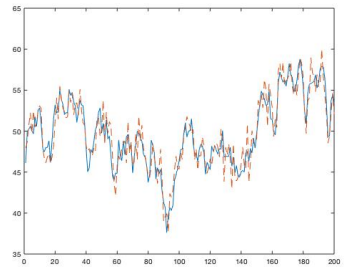

Figure 4.40: 15th Configuration: Drone 3 position estimation, y-axis.

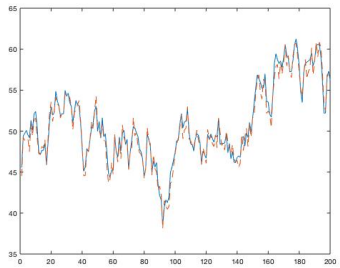

Figure 4.41: 15th Configuration: Drone 3 position estimation, z-axis.

Table 4.9 displays the root mean square deviation of Drone 3 in the fifteenth configuration and in all three configurations from the previous subsection: Configurations number 12, 13 and 14.

Configuration:	$x-axis$ $[m]$	$y-axis$ $[m]$	$z-axis$ $[m]$
12	51.8973	46.7386	28.5462
13	7.2012	7.9761	7.9149
14	0.6731	0.8565	0.6207
15	0.8114	1.5697	1.0639

Table 4.10: 15th Configuration - RMS Deviation Table of Drone 3.

Non-linear Filter - 3D Positioning and Orientation

This configuration is a three-dimensional implementation with orientation.

As in the previous configuration, each local referential, $\{V_i\}$, $i \in [1,2,3]$, sympathetic with the drone's kinematic, is not only defined by its position, ${}^{I}p_i$, $i \in [1,2,3]$, but also, its orientation, $[\phi_i, \theta_i, \psi_i]$, $i \in [1,2,3]$, in an inertial referential, $\{I\}$.

The plots show the real position in blue and the filter's optimal position estimation in orange.

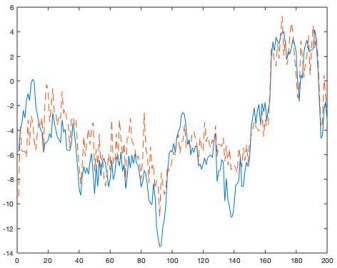

Figure 4.42: Non-linear Filter:Drone 3 position estimation, x-axis.

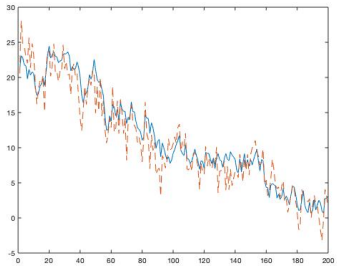

Figure 4.43: Non-linear Filter:Drone 3 position estimation, y-axis.

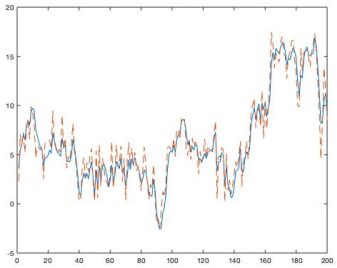

Figure 4.44: Non-linear Filter: Drone 3 position estimation, z-axis.

Table 4.9 displays the root mean square deviation of Drone 3 in the fifteenth and sixteenth configurations.

Configuration:	$x-axis$ $[m]$	$y-axis$ $[m]$	$z-axis$ $[m]$
EKF	0.8114	1.5697	1.0639
Non-linear	2.4023	2.3082	1.2903

Table 4.11: 15th Configuration - RMS Deviation Table of Drone 3.

4.0.1 Overall Results

The final configuration is a result of a series of implementations and simulations. During this process the deviation between the real position and the optimal estimate suffers alterations. Two plots were made of the deviation between the real position and the optimal estimate of Drone 1, a drone with a GPS receiver, and of Drone 3, a drone without one.

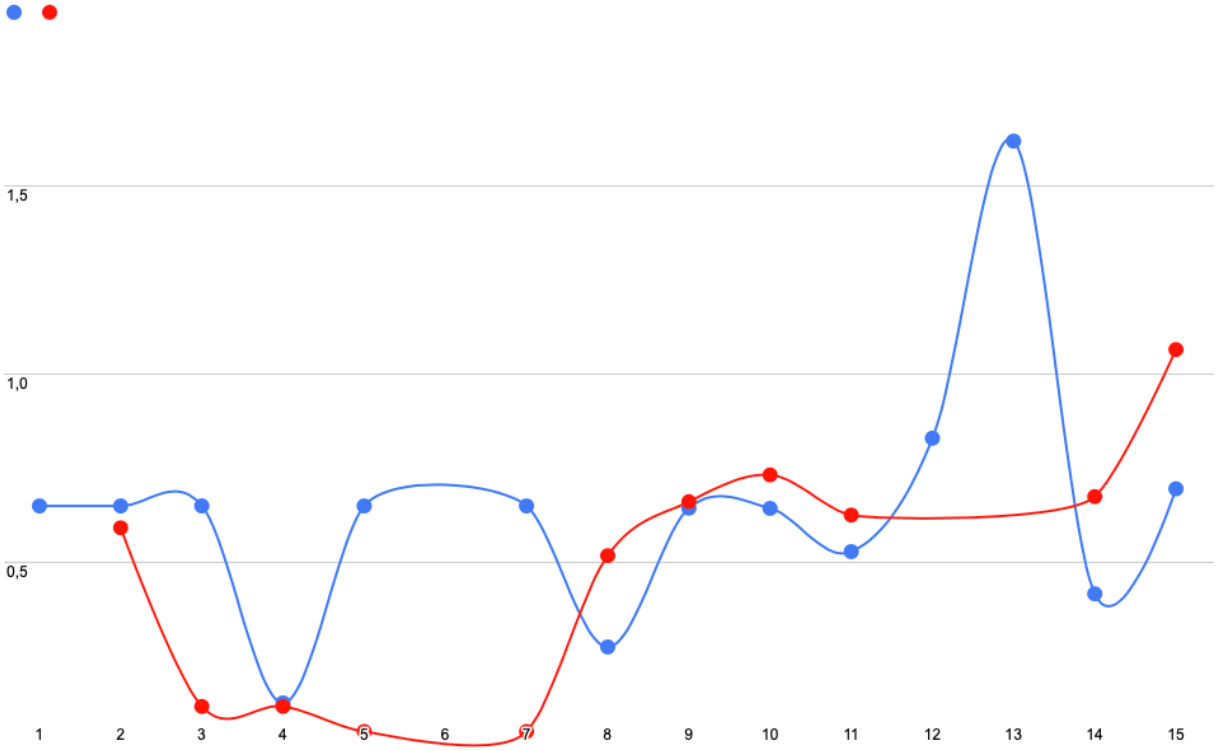

Figure 4.45: Drone 1: Blue; Drone 3: Red; RMS deviation [m] per configuration.

The blue plot portrays the evolution of Drone's 1 deviation as a function of the configuration. As seen in Table 5.1 and Figure 4.45, the deviation stays the same through the first three configurations as the simulation of Drone 1 has no changes. The simulation is a fusion between the model and the GPS data.

In red is the evolution of Drone's 3 deviation as a function of the configuration. In the first configuration there was no Drone 3. In the second configuration the filter's estimate is a fusion between the model and its relative position to the anchor, Drone 1. In the third configuration, Drone 2 was added, to provide another anchor point to Drone 3. This new anchor in combination with Drone 1, led to a huge decrease in the deviation between the real and the optimal estimation of Drone's 3 position.

With the introduction of another drone, Drone 2, with a sensor whose output is its relative position to Drone 1, in configuration number 4, the filter's estimate is now a fusion of the GPS data, the model and relative position to another drone. With more data, the optimal estimate is closer to the real position, as seen in the results of the fourth configuration.

The introduction of Drone 2 does not only affect the RMS deviation of Drone 1 but also Drone 3. Therefore, since the third configuration, all simulations have three drones, all aware of their relative position to the others.

The next trio of configurations is an attempt to use already existing AI vision based algorithms to obtain the relative position between drones. As the vector between drones, $\vec{d_{i,j}}$, is not an existing one, a non-linear output had to be implemented. Bearing that in mind three approaches were tested. The first was to use the distance between drones, $||\vec{d_{i,j}}||$, the second was to obtain the angle between two aircraft, $\alpha_{i,j}$ and the third was a concatenation of the two.

The distance between drones, fifth configuration, increased the deviation of Drone 1, blue, as the filter had to obtain two coordinates, (x, y), with one measure, $||\vec{d_{i,j}}||$, instead of the previous two, $\vec{d_{i,j}}$. But, surprisingly, decreased the deviation in the estimation of Drone's 3 position, red.

The angle between drones, sixth configuration, could not converge, therefore the deviation is not shown in Figure 4.45.

The combination of both, seventh configuration, due to the error of the angle sensor, is as good as the distance between drones, alone. Bearing this in mind, the angle approach had to be tested again.

With the goal of testing an angle based output, the states of the simulation, and filter were swapped in the next configurations. The drones are no longer independent identities but rather part of a collective simulation and filtering process, it is a centralised approach.

With the states going from position, $\vec{p_i}$, of the three drones to position of Drone 1, $\vec{p_1}$, and relative positions, $\vec{d_{1,3}}$ and $\vec{d_{3,2}}$, plus the fact that the process noise stayed the same, is expected that the deviation would increase, as it did with Drone 3, in red, but that did not happen with Drone 1, in blue. As direct opposition to the results of the fifth configuration.

Instead, the deviation of Drone 1, blue, decreased with the implementation of the distance between drones in this centralised approach, eighth configuration. The ninth configuration, the normalised vector between drones, $\frac{\vec{d_{i,j}}}{||\vec{d_{i,j}}||}$ increased slightly the deviation in comparison to the distance vector. And the combination of both, configuration number ten, was just slightly better than the ninth configuration, but these results must be studied in three dimensions, as well.

The same simulation, but now in three dimensions, (x, y, z), has different results to the same outputs as configurations number eight, nine and ten. In 3D, the best choice is the concatenation of both the distance and normalised vector between drones. The values of configuration number twelve and thirteen were suppressed from the red plot as its values were outliers.

The introduction of orientation in two dimensions, configuration eleven, has results paradoxically opposite to that imagined, as more states must be estimated and the deviation decreases. This contrasts with the implementation of orientation in three dimensions, configuration fifteen, which increased its deviation in comparison with null orientation, configuration fourteen.

Chapter 5

Conclusions

As seen by the results in table 5.1, the 16th configuration was not an improvement regarding the RMS deviation. That was already expected once the 16th configurations only uses the GPS and the angle sensor to estimate the position of the drones and the 15th configuration uses those sensors plus the distance sensor. The 6th configuration has as output the angle sensor (α) which was not able to converge, and therefore, the results are disproportionate. The 12th configuration was in 3D and has as output only the distance sensor, and due to the lack of information the algorithm wasn't able to identify the positioning of the drone.

	Drone 1			Drone 3		
	x	y	z	x	y	z
Configuration:						
1	0.6484	0.7594	-	-	-	-
2	0.6484	0.7594	-	0.5904	0.6903	-
3	0.6484	0.7594	-	0.1156	0.1633	-
4	0.1251	0.1787	-	0.1156	0.1633	-
5	0.6484	0.7594	-	0.9453	0.0492	-
6	55.455	58.682	-	91.263	40.221	-
7	0.6484	0.7594	-	0.9444	0.0494	-
8	0.3488	0.2737	-	0.5163	1.0386	-
9	0.6425	0.9274	-	0.6596	0.9294	-
10	0.6419	0.8403	-	0.7306	0.9777	-
11	0.5270	0.5309	-	0.6243	0.6504	-
12	0.8368	0.8282	0.5575	51.897	46.739	28.546
13	1.6177	2.7003	1.6159	7.2012	7.9761	7.9149
14	0.4065	0.4316	0.41495	0.6731	0.8565	0.6207
15	0.6938	0.7934	0.6415	0.8114	1.5697	1.0639
16	1.0741	1.0974	1.0742	2.4023	2.3082	1.2903

Table 5.1: RMS deviation per configuration table.

The introduction of the non-linear filter cannot be evaluated only based on the RMS deviation though. The need for less computing power and batteries due to the fewer sensors can be a plus in some

situations. In order to evaluate the results some context on the environment in which the drones operate is fundamental. Nevertheless, configurations 15 and 16 are the better suited for implementation.

Future work should include the further development of the non-linear filter technique. For instance, the generalisation of this filter for all the estimates of all drones' positions and not only Drone 3. Also, for further validation, these configurations should be tested using an UAV on the context of real time navigation.

Bibliography

[1] Clarity from above. PwC global report on the commercial applications of drone technology. Poland: PwC. Available at https://www.pwc.pl/pl/pdf/clarity-from-above-pwc.pdf.

[2] Confessions Of An Active Adopter: Kevin Costain. Available at https://bit.ly/3EtFkId.

[3] Ide-Flore Kenmogne, Vincent Drevelle, Eric Marchand - Cooperative Localization of Drones by using Interval Methods - Acta Cybernetica, University of Szeged, Institute of Informatics, 2019, pp.1-16. hal-02339451. Available at https://hal.inria.fr/hal-02339451/document.

[4] Kexin Guo, Zhirong Qiu, Wei Meng, Lihua Xie1 and Rodney Teo - Ultra-wideband based cooperative relative localization algorithm and experiments for multiple unmanned aerial vehicles in GPS denied environments. International Journal of Micro Air Vehicles [Em linha] vol. 9(3) 169–186, (2017), p.169-170. Available at https://journals.sagepub.com/doi/pdf/10.1177/1756829317695564.

[5] Marian Blachuta, Rafal Grygiel, Roman Czyba, Grzegorz Szafranski - Attitude and Heading Reference System Based on 3D Complementary Filter. Poland: Department of Automatic Control. Silesian University of Technology. Available at https://www.researchgate.net/publication/286800437_Attitude_and_heading_reference_system_based_on_3D_complementary_filter.

[6] Yen-Hsiang Huang, Yusie Rizal, Ming-Tzu Ho - Development of attitude and heading reference systems: 2015 International Automatic Control Conference (CACS). Available at https://www.researchgate.net/publication/304292219_Development_of_attitude_and_heading_reference_systems/references.

[7] Murty S. Challa, Jay G. Moore, Daniel J. Rogers - A Simple Attitude Unscented Kalman Filter: Theory and Evaluation in a Magnetometer-Only Spacecraft Scenario. Available at https://www.researchgate.net/publication/301716223_A_Simple_Attitude_Unscented_Kalman_Filter_Theory_and_Evaluation_in_a_Magnetometer-Only_Spacecraft_Scenario.

[8] Xiang Li, Chuan He, Yongjun Wang, Zhi Li - Generalized complementary filter for attitude estimation based on vector observations and cross products. Avilable at https://www.researchgate.net/publication/308836937_Generalized_complementary_filter_for_attitude_estimation_based_on_vector_observations_and_cross_products.

[9] Context Illustration available at https://www.bridgeweb.com/deck-solution-7423.

[10] Axis Rotation Matrices. Available at http://www.kwon3d.com/theory/transform/rot.html.

[11] Beard, Randal W., McLain, Timothy W. - Small Unmanned Aircraft: Theory and Practice: Princeton University Press, (2012).

[12] Matthew Leccadito - A Kalman Filter Based Attitude Heading Reference System Using a Low Cost Inertial Measurement Unit. Theses and Dissertations. Virginia Commonwealth University [Em linha] (2013) p.53 . Available at https://scholarscompass.vcu.edu/cgi/viewcontent.cgi?article=4188&context=etd

[13] Matthew Leccadito - A Kalman Filter Based Attitude Heading Reference System Using a Low Cost Inertial Measurement Unit. Theses and Dissertations. Virginia Commonwealth University [Em linha] (2013) p.57 . Available at https://scholarscompass.vcu.edu/cgi/viewcontent.cgi?article=4188&context=etd

[14] Lasse Klingbeil, Christian Eling, Florian Zimmermann, and Heiner Kuhlmann - Magnetic Field Sensor Calibration for Attitude Determination. Journal of Applied Geodesy 8(2):97 -108 (2014). Available at https://bit.ly/3vZSDNL.

[15] Satellite Navigation Branch, ANG-E66 NSTB/WAAS T&E Team - GLOBAL POSITIONING SYSTEM STANDARD POSITIONING SERVICE PERFORMANCE ANALYSIS REPORT. October 2020. Available at https://www.nstb.tc.faa.gov/reports/2020_Q3_SPS_PAN111_v1.0.pdf.

[16] Abdelkrim Belhaoua, Sophie Kohler and Ernest Hirsch - Error Evaluation in a Stereovision-Based 3D Reconstruction System.EURASIP Journal on Image and Video Processing (2010), p 8. Available at https://www.researchgate.net/publication/220539186_Error_Evaluation_in_a_Stereovision-Based_3D_Reconstruction_System.

[17] Furqan Asghar, Muhammad Talha, Sung Ho Kim and In-Ho Ra - Simulation Study on Battery State of Charge Estimation Using Kalman Filter. Available at https://www.researchgate.net/publication/311457802.

[18] Omkar Kabadagi - Understanding Kalman Filter and it's equations: Medium. Available at https://medium.com/team-rover/understanding-kalman-filter-and-its-equations-5fcc5d5fe61e.

[19] Rita Cunha - Sensors and State Estimation (additional notes).UAVs. MEAer - Spring Semester – 2019/2020. p 22.

[20] Rita Cunha - Sensors and State Estimation (additional notes).UAVs. MEAer - Spring Semester – 2019/2020. p 23.