UAV Pose Estimation using Sensor Fusion of SONAR and Inertial Sensors

B.J. Caasenbrood

D&C

Traineeship report (DC 2016.028.)

Coach(es):	prof. L. Wang
Supervisor:	prof. dr. H. Nijmeijer

Eindhoven University of Technology
Department of Mechanical Engineering
Dynamics & Control

Eindhoven, April, 2016

Abbreviations and symbols

Abbreviations

DCM	Direction-cosine Matrix
DOF	Degrees of Freedom
EKF	Extended Kalman Filter
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
PCB	Printed Circuit Board
SONAR	Sound Navigation and Ranging
RMS	Root Mean Square
UAV	Unmanned Aerial Vehicle
UKF	Unscented Kalman Filter

Notation

$\boldsymbol{x}$	Vector
$\boldsymbol{X}$	Matrix
$\boldsymbol{x}^T$	Transpose of vector $\boldsymbol{x}$
$\boldsymbol{X}^T$	Transpose of matrix $\boldsymbol{X}$
$\boldsymbol{X}^{-1}$	Inverse of matrix $\boldsymbol{X}$
$\bar{\boldsymbol{x}}$	Measurement vector
$\boldsymbol{\lambda}(\boldsymbol{X})$	Eigenvalues of matrix $\boldsymbol{X}$
$\text{rank}(\boldsymbol{X})$	Rank of matrix $\boldsymbol{X}$
$\text{diag}(\boldsymbol{x})$, $\text{diag}(\boldsymbol{X})$	Diagonal matrix with the elements of $\boldsymbol{x}$ or $\boldsymbol{X}$ as the main diagonal
$\text{COV}(\boldsymbol{x})$	Covariance matrix of vector $\boldsymbol{x}$

Symbols

γ	Beamwidth	Degree	$^\circ$
θ	Pitch angle	Radian	rad
f	Specific force	meter per second per second	$\text{m}\cdot\text{s}^{-2}$
m	Magnetic field strength	Gauss	G
ϕ	Roll angle	Radian	rad
ψ	Yaw angle	Radian	rad
S	Range	Meter	m
t	Time	Second	s
ω	Angular rate	Radian per second	$\text{rad}\cdot\text{s}^{-1}$

Contents

Abstract

This report presents a novel sensor fusion algorithm applicable to inertial measurement units (IMU's) and ultrasonic sensor arrays. The data fusion between the different sensors should improve the overall sensing system. The merit advantage of the approach is that it does not require a dynamic model of the UAV. For position measurements, the system deploys a setup consisting of four independent, active ultrasonic sensors. For motion measurements, the system deploys an IMU that measures the accelerations and the angular velocities of the UAV. Although the IMU has a good short-term precision, the sensor is susceptible to error accumulation due to drift.

An inertial navigation system (INS) has been employed to provide navigation information, such as position, velocity, and orientation, based on inertial sensors and navigation dynamics. Along the INS, an extended Kalman filter (EKF) is employed to filter and estimate the navigation states. In addition, the EKF provides a method to compensate the drift in the IMU. Unlike the EKF's linear counterpart, the EKF is not an optimal estimator and linearision might lead to an unstable filter if local linearity is violated. Nevertheless, the EKF should provide reasonable performance in this application.

The results indicate that the sensor fusion algorithm achieves levels of accuracy exceeding that of the sensor. For stationary measurements, the proposed EKF reduced the standard deviations in orientation estimates of the x-, y-, z-axis respectively to around 30.1%, 27.1%, and 58.7% of the corresponding measurements. The filter reduced standard deviations in position estimates of the x-, y-, and z-axis to around 6.49%, 7.45%, and 7.25% of the corresponding measurements. Although not quantifiable, the dynamics measurements show that the results are deemed accurate. However, additional rotor disturbance negatively impacts the system. The EKF reached poor absolute errors < 0.511 meter in position estimates and absolute errors $< 25.2°$ in orientation estimates.

In conclusion, proposed approach of fusing the data of inertial sensors and ultrasonic sensors can provide a system with accurate and precise navigation information. However, the practicality of the discussed sensors in combination with UAV platforms, which suffer from rotor disturbance, remains questionable.

1 Introduction

Remote controlled and autonomous unmanned air vehicles (UAV) have become immensely popular among researchers . UAV's can perform complex movements and they are very agile. UAV's posses the ability for vertical take-off, or hovering stationary mid-air. These abilities make UAV's suiting for many applications, such as accessing area that would be hazardous for humans. Moreover, they provide a low-cost platform for various embedded control application, e.g., auto-tuning control [1], swarm control [2], and optical flow sensors [3]. The UAV used in this report is an hexacopter, as can be seen in Figure 1.1.

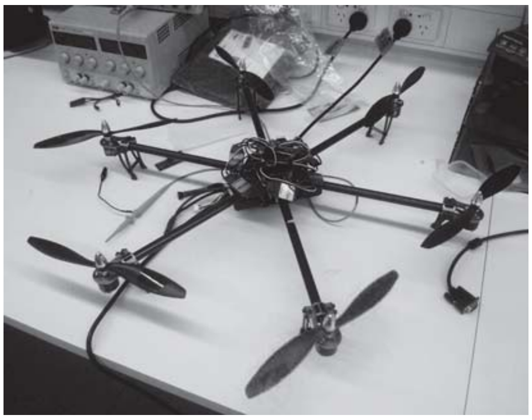


Figure 1.1: *Hexacopter UAV.*

In order to navigate and position through an unknown environment, UAV's rely heavily on sensors that provide information, like position, velocity, and orientation. A lot of research is also focused on the use of on-board cameras and laser rangefinders in navigation purposes [4],[5], however, these are often considered an expensive option. A more common sensor used for close-range detection are ultrasonic range finders.

Standard ultrasonic range finders used for robot applications are able to provide multiple distance measurements per second. Their angular resolution is usually one or two orders of magnitude worse than lasers ranging sensors, which have a sub-degree angular resolution [7]. Nonetheless, ultrasonic sensors are still appealing thanks to their low price and low power consumption.

Inertial measurement units (IMU's) are widely used in motion application. The prominent merits of IMU's, such as low cost, light weight, and allow direct measurement without influencing the system complexity, make them attractive in motion determinations for small moving objects. IMU's have a good short-term precision and high sampling rate, however, they suffer from serious errors in long-term position and orientation estimations due to drift, which is a major issue that limits the accuracy of inertial navigation. Hence, additional sensors are needed to complement the IMU's drift, e.g., GPS, or SONAR.

Nowadays, fusing data from different sensors to improve performance of the overall sensing system becomes necessary in many applications. For aerial navigation, fusion of ultrasonic sensor measurements with IMU measurements by means of filtering techniques is vital to deliver the level of localization precision required by UAV missions. Currently, the most used technique to fuse navigation data is the Kalman filter. In this case, an extended Kalman filter (EKF) is adopted. Although the extended Kalman filter is capable of providing real time vehicle position updates, it is based on linear system models and it suffers from linearization when dealing with nonlinear models.

In this report, we investigate the possibility for sensor fusion between an inertial measurement unit and an ultrasonic sensor array for UAV localization. Combining inertial and ultrasonic sensors is preferable, since they have shared characteristics and should therefor yield a robust and reliable system. Chapter 2 will provide an overview of the hardware used in report, along with some sensor dynamics. Chapter 3 will construct the system dynamics. Followed by Chapter 4, which will discuss sensor fusion. The experimental setup and results are assessed in Chapter 5. Lastly, Chapter 6 will provide a conclusion based on these finding, concluded by a brief recommendation for future work in Chapter 7.

2 Hardware overview

This chapter will provide information on the sensors used in the sensor fusion algorithm. Ultrasonic rangefinders, as inertial measurement units, are being discussed in this chapter. Moreover, the chapter will elaborate on the wave propagation profile of ultrasonic rangefinders, and the IMU's error dynamics.

2.1 SONAR

A Sound Navigation and Ranging sensors (SONAR) uses ultrasonic sound to detect and measure the range to an obstacle. An active SONAR utilize a sound transmitter and a receiver. The SONAR sensor creates a short sound pulse, then the receiver listens for the reflection (echo) of the pulse. The time interval between transmitting and receiving determines the distance between the SONAR sensor and the obstacle [6]. The prominent merits of active SONAR setups, such as low cost, and low power consumption, make them interesting for various UAV applications. Although ultrasonic sensors do not suffer from accumulated errors and are quite accurate in position determination, the sampling rate of ultrasonic sensors is rather low (30-50 Hz) due to the traveling distance of the sound pulse. In addition, the sampling rate of the SONAR is also influenced by the temperature of the medium in which the sound pulse propagates.

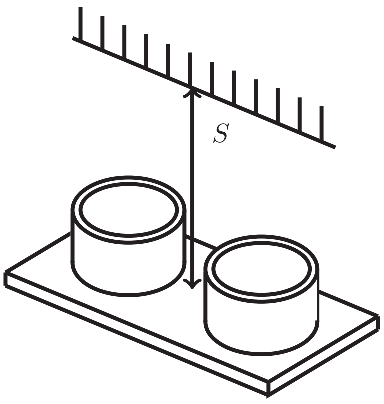
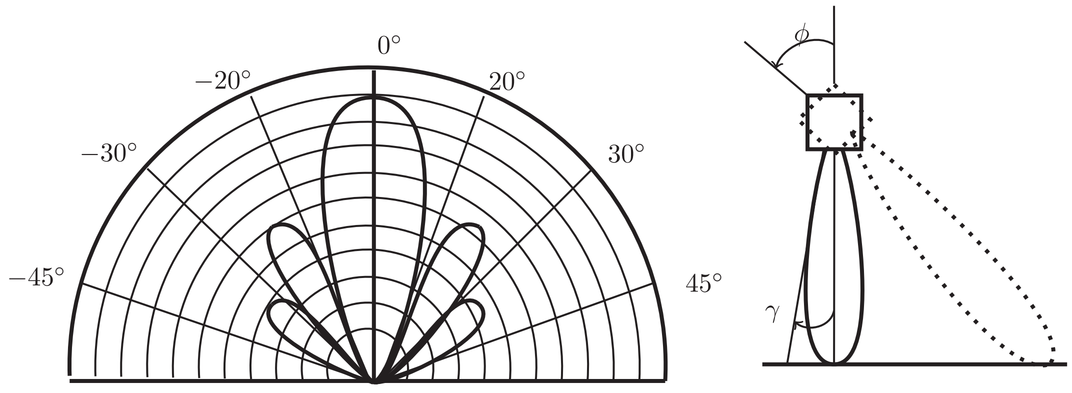


Figure 2.1: *Sonar measurement frame*

The received echo signal $r(t)$ can be related to the range S of the object as $r(t) = h(t) * s(t - 2S/c)$, where $*$ is denoted as a convolution, $h(t)$ the dynamic distortion of the signal, $s(t)$ the transmitted signal, and c the speed of sound. The range is defined as the radial distance between the SONAR and the object. Alternatively, the range can be computed with $S = \frac{1}{2}\tau c$, where the τ is the timespan between transmitting and receiving the sound pulse. Although the speed of sound c is dependent on the temperature in the air, it is assumed constant at room temperature ($20°C$). The ultrasonic rangefinder used is a HC-SR04. The sensor payload of the UAV is comprised of four ultrasonic rangefinders that each detect a range $\bar{\boldsymbol{S}} = (S_1, S_2, S_3, S_4)^T$ in units of meters (m). The orientation of these ultrasonic rangefinders will be elaborated further in Chapter 2.4.

2.2 SONAR characterization

One key characteristic of ultrasonic sensors is the propagation wave profile, also called the radiation area. An illustration of the propagation wave profile is shown in Figure 2.2(a). As can be seen, the propagation pattern consist of multiple detection lobes, which essentially define the angular ranges between the normal direction of the ultrasonic sensor [7]. The detection lobes of an ultrasonic ranging device is of primary importance in almost any application of such measuring systems, since these determine whether the measuring system is suitable for the application.



(a) Illustration of the propagation pattern. (b) Tilted measurement

Figure 2.2: *Illustration of lobes pattern, and illustration of influence of tilted SONAR measurement.*

As shown in Figure 2.2(a), the primary lobe covers an angulare range of $\pm 20°$, the secondary lobe is $\pm 30°$ wide, and the tertiary lobe is $\pm 45°$ wide [7]. The distinct propagation patterns arise from internal interference of the sound waves. The angle of the primary lobe is referred to as the beamwidth γ expressed in units of degree(°). For map-building tasks, a large beamwidth is undesirable since it increases the uncertainty about the actual location of an obstacle. According to the documentation of the HC-SR04, the beamwidth of the unit is $\gamma \approx 21.5°$ [8].

In this UAV application, the characterization of the primary lobe is of the very essence in range determination, since sensor units will likely never measure walls and floors with an angle of incidence $\phi = 0$. Thus, to ensure that the echo is detectable by the sound receiver, the angle of incidence must remains within the bounded beamwidth $\phi \leq \gamma$, as illustrated in Figure 2.2(b). Additionally, this ensures that the measured range of an object is truly the shortest distance between sensor and obstacle, and not a secondary reflection from different objects.

2.3 Inertial Measurement Unit

Used primarily as a navigation tool, an inertial measurement unit (IMU) is a broad term used to describe a package of electronic sensors. A 9 DOF IMU consist of a tri-axis accelerometer, a tri-axis gyroscope, and a tri-axis magnetometer. The gyroscope and accelerometer are the inertial sensors and measure the specific force (m/s^2) and angular velocity (rad/s), respectively. The magnetometer is used to measure the component of the earth's magnetic field in units of Gauss (G). The combination of the accelerometer and the magnetometer is referred to as the digital compass of the IMU. An IMU usually provides fast rate (100 - 1000 Hz) information about linear acceleration and angular velocities. The axes of all the electronic sensors are ideally orthogonal and aligned with the axes of the platform they are mounted on, as shown in Figure 2.3. The IMU's accelerations are depicted as $\bar{\boldsymbol{f}} = (f_x^b,\, f_y^b,\, f_z^b)^T$, the angular velocities as $\bar{\boldsymbol{\omega}} = (\omega_x^b,\, \omega_y^b,\, \omega_z^b)^T$ and the magnetic fields measurements $\bar{\boldsymbol{m}} = (m_x^b,\, m_y^b,\, m_z^b)^T$.

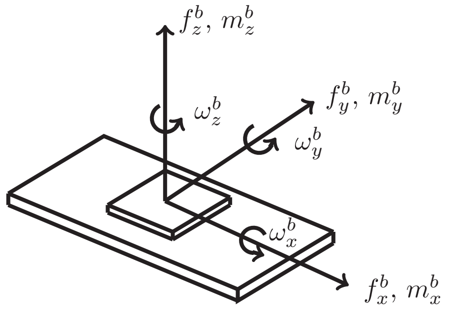


Figure 2.3: *IMU measurement frame*

Like any measurement device, data from the IMU could suffer from measurement errors due to several reasons, e.g., temperature, environment disturbances, or sensor defects. Unfortunately, such influences cannot be modeled and compensated, however, some measurement errors can be compensated with a proper model. The IMU's noise signals can be differentiated into two different types of errors: an additive noise term $\boldsymbol{n}(t)$, and a slow varying sensor bias $\boldsymbol{\delta}(t)$ [9]. Therefore, the general expression for the accelerometer, gyroscope, and magnetometer measurements

$$\begin{aligned} \bar{\boldsymbol{f}} &= \boldsymbol{f}^b + \boldsymbol{\delta f} + \boldsymbol{n}_f \\ \bar{\boldsymbol{\omega}} &= \boldsymbol{\omega}^b + \boldsymbol{\delta\omega} + \boldsymbol{n}_{\boldsymbol{\omega}}, \\ \bar{\boldsymbol{m}} &= \boldsymbol{m}^b + \boldsymbol{\delta m} + \boldsymbol{n}_{\boldsymbol{m}}, \end{aligned} \tag{2.1}$$

where the noise terms $\boldsymbol{n}$ are modeled as a zero-mean, white Gaussian noise process, $\boldsymbol{f}^b$, $\boldsymbol{\omega}^b$, and $\boldsymbol{m}^b$ are the true measurements. Unlike the accelerometers and gyroscopes, magnetometers are less susceptible to drift. Thus, the time-varying bias is not taken in account for the magnetometer. The accelerometer's and gyroscope's slowing varying sensor biases are modeled as a Brownian motion process, also referred to as a 'random walk' in discrete-time [10]. Therefore,

$$\begin{aligned} \boldsymbol{\delta}\dot{\boldsymbol{f}} &= \boldsymbol{\delta f}_0 + \boldsymbol{n}_{\boldsymbol{\delta f}} \\ \boldsymbol{\delta}\dot{\boldsymbol{\omega}} &= \boldsymbol{\delta\omega}_0 + \boldsymbol{n}_{\boldsymbol{\delta\omega}}, \end{aligned} \tag{2.2}$$

where $\boldsymbol{\delta}_0$ is the constant null shift, and the noise terms $\boldsymbol{n}_{\boldsymbol{\delta}}$ are modeled as a zero-mean, white Gaussian noise process. The constant null shift $\boldsymbol{\delta}_0$ is referred to as the drift rate, which defines the growth of the sensor bias in time. The accelerometer's and gyroscope's biases growing trend, respectively $\boldsymbol{\delta f}$ and $\boldsymbol{\delta\omega}$, is close to linear in time.

2.4 Physical system

The flight platform is a hexacopter drone and is capable of flying 20 m/s and upto 100 m. The payload sensors are part of the flight platform. The payload sensors comprise of an digital compass, a gyroscope, and four ultrasonic range finders. The ultrasonic rangefinders are aligned ideally orthogonal, such that the position of the object can be calculated using distance measurements. The on-board sensors can only communicate if there is a microcontroller reading the sensor values and processing the data signals. Thus, a custom printer circuit board (PCB) was manufactured compatible with an Arduino Nano for signal processing. The Arduino Nano is equipped with an ATmega328 mircocontroller. Figure 2.4 shows the payload sensor package, including the sensor body reference frames. The circuit schematic have been provided in Appendix A.3.

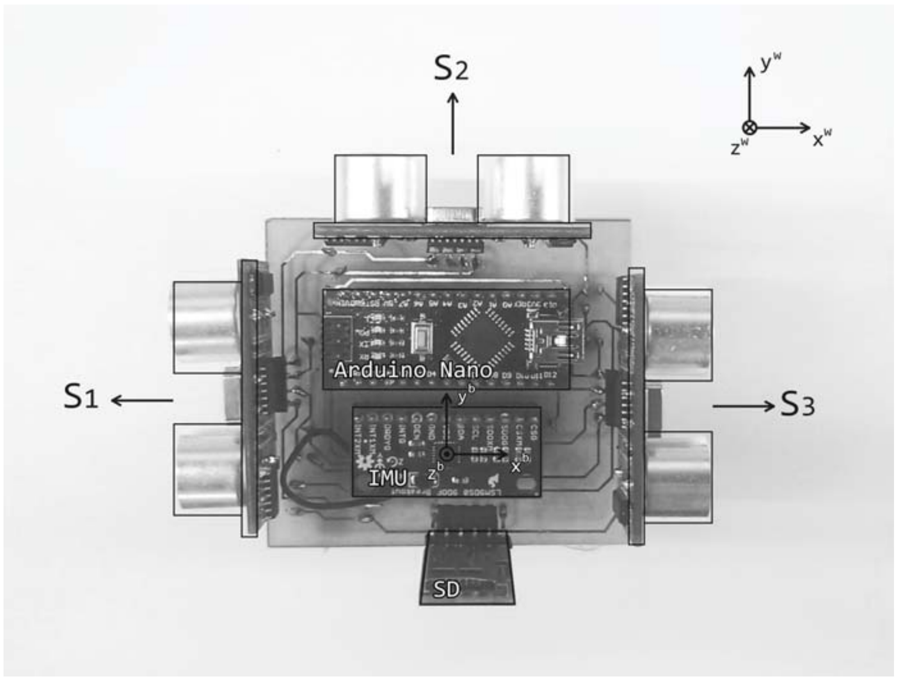


Figure 2.4: *Payload sensor package comprised of an IMU, Arduino Nano, SD-card reader and four ultrasonic sensors.*

As shown in Figure 2.4, the ultrasonic sonar measurement are defined as $\bar{\boldsymbol{S}} = (S_1,\, S_2,\, S_3,\, S_4)^T$, where S_4 is not shown but it is oriented in the opposite direction of the z^b-axis of the bodyframe. The microcontroller controls the ultrasonic emitter through pulsewidth modulation. The microcontroller can reach a sampling frequency of 100 Hz for the IMU signals, and a sampling frequencies varying between 20 - 35 Hz for the SONAR signals. The C-program running on the Arduino Nano processes the signals, and stores them onto a SD-card. Unfortunately, the dynamic memory of the mircocontroller (2KB) is too small to perform an on-board sensor fusion algorithm. The sensor fusion algorithm is currently performed in MATLAB, and the sensor fusion code has been included in Appendix A.4.

3 Navigation dynamics

In this chapter, the system orientations depicted in Euler angles will be discussed. Moreover, the chapter will provide detail on deduction of the Euler angles from the measurements of the inertial measurement unit. Lastly, the equations for the inertial navigation system will be formulated.

3.1 System Orientation

The aim of the report is to determine the position and orientation of the UAV w.r.t. a fixed world frame, indicated with the label w, as depicted in Figure 3.1. The body frame of the UAV is denoted with the label b. The axis of the sensor body frame are aligned parallel with the fixed body frame. The coordinate system is defined as Cartesian, where the rotation around the x^w, y^w and z^w-axis are respectively denoted as ϕ (roll), θ (pitch) and ψ (yaw). The position and rotation vectors are respectively expressed as $\boldsymbol{p}^w = (x,\, y,\, z)^T$ in units of meters (m) and $\boldsymbol{\theta} = (\phi,\, \theta,\, \psi)^T$ in units of radians (rad).

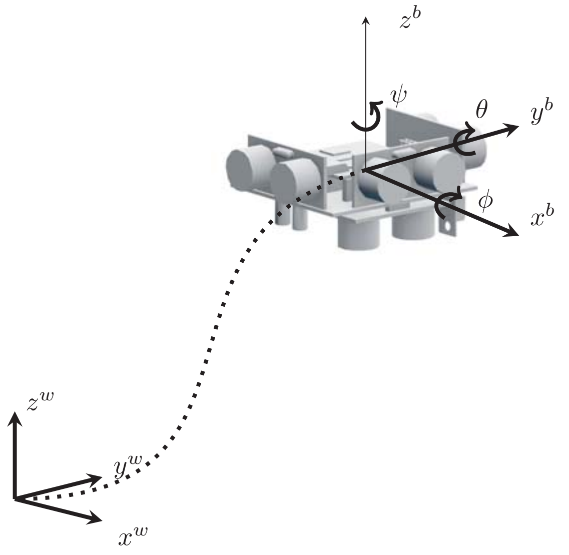


Figure 3.1: *Depiction of the world frame and body frame*

The gravitational vectors and the Earth's magnetic field can be obtained by using the coordinate transformation from the body frame. In static state, the roll and pitch can be calculated using coordinate transformation of the gravitational vectors measured with the accelerometer. The direction cosine matrix $\boldsymbol{R}_b^w$ translates vector from the body frame b to the world frame w, and is expressed as

$$\boldsymbol{R}_b^w = \begin{bmatrix} c\theta\, c\psi & c\theta\, s\psi & -s\theta \\ s\phi\, s\theta\, c\psi - c\phi\, s\psi & s\phi\, s\theta\, s\psi + c\phi\, c\psi & s\phi\, c\theta \\ c\phi\, s\theta\, c\psi + s\phi\, s\psi & c\phi\, s\theta\, s\psi - s\phi\, c\psi & c\phi\, c\theta \end{bmatrix}, \tag{3.1}$$

where $c\phi$ is defined as $cos(\phi)$, and $s\phi$ is defined as $sin(\phi)$. An useful property of the DCM is that the transposed of the DCM $\boldsymbol{R}_b^w = \left(\boldsymbol{R}_w^b\right)^T$. The expression for the acceleration $\boldsymbol{f}^w$ and magnetic field $\boldsymbol{m}^w$ expressed in the world frame

$$\boldsymbol{g}^w = \boldsymbol{R}_b^w \boldsymbol{f}^b \tag{3.2}$$
$$\boldsymbol{m}^w = \boldsymbol{R}_b^w \boldsymbol{m}^b. \tag{3.3}$$

Since, the gravitational acceleration g and the Earth's magnetic field B are constant vectors, we are able express them w.r.t. the world frame. Therefore,

$$\boldsymbol{g}^w = \begin{pmatrix} 0 & 0 & -g \end{pmatrix}^T \tag{3.4}$$
$$\boldsymbol{m}^w = \begin{pmatrix} B_x & 0 & B_z \end{pmatrix}^T \tag{3.5}$$

In static state, all three gravity components and magnetic field components can be measured using the IMU. The gravity at the current state can be obtained from the initial state using a coordinate transformation with a DCM that contains the Euler angles. By correlating the measured values of the current state to the initial state using coordinate transformation, the Euler angles can be solved. From Eq. (3.2), the inverse transformation of the gravity is

$$\boldsymbol{f}^b = \boldsymbol{R}_w^b \, \boldsymbol{g}^w \tag{3.6}$$
$$\begin{pmatrix} f_x^b & f_y^b & f_z^b \end{pmatrix}^T = \begin{pmatrix} -s\theta & s\phi \, c\theta & c\phi \, c\theta \end{pmatrix}^T (-g), \tag{3.7}$$

Therefore, the roll ϕ and pitch θ can be determined

$$\phi = atan \left[f_y^b, \, f_z^b \right] \tag{3.8}$$
$$\theta = atan \left[-f_x^b, \, f_y^b \, s\phi + f_z^b \, c\phi \right] \tag{3.9}$$

The yaw ψ can be deduced from the magnetometer. The measured magnetic field of the current state is correlated to the initial state, in order to determine the yaw angle. From Eq. (3.3), the inverse transformation of the magnetic field is

$$\boldsymbol{m}^b = \boldsymbol{R}_w^b \, \boldsymbol{m}^w \tag{3.10}$$

Therefore, expressing of the yaw angle

$$\psi = atan \left[m_z^b \, s\phi - m_y^b \, c\phi, m_x^b \, c\theta + m_y^b \, s\theta \, s\phi - m_z^b \, s\theta \, c\phi \right] \tag{3.11}$$

Eq. (3.11) allow solutions for the yaw angle based on the magnetic field of the earth. Unfortunately, Eq. (3.8) and Eq. (3.9) have an infinite number of solutions at multiples of 2π. The loss of one DOF in a three-dimensional mechanism that occurs when the two axes are driven into a parallel configuration, is referred to as gimbal lock [11]. To ensure an unique solution for the orientation angles, a restriction is imposed on θ that limits the range $-\frac{\pi}{2}$ to $\frac{\pi}{2}$. A further constraint is imposed on ϕ and ψ that limits their range $-\pi$ to π. These imposed angular boundary should not impose any limitation on the UAV flight capabilities, since UAV's commonly hover with a planar attitude and they only make small inclinations when in motion.

3.2 Inertial Navigation System

Prior to the availability of GPS, inertial navigation system (INS) provided reasonable accuracy for navigation and found use as a passive position sensor for both military and commercial aircraft. An inertial navigation system is a system that provides navigation information based on inertial sensors and navigation dynamics. Although GPS provides good long-term measurements of position, it can introduce discontinuities, whereas, an INS has good short-term accuracy but accumulates drift errors as a consequence of the method of position and attitude determination. By integrating the INS with additional sensors, such as SONAR, the information can be used to realign the INS, bounding the drift errors. The navigation requires accelerometers and a rotation sensor to continuously calculate the position, velocity and orientation of the system [12]. The IMU will measure the specific forces $\boldsymbol{f}^b$ and angular velocity $\boldsymbol{\omega}^b$ w.r.t. the fixed body frame b. Then, these measurements can utilized to describe the motion in the world frame relative to the fixed body frame.

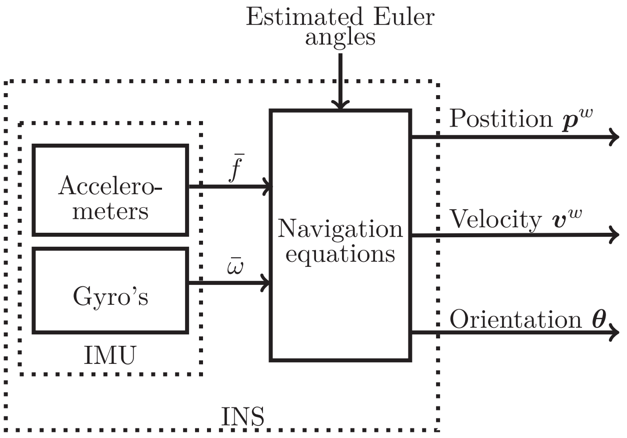


Figure 3.2: *Inertial navigation diagram*

Let the position and velocity w.r.t. the world frame w be depicted as $\boldsymbol{p}^w \in \mathbb{R}^3$ and $\boldsymbol{v}^w \in \mathbb{R}^3$, respectively. Whereas, the orientation of the UAV is parameterized by the Euler angles $\boldsymbol{\theta}^w \in \mathbb{R}^3$, the rotation matrix $\boldsymbol{R}_b^w(\theta) \in \mathbb{R}^{3\times3}$, and transformation matrix between Euler and body angular velocities $\boldsymbol{C}_b^w(\theta) \in \mathbb{R}^{3\times3}$. The expression for the matrix $\boldsymbol{C}_b^w$ can be found in Appendix A.1. For ideal IMU measurements, the navigation equations are given by

$$\begin{aligned}\dot{\boldsymbol{p}}^w &= \boldsymbol{v}^w \\ \dot{\boldsymbol{v}}^w &= \boldsymbol{R}_b^w \boldsymbol{f}^b + \boldsymbol{g}^w \\ \dot{\boldsymbol{\theta}} &= \boldsymbol{C}_b^w \boldsymbol{\omega}^b.\end{aligned} \tag{3.12}$$

Note that Eq. (3.12) only applies if the IMU measurements are ideal. However, the IMU measurements are baised and contain additive noise. Therefore, the substituting the ideal values of the IMU measurement in Eq. (3.12), with the IMU sensor model found in Chapter 2.3.

$$\begin{aligned}\dot{\boldsymbol{p}}^w &= \boldsymbol{v}^w \\ \dot{\boldsymbol{v}}^w &= \boldsymbol{R}_b^w(\boldsymbol{f}^b + \boldsymbol{\delta f} + \boldsymbol{n_f}) + \boldsymbol{g}^w \\ \dot{\boldsymbol{\theta}} &= \boldsymbol{C}_b^w \left(\boldsymbol{\omega}^b + \boldsymbol{\delta\omega} + \boldsymbol{n_\omega}\right),\end{aligned} \tag{3.13}$$

which describes the estimate of the navigation states containing deviations from the true state.

4 Sensor Fusion

This chapter provides knowledge and implementation of sensor fusion, and how it can improve the pose estimation of an UAV equipped with an IMU and ultrasonic sensor array. The chapter also will discuss the extended Kalman filter and the discrete time system, which includes a discrete observation and system model. Lastly, the chapter will provide detail on the observability of a linear time-vary system.

4.1 Extended Kalman Filter

Filtering and estimation are two of the most pervasive tools of engineering. Whenever the state of a system must be estimated from noisy sensor information, a state estimator is employed to fuse the data from different sensors together to produce an accurate estimate of the true system state. For such sensor data fusion, a classic approach to the state estimate problems for a nonlinear stochastic system is an extended Kalman filter (EKF). An extended Kalman filter is a first-order approximation solution to non-linear systems. The EKF applies the Kalman filter to nonlinear systems by simply linearizing all the nonlinear models along a trajectory, so that the traditional linear Kalman filter equations can be applied. Unfortunately, the EKF suffers from some drawbacks. Unlike the traditional Kalman filter, the extended Kalman filter is not an optimal estimator. In addition, if the initial estimates of the system are poor, or the process is modeled incorrectly, the system might quickly diverge [15]. Nevertheless, an extended Kalman filter can still provide a reasonable performance, and is the preferred filter for many navigation systems and GPS applications [16],[17],[18].

For the non-linear behavior, a generic stochastic process can be described by a discrete system

$$\begin{aligned} \boldsymbol{x}_{k|k-1} &= \boldsymbol{f}(\boldsymbol{x}_{k-1|k-1}, \boldsymbol{u}_k, \boldsymbol{v}_k) \qquad k \geq 0 && (4.1)\\ \boldsymbol{y}_k &= \boldsymbol{h}(\boldsymbol{x}_{k,k-1}, \boldsymbol{w}_k) \qquad k \geq 0 && (4.2)\end{aligned}$$

where $\boldsymbol{f}(\cdot)$ and $\boldsymbol{h}(\cdot)$ are respectively the state and observation functions, $\boldsymbol{x}_k$ and $\boldsymbol{y}_k$ are the state and observation vectors, $\boldsymbol{u}_k$ the control input, $\boldsymbol{v}_k$ and $\boldsymbol{w}_k$ are white Gaussian noise and the measurement noise, where both have different covariance matrices $COV(\boldsymbol{v}) = \boldsymbol{Q}_k$ and $COV(\boldsymbol{w}) = \boldsymbol{R}_k$. At each timestep the Jacobian is evaluated with current predicted states, essentially linearizes the non-linear function around the current estimate. These matrices can be used in the Kalman filter equations. Thus,

$$\boldsymbol{A} = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}_{k-1|k-1}} \qquad \boldsymbol{G} = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{v}}\right|_{\boldsymbol{x}_{k-1|k-1}} \qquad \boldsymbol{H} = \left.\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}_{k|k-1}}. \tag{4.3}$$

Since in this case the control input is zero, the Jacobian of $\boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{v}_k)$ w.r.t. the control input $\boldsymbol{u}_k$ is ignored. Note that the state $\boldsymbol{x}_{k|k}$ is the posteriori state. Now, the discrete time linear system can be described with the state space model

$$\begin{aligned} \boldsymbol{x}_{k|k-1} &= \boldsymbol{A}_k\, \boldsymbol{x}_{k-1,k-1} + \boldsymbol{G}_k\, \boldsymbol{v}_k \qquad k \geq 0 && (4.4)\\ \boldsymbol{y}_k &= \boldsymbol{H}_k\, \boldsymbol{x}_{k,k-1} + \boldsymbol{w}_k \qquad k \geq 0 && (4.5)\end{aligned}$$

Assuming, that the initial estimate $\boldsymbol{x}_{0|-1}$ of the state and covariance $\boldsymbol{P}_{0|-1} = COV(\mathbf{x}_0)$ is known, the Kalman filter then provides an estimate $\boldsymbol{x}_{k|k}$ at time k, given the measurement $\boldsymbol{y}_k$ up to time k. The

Kalman filter algorithm can be separated into two different parts. First, in the Time update process model and noise properties are utilized to estimate the state propagation $\boldsymbol{x}_{k|k-1}$ and its covariance $\boldsymbol{P}_{k|k-1}$ in the next time step k. Second, in the Measurement update the state $\boldsymbol{x}_{k|k}$ and covariance $\boldsymbol{P}_{k|k}$ are computed, based on the previous estimations of $\boldsymbol{x}_{k|k-1}$ and $\boldsymbol{P}_{k|k-1}$. The algorithm below shows the process.

Algorithm 4.1: Extended Kalman Filter

```
1 Initializations of $\boldsymbol{x}_{0|-1}$ and covariance matrix $\boldsymbol{P}_{0|-1}$
2 Time update:
        $\boldsymbol{x}_{k|k-1} = \boldsymbol{f}(\boldsymbol{x}_{k-1|k-1}, \boldsymbol{u}_k, \boldsymbol{v}_k)$
        $\boldsymbol{P}_{k|k-1} = \boldsymbol{A}_k\, \boldsymbol{P}_{k-1|k-1}\, \boldsymbol{A}_k^T + \boldsymbol{G}_k\, \boldsymbol{Q}_k\, \boldsymbol{G}_k^T$
3 Measurement update:
        $\boldsymbol{x}_{k|k} = \boldsymbol{x}_{k|k-1} + \boldsymbol{K}_k\, \epsilon_k$
        $\boldsymbol{P}_{k|k} = [\boldsymbol{I} - \boldsymbol{K}_k\, \boldsymbol{H}_k]\, \boldsymbol{P}_{k|k-1}$
4 Next time step $k = k + 1$,
5 Repeat Steps 2 to 4
```

The innovation $\boldsymbol{\epsilon}_k$ can be determined with

$$\boldsymbol{\epsilon}_k = \boldsymbol{z}_k - \boldsymbol{H}_k\, \boldsymbol{x}_{k|k-1}, \tag{4.6}$$

which is essentially the difference between the measurement and the predicted output $\boldsymbol{y}_k$. The covariance of the innovation $\boldsymbol{S}_k$ can be obtained by

$$\boldsymbol{S}_k = \boldsymbol{H}_k\, \boldsymbol{P}_{k|k-1}\, \boldsymbol{H}_k^T + \boldsymbol{R}_k. \tag{4.7}$$

Knowing covariance of the innovation, the Kalman gain $\boldsymbol{K}_k$ can be computed as follow

$$\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1}\, \boldsymbol{H}_k^T\, (\boldsymbol{S}_k)^{-1} \tag{4.8}$$

Another notable problem with the extended Kalman filter is that the estimated covariance matrix tends to underestimate the true covariance matrix and therefor risks becoming inconsistent in the statistical sense. In addition, the covariance matrix $\boldsymbol{P}_k$ should be a symmetric and positive definite matrix, however, various numerical errors can result in $\boldsymbol{P}_k$ losing its symmetric property. An common approach to dealing with this is by symmetrizing it using

$$\boldsymbol{P} = \frac{1}{2}\left(\boldsymbol{P} + \boldsymbol{P}^T\right), \tag{4.9}$$

which needs to be done at a regular interval.

4.2 Discrete Time System

Most physical systems are represented as continuous-time models, while discrete-time measurements are frequently taken for state estimations. The Kalman filters are based on linear dynamic systems discretized in the time domain. The states of the system are represented as in a state vector. The total state vector $\boldsymbol{x}$ is formulated as

$$\boldsymbol{x} = \left[(\boldsymbol{p}^w)^T \quad (\boldsymbol{v}^w)^T \quad (\boldsymbol{\theta})^T \quad (\boldsymbol{\delta f})^T \quad (\boldsymbol{\delta\omega})^T \right]^T \tag{4.10}$$

with process noise vector,

$$\boldsymbol{v}_k = \left[(\boldsymbol{n_f})^T \quad (\boldsymbol{n_\omega})^T \quad (\boldsymbol{n_{\delta f}})^T \quad (\boldsymbol{n_{\delta\omega}})^T \right]^T \tag{4.11}$$

and the measurement noise vector

$$\boldsymbol{w}_k = \left[(\boldsymbol{n_{S_x}})^T \quad (\boldsymbol{n_{S_y}})^T \quad (\boldsymbol{n_{S_z}})^T \quad (\boldsymbol{n_\phi})^T \quad (\boldsymbol{n_\theta})^T \quad (\boldsymbol{n_\psi})^T \right]^T \tag{4.12}$$

Now that the equations of motions in Eq. (3.13) are set and the sensor dynamics are modeled in Eq. (2.2), the system can be discretized into an discrete time system by using an first order Taylor approximation of the time derivatives. Hence, the discrete state function $\boldsymbol{f}(\cdot)$ for the priori states $\boldsymbol{x}$

$$\begin{aligned}
\boldsymbol{p}^w_{k+1} &= \boldsymbol{p}^w_k + \mathrm{dt}_k\, \boldsymbol{v}_k \\
\boldsymbol{v}^w_{k+1} &= \boldsymbol{v}^w_k + \mathrm{dt}_k\, \boldsymbol{R}^w_b(\boldsymbol{f}^b + \boldsymbol{\delta f} + \boldsymbol{n_f}) + \mathrm{dt}_k\, \boldsymbol{g}^w \\
\boldsymbol{\theta}_{k+1} &= \boldsymbol{\theta}_k + \mathrm{dt}_k\, \boldsymbol{C}^w_b(\boldsymbol{\omega}^b + \boldsymbol{\delta\omega} + \boldsymbol{n_\omega}) \\
\boldsymbol{\delta f}_{k+1} &= \boldsymbol{\delta f}^b_k + \mathrm{dt}_k(\boldsymbol{\delta f}_0 + \boldsymbol{n_{\delta f}}) \\
\boldsymbol{\delta\omega}_{k+1} &= \boldsymbol{\delta\omega}^b_k + \mathrm{dt}_k(\boldsymbol{\delta\omega}_0 + \boldsymbol{n_{\delta\omega}})
\end{aligned} \tag{4.13}$$

Which, can be linearized along the predicted state vector trajectory using Eq. (4.3).

$$\boldsymbol{A}_k = \begin{bmatrix}
\boldsymbol{I}^3 & \mathrm{dt}_k\, \boldsymbol{I}^3 & \boldsymbol{O}^3 & \boldsymbol{O}^3 & \boldsymbol{O}^3 \\
\boldsymbol{O}^3 & \boldsymbol{I}^3 & \mathrm{dt}_k\, \hat{\boldsymbol{R}}^w_b(\boldsymbol{f}^b + \boldsymbol{\delta f}) & \mathrm{dt}_k\, \boldsymbol{R}^w_b & \boldsymbol{O}^3 \\
\boldsymbol{O}^3 & \boldsymbol{O}^3 & \mathrm{dt}_k\, \hat{\boldsymbol{C}}^w_b(\boldsymbol{\omega}^b + \boldsymbol{\delta\omega}) + \boldsymbol{I}^3 & \boldsymbol{O}^3 & \mathrm{dt}_k\, \boldsymbol{C}^w_b \\
\boldsymbol{O}^3 & \boldsymbol{O}^3 & \boldsymbol{O}^3 & \boldsymbol{I}^3 & \boldsymbol{O}^3 \\
\boldsymbol{O}^3 & \boldsymbol{O}^3 & \boldsymbol{O}^3 & \boldsymbol{O}^3 & \boldsymbol{I}^3
\end{bmatrix} \tag{4.14}$$

$$\boldsymbol{G}_k = \begin{bmatrix}
\boldsymbol{O}^3 & \boldsymbol{O}^3 & \boldsymbol{O}^3 & \boldsymbol{O}^3 \\
\mathrm{dt}_k\, \hat{\boldsymbol{R}}^w_b & \boldsymbol{O}^3 & \boldsymbol{O}^3 & \boldsymbol{O}^3 \\
\boldsymbol{O}^3 & \mathrm{dt}_k\, \hat{\boldsymbol{C}}^w_b & \boldsymbol{O}^3 & \boldsymbol{O}^3 \\
\boldsymbol{O}^3 & \boldsymbol{O}^3 & \mathrm{dt}_k\, \boldsymbol{I}^3 & \boldsymbol{O}^3 \\
\boldsymbol{O}^3 & \boldsymbol{O}^3 & \boldsymbol{O}^3 & \mathrm{dt}_k\, \boldsymbol{I}^3
\end{bmatrix}, \tag{4.15}$$

The representations $\hat{\boldsymbol{R}}^w_b = \frac{d}{d\boldsymbol{\theta}}(\boldsymbol{R}^w_n)$, and $\hat{\boldsymbol{C}}^w_b = \frac{d}{d\boldsymbol{\theta}}(\boldsymbol{C}^w_n)$ are used to simplify the expression, these are further evaluated in Appendix A.2. The notation of $\boldsymbol{I}^3$ is an unit matrix of size 3 by 3.

In the extended Kalman filter, the observation $\boldsymbol{z}_k$ consist of the orientation angles provided by the digital compass of the IMU, and the position provided by the ultrasonic rangefinders. The four distinct measurements from the ultrasonic rangefinder are assessed as follow $\bar{\boldsymbol{S}} = (S_1 - S_3,\, S_2,\, S_4)^T = (S_x,\, S_y,\, S_z)^T$, where $\bar{\boldsymbol{S}}$ is the position measurement. Since, the sensor payload can provide direct information on the positions $\boldsymbol{p}_b^w$ and the orientation $\boldsymbol{\theta}$ without any additional dynamics, the observation model $\boldsymbol{H}$, is chosen as linear and defined as

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{I}^3 & \boldsymbol{O}^3 & \boldsymbol{O}^3 & \boldsymbol{O}^3 & \boldsymbol{O}^3 \\ \boldsymbol{O}^3 & \boldsymbol{O}^3 & \boldsymbol{I}^3 & \boldsymbol{O}^3 & \boldsymbol{O}^3 \end{bmatrix}, \tag{4.16}$$

4.3 Observability of LTV system

A system with an initial state $\boldsymbol{x}(t_0)$ is observable if and only if the value of the initial state can be determined from the system output $\boldsymbol{y}(t)$ that has been observed throughout the time interval $t_0 < t < t_N$. If the initial state cannot be so determined, the system is considered unobservable. In this case, the linear time-varying (LTV) system can be expressed as

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}(\boldsymbol{x}_k)\,\boldsymbol{x}_k + \boldsymbol{G}(\boldsymbol{x}_k)\,\boldsymbol{v}_k \tag{4.17}$$

$$\boldsymbol{y}_k = \boldsymbol{H}\,\boldsymbol{x}_k + \boldsymbol{w}_k \tag{4.18}$$

The determination of observability for nonlinear systems, such as the system in (4.17), (4.18), is substantially more difficult than for linear systems. For linear systems, either one state is the optimal estimate, or infinitely many states are optimal estimates, in which case the system is unobservable. Nonlinear systems have the additional complication that finitely many states may be locally optimal estimates. One test to establish observability of a system that involves evaluating the rank of the observability grammiam $\boldsymbol{\mathcal{O}}^T\boldsymbol{\mathcal{O}}$,

$$\boldsymbol{\mathcal{O}}^T\,\boldsymbol{\mathcal{O}} = \boldsymbol{H}_0^T\,\boldsymbol{H}_0 + \sum_{k=1}^{N} \boldsymbol{\Phi}_{k-1,0}^T\,\boldsymbol{H}_k^T\,\boldsymbol{H}_k\,\boldsymbol{\Phi}_{k-1,0} \tag{4.19}$$

where $\boldsymbol{\Phi}_{k-1,0} = \boldsymbol{A}(\boldsymbol{x}_{k-1})...\boldsymbol{A}(\boldsymbol{x}_0)$ and $\boldsymbol{\mathcal{O}}_k$ refers to the kp $\times$ n observability matrix. If the observability grammiam $\boldsymbol{\mathcal{O}}^T\boldsymbol{\mathcal{O}}$ has full rank, the system is observable [20].

However, the observability does not ensure that the EKF implemented for a system is stochastically observable. Observability of a stochastic system is a necessary, but not sufficient, condition to ensure that the errors of the estimated state vector have bounded variance. A system is stochastically observable if there exists a finite time t_N such that the estimated state covariance matrix is bounded or less than a predefined threshold T_v [20],

$$\sigma_{max}\left(\boldsymbol{P}_k\right) < T_v \qquad\qquad k \geq N \tag{4.20}$$

where $N < \infty$ and $\sigma_{max}(\cdot)$ is the largest singular value of the matrix $(\cdot)$. Since, both condition are mathematically difficult to prove, both conditions are checked numerically for each test. The results are elaborated further in Chapter 5.7.

4.4 Sensor Fusion Algorithm

The propagation of the navigation states $\boldsymbol{p}_{\boldsymbol{b}}^{\boldsymbol{w}}$, $\boldsymbol{v}_{\boldsymbol{b}}^{\boldsymbol{w}}$, and $\boldsymbol{\theta}$ is driven by the measurements from the IMU according to the aided inertial navigation system Eq. (4.13). In order for the INS to propagate the navigation states properly, the system requires a recommend sampling at 100 Hz. Since, the SONAR has a lower sampling rate, the INS and EKF run separately.

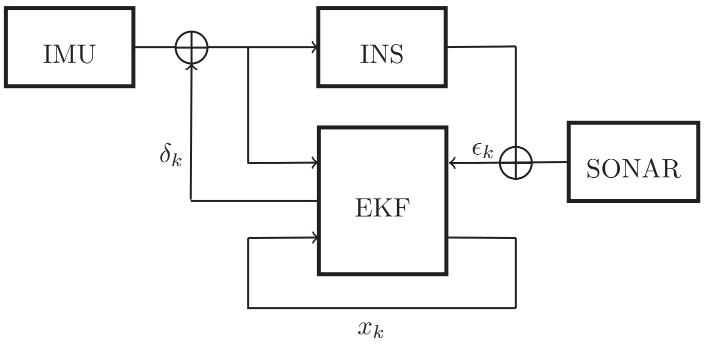


Figure 4.1: *Sensor fusion algorithm visualization.*

The ultrasonic sensors provide a measurement of the position $\bar{\boldsymbol{p}}$, which is linearly modeled as stated in the previous chapter. Whenever the system is observable, the Kalman filter will compare the measurements $\boldsymbol{z}_k$ with the predicted states according to the INS equations. As result, the filter will provide an estimated deviation $\boldsymbol{\delta}$. These deviations are used to correct the navigation states, but also the bias parameters of the accelerometer and gyroscope, respectively $\boldsymbol{\delta f}$, and $\boldsymbol{\delta\omega}$. The estimated biases are subtracted from the current IMU measurements, such that $\bar{\boldsymbol{f}}_k = \bar{\boldsymbol{f}}_k - \boldsymbol{\delta f}_k$, $\bar{\boldsymbol{\omega}}_k = \bar{\boldsymbol{\omega}}_k - \boldsymbol{\delta\omega}_k$, correcting the measurement before the INS equations are applied. Since the biases are corrected for, the estimated value of the biases after each measurement update in the Kalman filter are set to zero $\boldsymbol{\delta f}_k = \boldsymbol{\delta\omega}_k = 0$.

Algorithm 4.2: Sensor Fusion Algorithm

```
Initializations of $\boldsymbol{x}_{0|-1} = E(\boldsymbol{x}_0)$ and $\boldsymbol{P}_{0|-1} = COV(\boldsymbol{x}_0)$
Measure the IMU biases, and set $\boldsymbol{\delta f}_0$, $\boldsymbol{\delta\omega}_0$
for $k$ = 1,2,... do
    Get IMU measurements $\bar{\boldsymbol{f}}_k^b$, $\bar{\boldsymbol{\omega}}_k^b$
    Preform biases compensation:
        $\bar{\boldsymbol{f}}_k^b = \bar{\boldsymbol{f}}_k^b - \boldsymbol{\delta f}_k$
        $\bar{\boldsymbol{\omega}}_k^b = \bar{\boldsymbol{\omega}}_k^b - \boldsymbol{\delta\omega}_k$
    Propagate INS using Eq. (4.13) $\boldsymbol{p}_k^w$, $\boldsymbol{v}_k^w$, $\boldsymbol{\theta}_k$
    if System is observable then
        Form measurement $\boldsymbol{z}_k$
        Measurement update:
            Form innovation $\boldsymbol{\epsilon}_k = \boldsymbol{z}_k - \boldsymbol{H}\,\boldsymbol{x}_k$
            Compute Kalman gain $\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1}\,\boldsymbol{H}^T\,(\boldsymbol{S}_k)^{-1}$
            Correct states $\boldsymbol{x}_{k|k} = \boldsymbol{x}_{k|k-1} + \boldsymbol{K}_k\,\epsilon_k$
            Correct covariance $\boldsymbol{P}_{k|k} = [\boldsymbol{I} - \boldsymbol{K}_k\,\boldsymbol{H}_k]\,\boldsymbol{P}_{k|k-1}$
    else
        Maintain states $\boldsymbol{x}_{k|k} = \boldsymbol{x}_{k|k-1}$
        Maintain covariance $\boldsymbol{P}_{k|k} = \boldsymbol{P}_{k|k-1}$
    Time update:
        Update Jacobians $\boldsymbol{A}_k$, $\boldsymbol{G}_k$
        Update covariance $\boldsymbol{P}_{k+1|k} = \boldsymbol{A}_k\,\boldsymbol{P}_{k|k-1}\,\boldsymbol{A}_k^T + \boldsymbol{G}_k\,\boldsymbol{Q}_k\,\boldsymbol{G}_k^T$
```

4.5 Noise covariance matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$

In the previous section, we have assessed a method of fusing the data from an IMU and an ultrasonic sensor array, aided by an extended Kalman filter. The performance of the extended Kalman filtering depends heavily on the choice of noise covariance matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$. The noise parameters for the matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ were chosen with the aid of experimentations. The IMU was placed on a flat surface and remained stationary for a period 8.5 minutes. Since the IMU remains stationary on a flat surface, the expected output for the gyroscope is $\omega_x^b = \omega_y^b = \omega_z^b = 0$ rad/s for all axis and the expected out for the accelerometer is $f_z^b = 9.81$ m/s^2 and $f_x^b = f_y^b = 0$ m/s^2 for the planar field.

However, the data from the IMU is susceptible to additive noise and time-dependent sensor bias. As stated in Chapter 2.3, the time-dependent biases can be modeled as Brownian motion process, that is partly characterized by a constant null shift, also refereed to as the drift rate. An approximation of the drift rates of the accelerometer and gyroscope, respectively $\boldsymbol{\delta f}_0$ and $\boldsymbol{\delta\omega}_0$, can be found with linear regression of the stationary IMU's signals. The accelerometer and the gyroscope stationary measurements are respectively shown in Figure 4.2 and Figure 4.3.

Table 4.1: *Drift rate values found with the regression model $\delta(t) = \delta_0 \cdot t + a$.*

	x-axis $(\cdot 10^{-6})$	y-axis $(\cdot 10^{-6})$	z-axis $(\cdot 10^{-6})$
$\boldsymbol{\delta f}_0$	14.8	−1.01	9.85
$\boldsymbol{\delta\omega}_0$	0.57	0.54	−1.87

Table 4.1 provides information of the drift rates of the accelerometer and gyroscope. As can be seen, the drift rates are not very significant for the precision of short measurements. For long measurements, the drift rates may play a bigger role in accumulated error, however, the UAV's short flight timespan (10 - 30 min) limits the sensor errors to accumulate. The covariance of the process noise $\boldsymbol{v}$ define the parameters of the covariance matrix $\boldsymbol{Q}$. The standard deviation is the square root of the variance, thus, the expression for the process covariance matrix,

$$\boldsymbol{Q} = \alpha \cdot \mathrm{diag}\left(\sigma_{\boldsymbol{f}_x}{}^2, \sigma_{\boldsymbol{f}_y}{}^2, \sigma_{\boldsymbol{f}_z}{}^2, \sigma_{\boldsymbol{\omega}_x}{}^2, \sigma_{\boldsymbol{\omega}_y}{}^2, \sigma_{\boldsymbol{\omega}_z}{}^2, \sigma_{\boldsymbol{\delta f}}{}^2 \times \boldsymbol{I}^3, \sigma_{\boldsymbol{\delta\omega}}{}^2 \times \boldsymbol{I}^3\right) \tag{4.21}$$

where α is a scaling factor for tweaking the process covariance matrix $\boldsymbol{Q}$. The process noise signals are assumed to zero-mean Gaussian noise $\boldsymbol{v} \sim \mathcal{N}(0, \sigma)$. To verify the claim, the distributions of the signals are investigated and shown in Figure 4.4. The spread of the distribution will determine the values for the process covariance matrix. Table 4.2 shows the standard deviations of the accelerometer measurements $\boldsymbol{f}^b$, the gyroscope measurements $\boldsymbol{\omega}^b$, and the sensor biases $\boldsymbol{\delta}$

Table 4.2: *Values for the measurement noise covariance matrix $\boldsymbol{Q}$.*

$\sigma_{\boldsymbol{f}_x}$	$\sigma_{\boldsymbol{f}_y}$	$\sigma_{\boldsymbol{f}_z}$	$\sigma_{\boldsymbol{\omega}_x}$	$\sigma_{\boldsymbol{\omega}_y}$	$\sigma_{\boldsymbol{\omega}_z}$	$\sigma_{\boldsymbol{\delta f}}$	$\sigma_{\boldsymbol{\delta\omega}}$
0.0150	0.0117	0.0267	0.0033	0.0027	0.0032	0.0018	0.00062

Similar to the process noise, the measurement noise signals are assumed to zero-mean Gaussian noise $\boldsymbol{w} \sim \mathcal{N}(0, \sigma)$. The spread of the distribution will determine the values for the measurement covariance matrix. To verify the claim, the distributions of the measurement signals are investigated and shown in Figure 4.5, it is assumed that the distribution in each axis of the SONAR is similar. The measurement noise covariance matrix is shown in Eq. (4.22), where β is a scaling parameter. Table 4.3 shows the standard deviations of the ultrasonic rangefinder measurements $\bar{\boldsymbol{S}}$, and the orientation measurements $\bar{\boldsymbol{\theta}}$ provided by the digital compass of the IMU

$$\boldsymbol{R} = \beta \cdot \text{diag}\left({\sigma_{\boldsymbol{S}_x}}^2, {\sigma_{\boldsymbol{S}_y}}^2, {\sigma_{\boldsymbol{S}_z}}^2, {\sigma_{\boldsymbol{\phi}}}^2, {\sigma_{\boldsymbol{\theta}}}^2, {\sigma_{\boldsymbol{\psi}}}^2\right) \tag{4.22}$$

Table 4.3: *Values for the measurement noise covariance matrix* $\boldsymbol{R}$.

$\sigma_{\boldsymbol{S}_x}$	$\sigma_{\boldsymbol{S}_y}$	$\sigma_{\boldsymbol{S}_z}$	$\sigma_{\boldsymbol{\phi}}$	$\sigma_{\boldsymbol{\theta}}$	$\sigma_{\boldsymbol{\psi}}$
0.0220	0.0220	0.0220	0.1000	0.1000	0.0300

4.6 Summary

This chapter has presented the extended Kalman filter that is employed for filtering and estimating the navigation states of the INS. Although the EKF suffers from drawbacks, the system can still provide a reasonable performance. Kalman filters are based on linear discrete-time system, thus the nonlinear equations are linearized around the current estimates. Observability of nonlinear system is substantially more difficult than linear systems. For our LTV system, observability can be established if the observability has full rank. In addition, stochastic observability ensure that the errors of the state vector are bounded. Stochastic observability can be established if the state covariance matrix is bounded. Lastly, the chapter elaborated on the process noise and measurement noise covariance matrices, which were chosen based on a stationary measurement of the sensor payload.

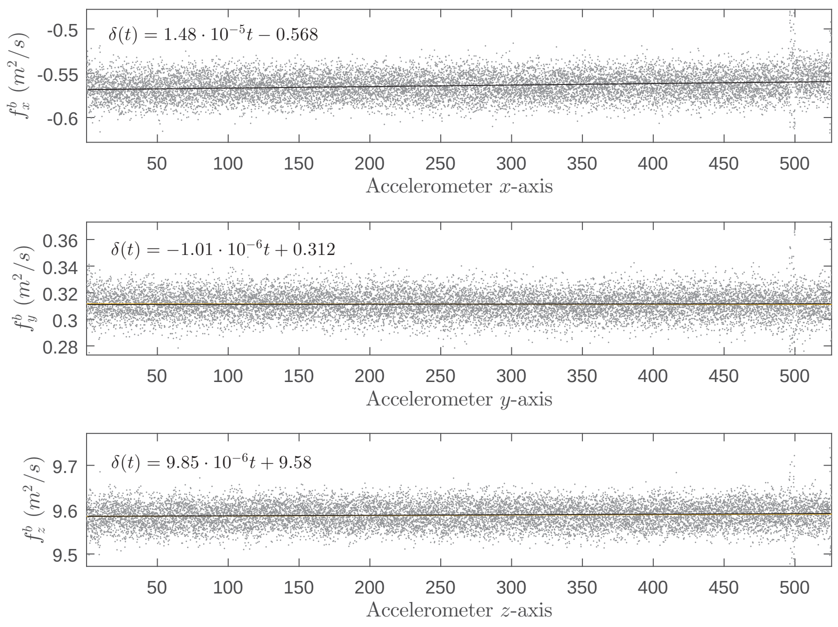


Figure 4.2: *The linear regression of the accelerometer signals $\boldsymbol{f}^b$ for the stationary case, where the regression model $\delta(t) = \delta_0 \cdot t + a$ is applied.*

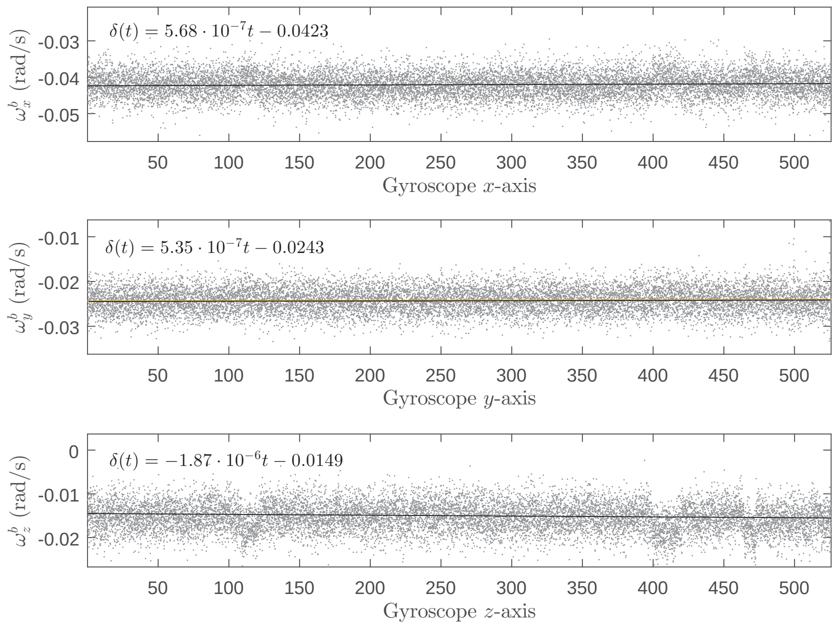
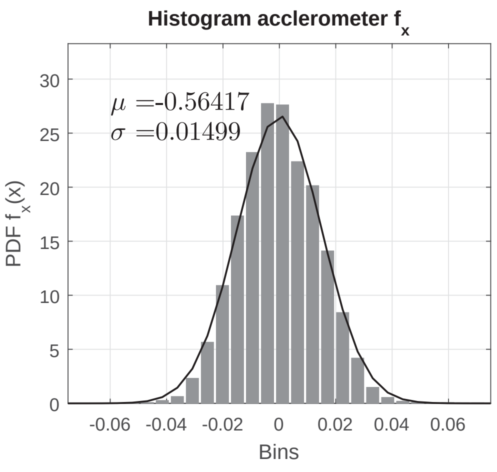
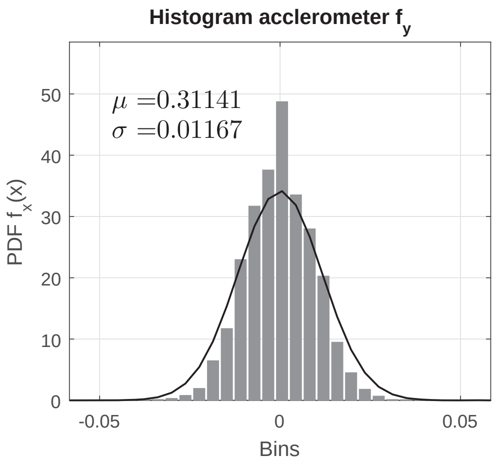


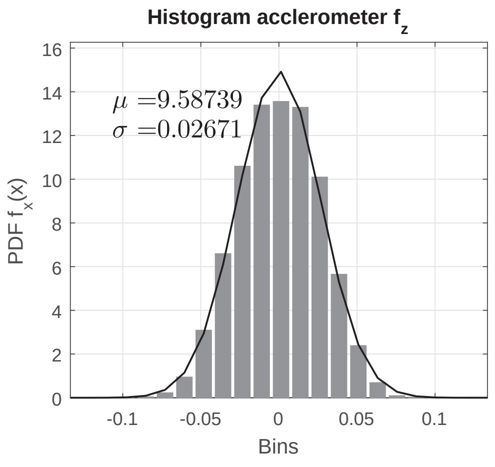
Figure 4.3: *The linear regression of the gyroscope signals $\boldsymbol{\omega}^b$ for the stationary case, where the regression model $\delta(t) = \delta_0 \cdot t + a$ is applied.*



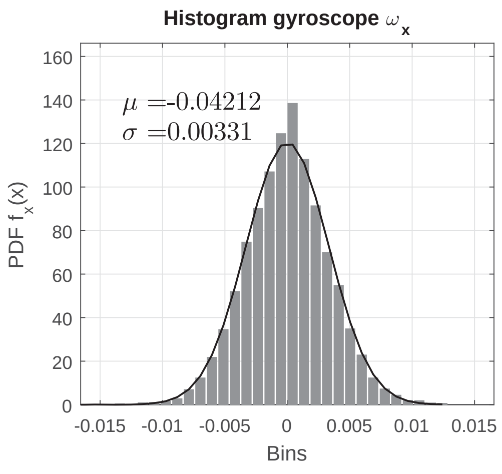
(a) Normal deviation accelerometer $\boldsymbol{f}_x$



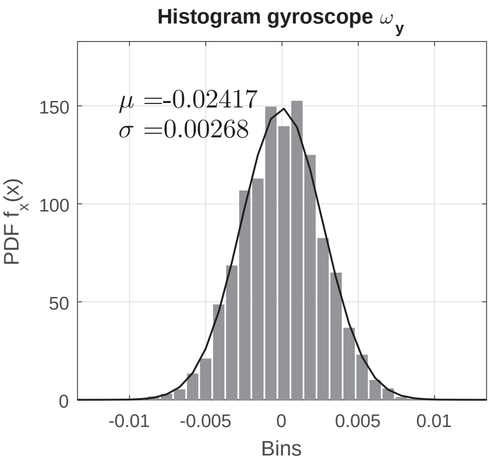
(b) Normal deviation accelerometer $\boldsymbol{f}_y$



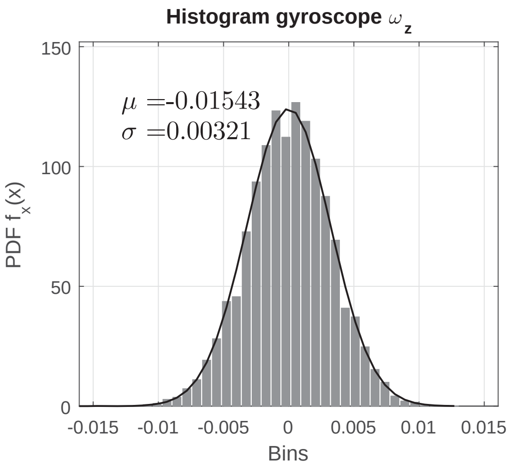
(c) Normal deviation accelerometer $\boldsymbol{f}_z$



(d) Normal deviation gyroscope $\boldsymbol{\omega}_x$

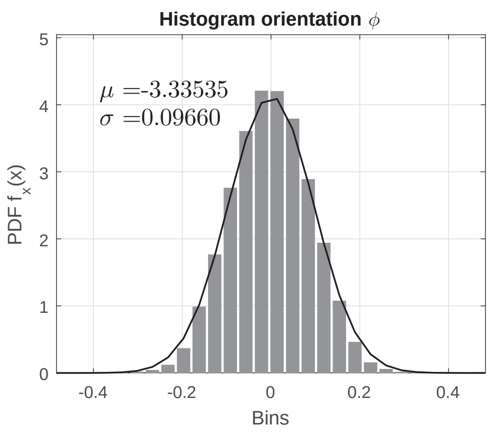


(e) Normal deviation gyroscope $\boldsymbol{\omega}_y$

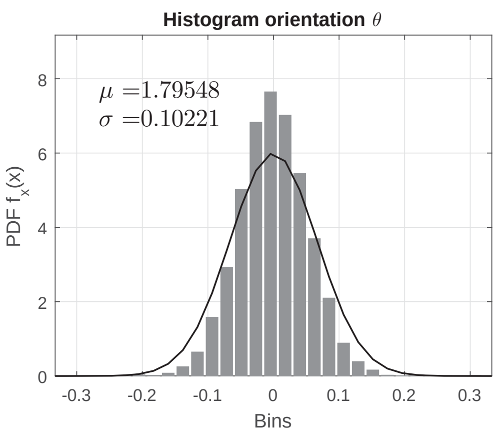


(f) Normal deviation gyroscope $\boldsymbol{\omega}_z$

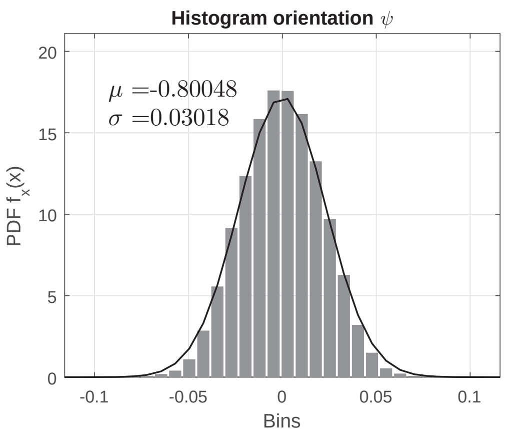
Figure 4.4: *Normal distribution plots of the process signals from the accelerometer and the gyroscope*



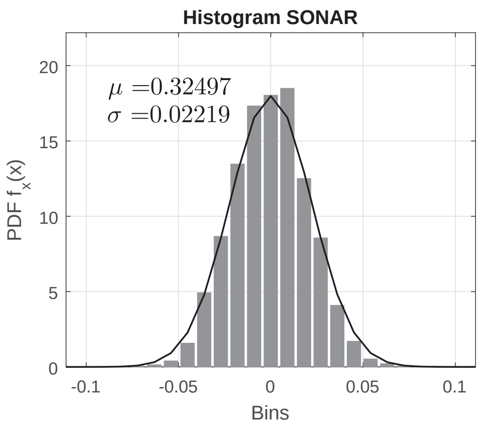
(a) Normal deviation roll angle ϕ



(b) Normal deviation pitch angle θ



(c) Normal deviation yaw angle ψ



(d) Normal deviation SONAR x-axis measurement S_x

Figure 4.5: *Normal distribution plots of the measurement signals from the digital compass and the SONAR sensor*

5 Experiments and Results

This chapter elaborates on the experimental setups and the findings. Insight into the SONAR characteristics will be provide, as well as the influences of the weighting on noise covariance matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$. The EKF aided INS is tested with an simulation, an stationary test, and two dynamic test with and without additional disturbance. Lastly, the numerical observability test will be performed.

5.1 SONAR Characterization

As stated in Chapter 2.2, the transmitted signal of an ultrasonic rangefinder propagates with propagation profile characterized by a beamwidth γ. According to the documentation [8], the beamwidth should be approximate $\gamma = 21.5°$. Lets assume that beamwidth is defined by the threshold angular range where the measurement error of the SONAR is larger than $1\pm$ cm. The sensor payload is maintained at a fixed height (1 meter), then the payload is gradually tilted over the x-axis upto an angle of approximately of 40°. The angle of incidence ϕ is measured through tilt-sensing by the IMU's digital compass. According to theory, the ultrasonic rangefinder should measure the height upto an angle $\phi \leq 21.5°$. During this procedure, it is made sure that the sensor is kept at the fixed height from the ground, so no additional errors are introduced.

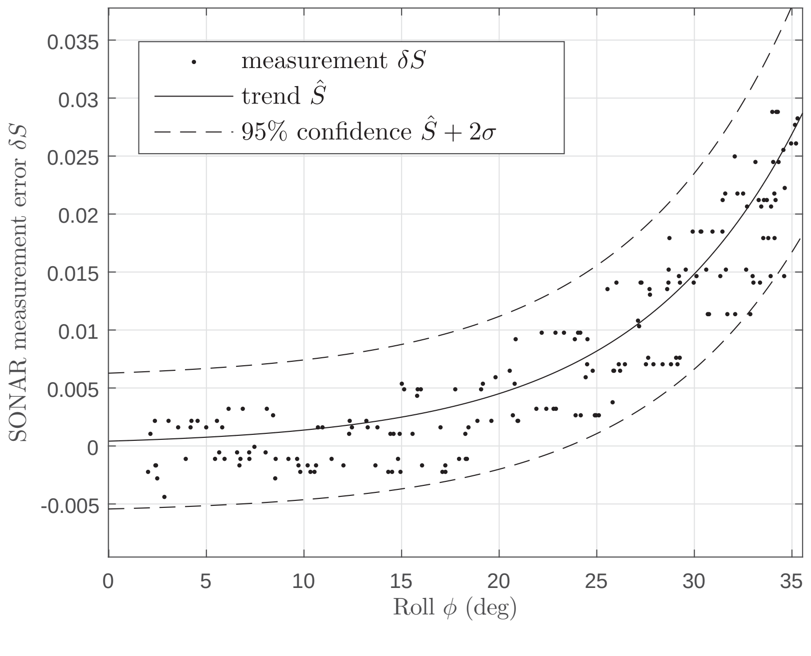


Figure 5.1: *The deviation of the sonar measurements in relation to sensor angle and estimated exponential trend. The standard deviation* $\sigma = 3.1$ *(mm).* $\hat{S} = b{\cdot}e^{a \cdot \phi}$ *where* $a = 4.18{\cdot}10^{-4}$ *and* $b = 1.19{\cdot}10^{-1}$.

Figure 5.1 shows the SONAR measurement errors δS in relation with the angle of incidence ϕ. Moreover, the figure is provided with an exponential regression $\hat{S}$, which indicate the trend of the SONAR measurement errors. As can be seen in Figure 5.1, the measurement errors remain under 1 cm upto an angle $\gamma \approx 18°$ with a 95% confidence interval. Hence, the standard deviation with respect to the regression $\hat{S}$ is $\sigma = 3.1$ mm. The deviation is close to the documented deviation of 3 mm [8].

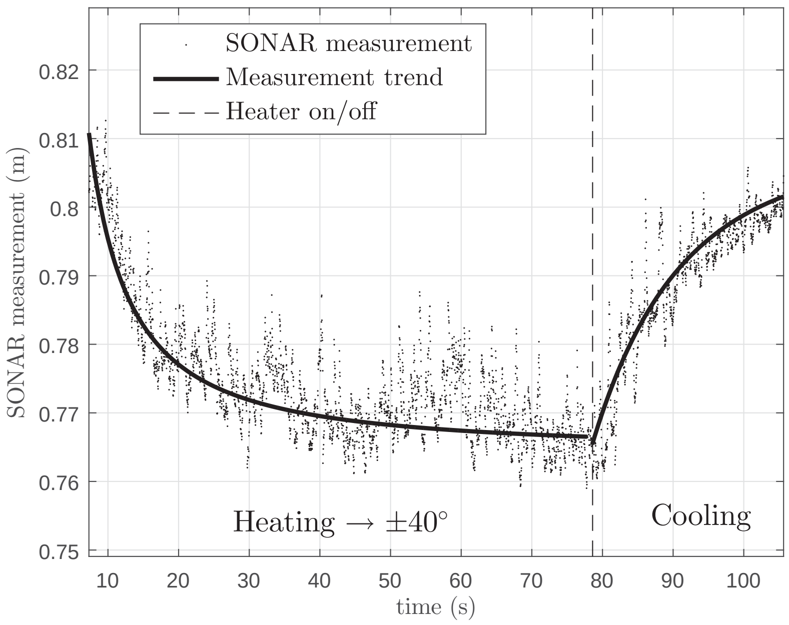


Figure 5.2: *Temperature dependency of the SONAR sensor, where the temperate ranges from* 20 - 40°*C*.

In addition, the temperature dependency of the SONAR sensors is checked. The sensor payload is positioned in a cylinder, where one SONAR sensor measures a fixed height. Then, the air in the cylinder is heated from room temperature (±20°C) upto a temperature of 40°C. After reaching the temperature of 40°C the air is cooled by natural convection. The purpose of the test is to validate that temperature disturbances would not dramatically interfere with the ultrasonic sensors. Figure 5.2 shows the behavior of the SONAR sensor measurements under temperature changes. As can be seen, the ultrasonic sensor shift from a measured 81 cm to 77 cm when heating up to 40°C. On ground of this finding, we can conclude that temperature changes should not be a significant issue for the UAV.

5.2 Simulation results

Before the physical system was tested, a virtual model is used to validate the proposed nonlinear EKF filter for the autonomous UAV navigation problem. The virtual model is made in SIMULINK, which has been provided Appendix A.5. The model uses the Runge-Kutta numerical solver (`ode4`) with a fixed timestep set to $\Delta t = 1 \cdot 10^{-2}$ (100 Hz), which has been set to match the IMU's sampling rate. The noise signals are modeled as white Gaussian-noise with a spread equal to the standard deviations found in Chapter 4.5.

In contrast to the physical system, the virtual model can reach smaller timesteps. Whereas, the sensor payload has an average sampling frequency of 35 Hz or $\Delta t = 2.8 \cdot 10^{-2}$ s, which is considered fairly poor for the INS to propagate the states properly . Moreover, the virtual model gives the advantage of providing the true states of the system, hence we are able to quantify the amount of noise filtering, which is difficult for the dynamical setup. The virtual movement of the UAV was simulated with an arbitrary acceleration $\boldsymbol{f}^b$ and an arbitrary angular rate $\boldsymbol{\omega}^b$. Figure 5.3 show the z-axis position estimates and Figure 5.4 show the three-axes orientation estimates. Both figures have been provide with the raw measurements.

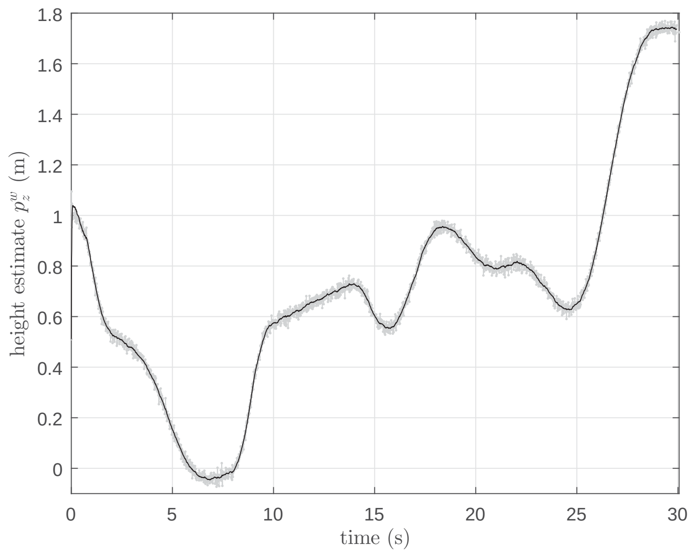


Figure 5.3: *Estimated and measured positions for the static measurement, where the EKF poses (black) and measured poses (gray). The scaling parameters $\alpha = 10$ and $\beta = 1$.*

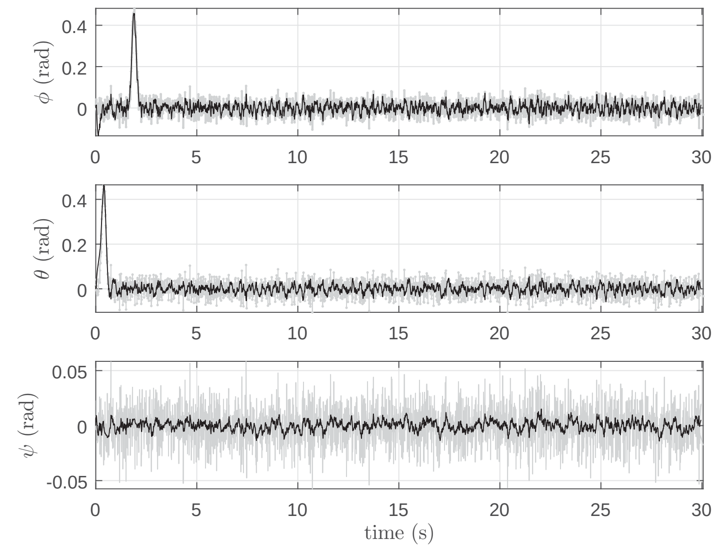


Figure 5.4: *Estimated and measured Euler angles for the simulation measurement, where the EKF poses (black) and measured poses (gray). The scaling parameters $\alpha = 10$ and $\beta = 1$.*

As can be seen, the sensor fusion algorithm works properly. It is clear that the EKF estimates provide navigation information more accurate in comparison to the measurements. The noise filtering is most noticeable in the amount of noise filtered in the orientations of the UAV. Since the true attitudes of the system are known in the virtual model, we are able to express the standard deviation of the measurements and signals. Table 5.1 provides insight in the effectiveness of the sensor fusion algorithm. As can be seen, the sensor fusion algorithm has a vast improvement upon the raw measurements from the sensors. The EKF reduced the orientation standard deviations of the x-, y-, and z-axis to around 24.1%, 30.1%, and 74.3% of the corresponding standard deviations measured by the digital compass. The filter reduced the position standard deviations of the x-, y-, and z-axis to around 12.6%, 10.5%, and 14.7% of the corresponding standard deviations measured by the SONAR sensors. The results show that the proposed EKF is effective in noise reduction and enables an significant accuracy improvement in orientation measurement, as position measurements.

Table 5.1: *Standard deviations of the measurements and state estimates in the simulation test.*

	Measurement	Sensor Fusion	Improvement
$\sigma_{\boldsymbol{p}_x}$	0.0143	0.0125	12.6%
$\sigma_{\boldsymbol{p}_y}$	0.0143	0.0128	10.5%
$\sigma_{\boldsymbol{p}_y}$	0.0143	0.0122	14.7%
$\sigma_{\boldsymbol{\phi}}$	0.0306	0.0231	24.1%
$\sigma_{\boldsymbol{\theta}}$	0.0306	0.0214	30.1%
$\sigma_{\boldsymbol{\theta}}$	0.0167	0.0043	74.3%

5.3 Influence of the covariance matrices Q and R

As stated previously, values of the noise sources influence the overall performance of the extended Kalman filter, therefore, the covariances matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ play an important roll. The process noise covariance $\boldsymbol{Q}$ contribute to the overall uncertainty. The measurement noise covariance $\boldsymbol{R}$ determine how much information from the measurement is used. Proper convergence of the estimates requires correct values of both matrices. In practice, the covariance matrices are often considered design variables, and chosen ad hoc [19]. In this case, the weighting of these matrices is determined by scaling the parameters α and β, where the ratio between these parameter α/β is investigated. Figure 5.5 shows the height estimate p_z^w and Figure 5.6 the roll estimate ϕ with different weightings for the covariance matrices.

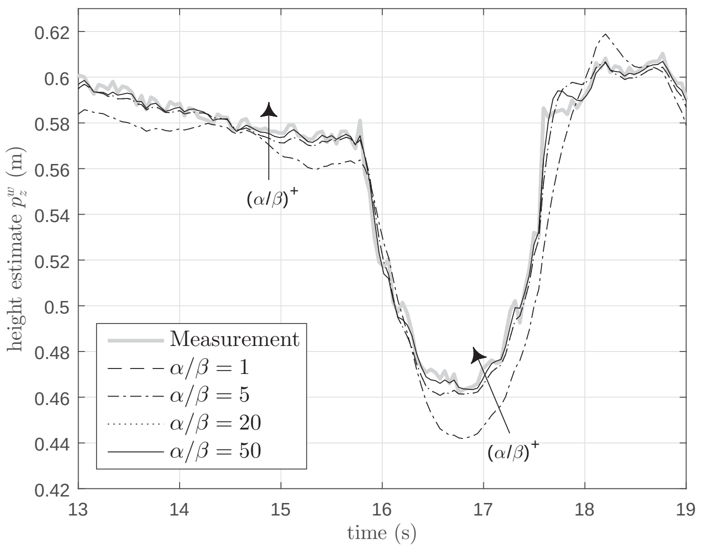


Figure 5.5: *Estimated position in the z-axis for different weighting on the covariance matrix.*

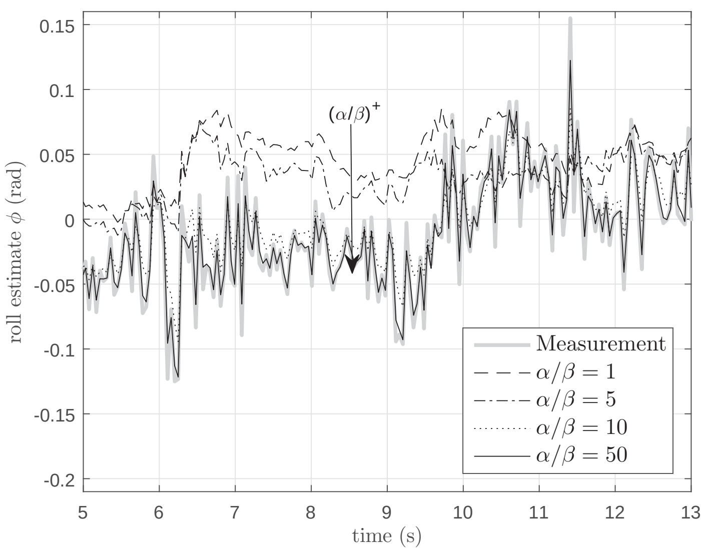


Figure 5.6: *Estimated Euler angles in the x-axis for different weighting on the covariance matrix.*

As can be seen, the covariances matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ have a huge impact upon the performance of the system. When $\boldsymbol{Q}$ is large, the Kalman filter tracks large changes in the data more closely than for smaller $\boldsymbol{Q}$. However, close tracking is at the cost of poor noise filtering. If $\boldsymbol{R}$ is large, the extended Kalman filter considers the measurements as not very accurate, resulting in smoother estimates. However, the absolute error of the estimates becomes larger. For smaller $\boldsymbol{R}$, the estimates will follow the measurements more closely. Both tracking and noise are equally important in the performance of sensor fusion in UAV platforms. Hence, the weightings $\alpha = 10$ and $\beta = 1$ are chosen as sufficient for this application.

5.4 Results of Static Measurements

In dynamic tests, the performance of the sensor fusion is more difficult be quantify, since the true states of the system are observed though measurements used for data fusion. In addition, high-precision three dimensional tracking of UAV platforms is difficult. Nevertheless, the performance of the sensor fusion in static tests are quantifiable. The SONAR sensor have an estimated sampling rate of $\Delta t = 2.8 \cdot 10^{-2}$ s (or 35 Hz) during the static test. Figure 5.7 shows the three-axis position measured by the SONAR sensors and Figure 5.8 shows the three-axis orientation measured by the digital compass. Both figure have been provided with the position and orientation state estimations from the EKF.

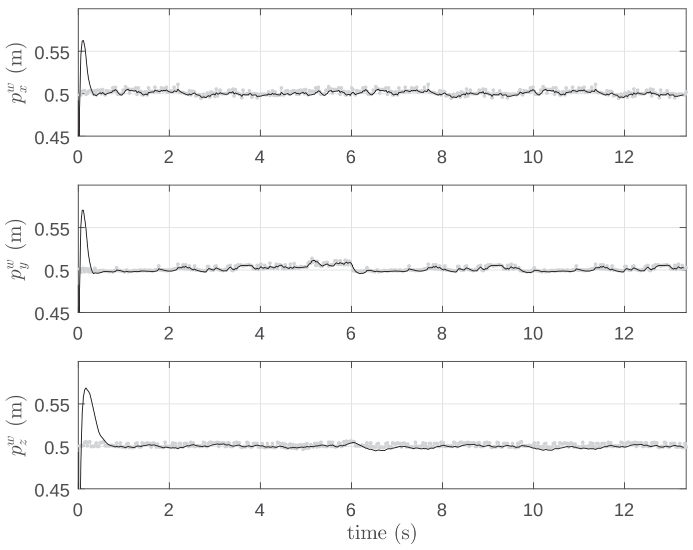


Figure 5.7: *Estimated and measured positions for the static measurement, where the EKF poses (black) and measured poses (gray). The scaling parameters $\alpha = 10$ and $\beta = 1$.*

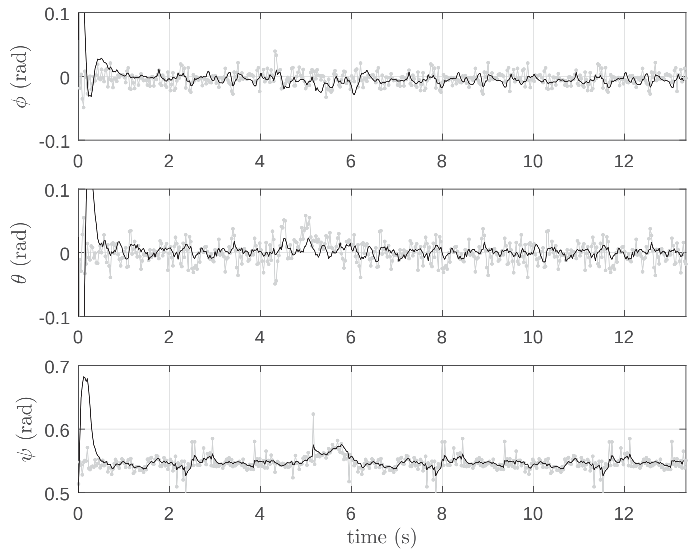


Figure 5.8: *Estimated and measured Euler angles for the static measurement, where the EKF poses (black) and measured poses (gray). The scaling parameters $\alpha = 10$ and $\beta = 1$.*

As can be seen, the sensor fusion algorithm works properly in the physical system. It is clear that the estimation results are slightly improved upon with the EKF in comparison to the raw measurements. Table 5.2 provides insight in the effectiveness of the sensor fusion algorithm. As can be seen, the sensor fusion algorithm provides the navigation information more accurately than raw measurements. The EKF reduced the orientation standard deviations of the x-, y-, and z-axis to around 30.1%, 27.1%, and 58.7% of the corresponding standard deviations measured using the digital compass. The filter reduced the position standard deviations of the x-, y-, and z-axis to around 6.49%, 7.45%, and 7.25% of the corresponding standard deviations measured using the SONAR sensors.

Table 5.2 shows that EKF is more effective in reducing the measurement variance of z-axis rotation than the other two axes. This indicates that introducing gyroscopes and EKF is of great help to improve the yaw measurement accuracy. However, the absolute errors and the ripples of z-axis are larger than the other two axes. This might due to the yaw calculation that is based on the measurement results of the roll and pitch, and therefore the errors of orientations in the z-axis are larger than those of x- and y-axis. An other explanation might be due to electrical disturbance, which the magnetometer is susceptible to, introduced by the stepper-motors of the UAV's rotors.

Table 5.2: *Standard deviations of the measurements and state estimates in the static test.*

	Measurement	Sensor Fusion	Improvement
$\sigma_{\boldsymbol{p}_x}$	0.0154	0.0144	6.49%
$\sigma_{\boldsymbol{p}_y}$	0.0188	0.0174	7.45%
$\sigma_{\boldsymbol{p}_y}$	0.0207	0.0192	7.25%
$\sigma_{\boldsymbol{\phi}}$	0.0103	0.0072	30.1%
$\sigma_{\boldsymbol{\theta}}$	0.0170	0.0124	27.1%
$\sigma_{\boldsymbol{\theta}}$	0.0165	0.0104	58.7%

It is worth noting that in dynamic tests the proposed extended Kalman filter might be performing significantly worse, due to linearisation producing unstable filters if the assumptions of local linearity is violated.

5.5 Results of Dynamic Measurements

The performance of the EKF in dynamics setups is evaluated using two scenarios. First scenario, a general dynamic test will be performed, where the purpose of the test is solely to investigate the noise-filtering capabilities. The sensor payload will be moved left - right, then forward - back and finally down - up. During the test, it is made sure that the sensor payload moves only orthogonally w.r.t. the sensor body frame, avoiding diagonal movement. The purpose of this test is to assess the EKF with dynamic measurements. Second scenario, a more complex movement will be performed where the purpose of the test is to validate that the aided INS can propagate specific movement pattern. In this case, the trajectory is a half circle with a diameter of 0.275 m. The sensor payload is navigated through the trajectory with the aid of guidance. Throughout both tests, the sensor payload is kept roughly planar, similar to the flight characteristics of an UAV. Like the static measurements, the EKF has a sampling rate of 35 Hz.

Figures 5.9 to Figure 5.12 show the state estimations from the first experimental setup, the figures also have been provided with the raw measurements.

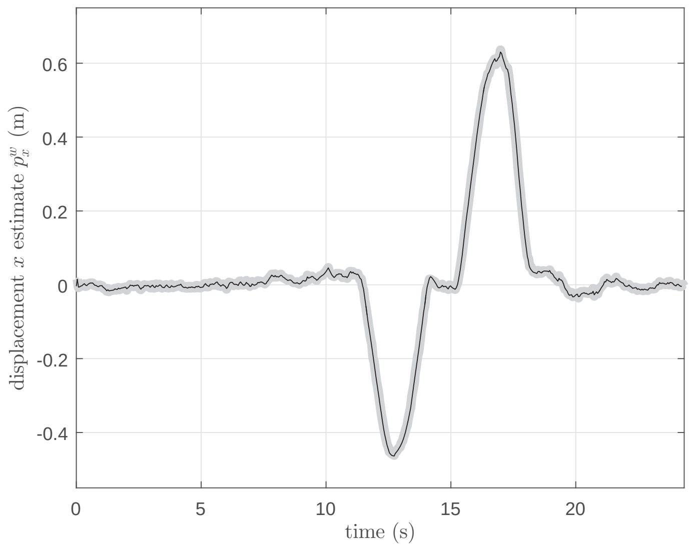


Figure 5.9: *Estimated and measured position x-axis for the dynamic measurement, where the EKF poses (black) and measured poses (gray). The scaling parameters $\alpha = 20$ and $\beta = 1$.*

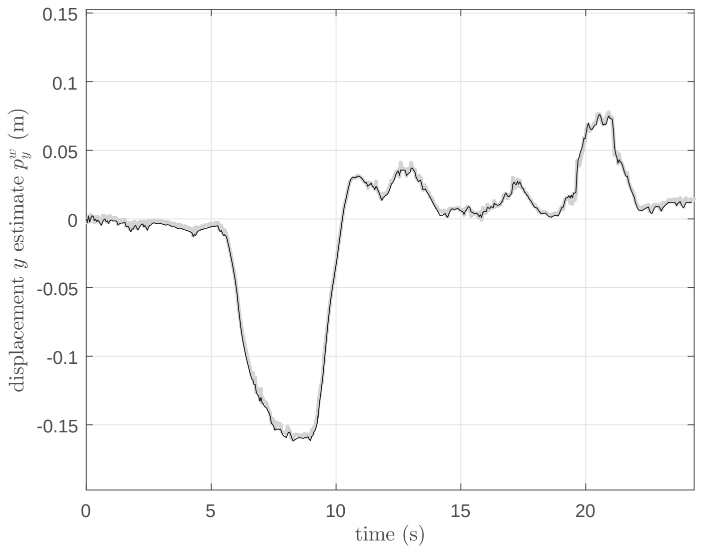


Figure 5.10: *Estimated and measured position y-axis for the dynamic measurement, where the EKF poses (black) and measured poses (gray). The scaling parameters $\alpha = 10$ and $\beta = 1$.*

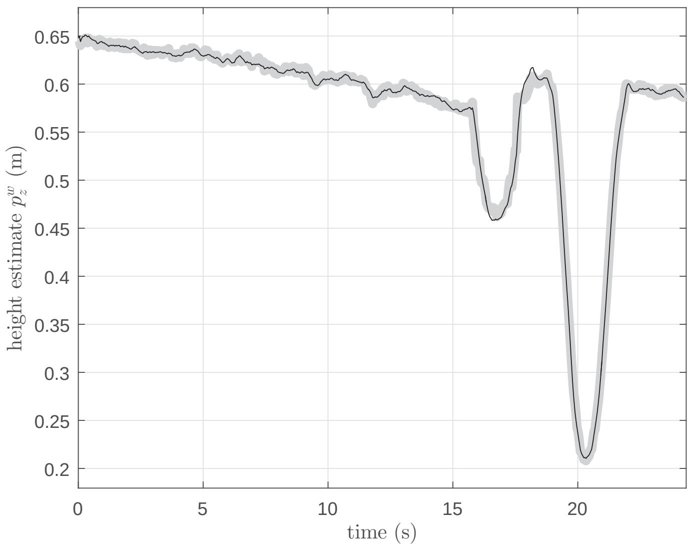


Figure 5.11: *Estimated and measured position z-axis for the dynamic measurement, where the EKF poses (black) and measured poses (gray). The scaling parameters $\alpha = 10$ and $\beta = 1$.*

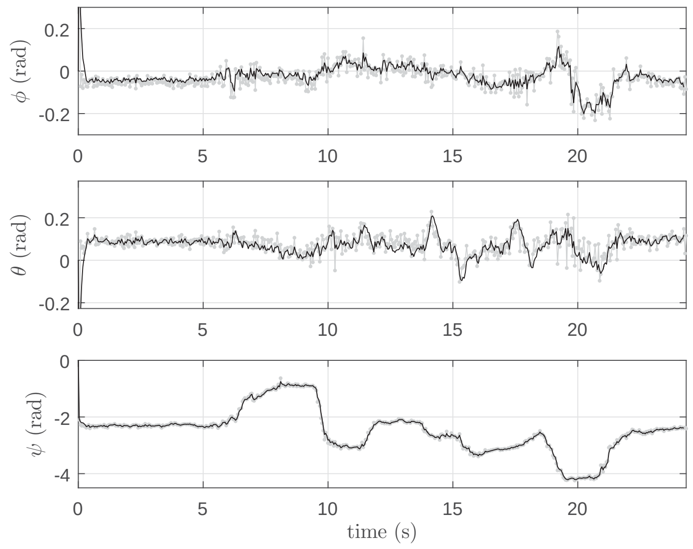


Figure 5.12: *Estimated and measured Euler angles for the dynamic measurement, where the EKF poses (black) and measured poses (gray). The scaling parameters $\alpha = 10$ and $\beta = 1$.*

As can be seen, the sensor fusion algorithm works properly in the dynamic test, as well. The estimates follow the measurements properly. Although, the results are not quantifiable, unlike the virtual model and the static measurements, the results show that sensor fusion algorithm offers a reliable system. Like the previous results, most significant noise reduction is in the orientation estimates. Next, the second scenario is considered. The purpose of the setup is to validate that the system can track a specific pattern (half circle) accurately. As with the previous test, the sensor payload is kept at planar attitude, to simulate UAV flight conditions. The results for the planar displacement estimates p_x^w and p_y^w are shown in Figure 5.13. The reference pattern is also shown.

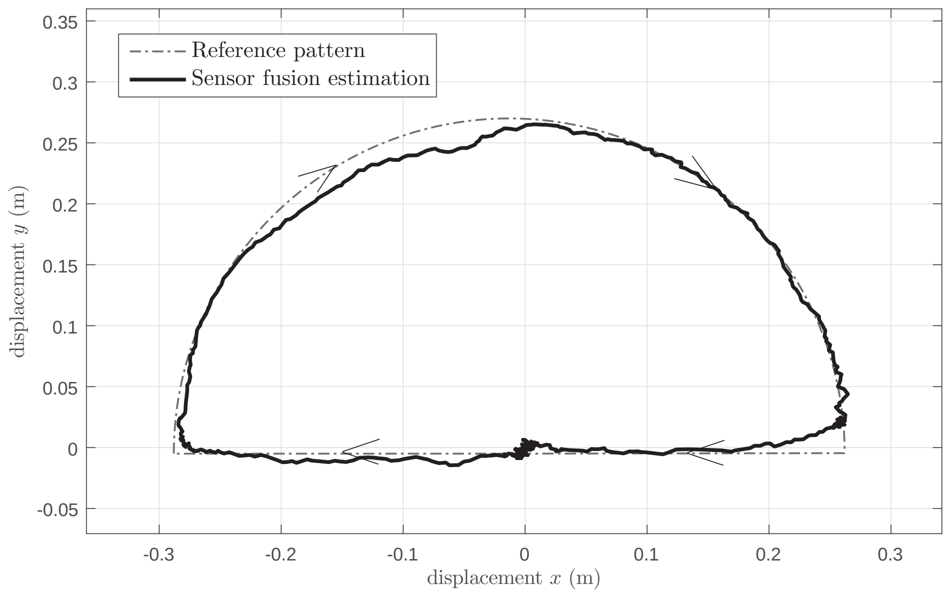


Figure 5.13: *Estimated and measured poses for the circle test setup, where the estimates (black) and reference pattern (black dotted). The scaling parameters $\alpha = 10$ and $\beta = 1$.*

As can be seen in Figure 5.13, the tracking is relatively close to the desired pattern. An noticeable result, is an unexpected little dent in the curve at approximately $x = -0.2$, $y = 0.25$. The algorithm deviates significantly from the reference pattern. The reason for the deviation might be due to interference with the other ultrasonic sensor. All the ultrasonic sensor work on on same frequencies (40 kHz) and same impulse signals. It is worth noting that the tracking of the EKF at the corners of the half circle is surprisingly good, since the linearisation around the estimate $\boldsymbol{x}_k$ has a sharp changes in trajectory. In short, the results indicate that the EKF aided INS has the capability of properly tracking patterns, without the issue of scaling or deformation of the reference pattern.

5.6 Results with Periodic Rotor Disturbance

The previous test were all performed in disturbance free setups, where rotor's periodic disturbance is neglected. Typically, the angular rates of the rotor vary between 3000 - 7000 RPM. To balance the angular momentum induced by the rotors, UAV make use of clock and anti-clock wise rotors. In this case, the sensor payload clammed to an UAV with a broken rotor, hence the angular momentum around the center of gravity in non-zero. All the rotors are operated at a fixed angular velocity of approximately 3000 RPM (314 rad/s). The purpose of the setup is to test if the system would be able properly localize with the additional periodic disturbance. Note, that the system would never be subjected to such disturbance, since rotor combination would reduce the oscillations. Additionally, some isolation between sensor payload and UAV would decrease the perturbations. However, the test should illustrate the influence the rotor disturbance has on the inertial measurement unit and the ultrasonic sensor array. The Figure 5.14 to Figure5.17 show the dynamics result with the additional rotor disturbance.

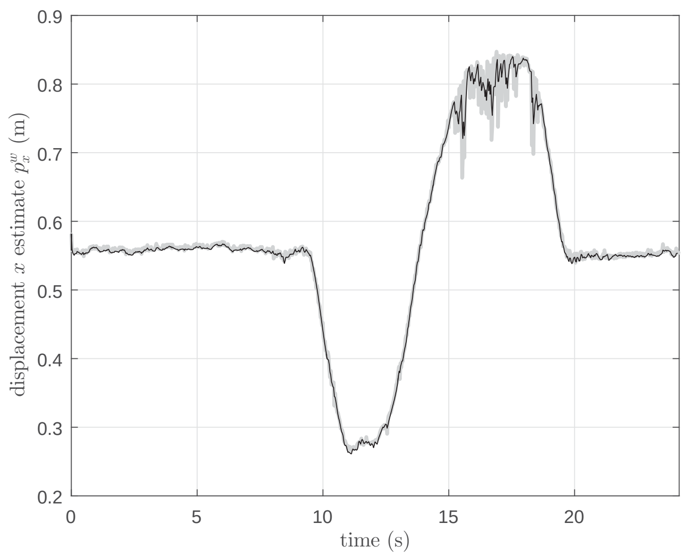


Figure 5.14: *Estimated and measured position x-axis for the dynamic measurement, where the EKF poses (black) and measured poses (gray). The scaling parameters $\alpha = 20$ and $\beta = 1$.*

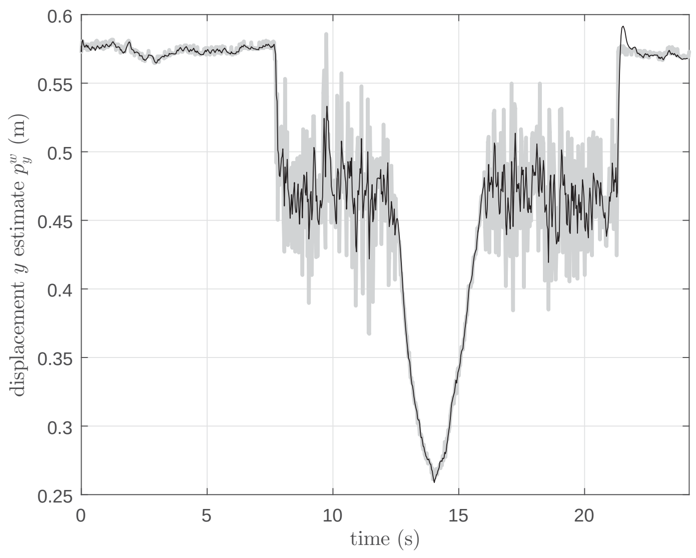


Figure 5.15: *Estimated and measured position y-axis for the dynamic measurement, where the EKF poses (black) and measured poses (gray). The scaling parameters $\alpha = 20$ and $\beta = 1$.*

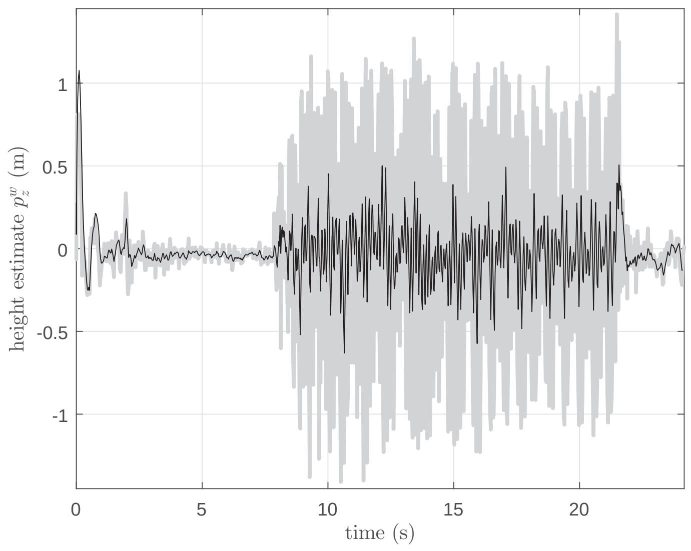


Figure 5.16: *Estimated and measured position z-axis for the dynamic measurement, where the EKF poses (black) and measured poses (gray). The scaling parameters $\alpha = 20$ and $\beta = 1$.*

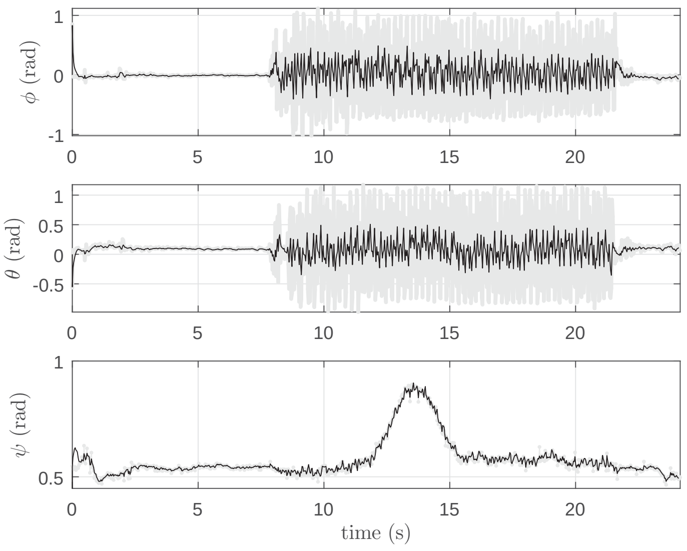


Figure 5.17: *Estimated and measured Euler angles for the dynamic measurement, where the EKF poses (black) and measured poses (gray). The scaling parameters $\alpha = 20$ and $\beta = 1$.*

As can be seen, the system performs poorly with the addition of the external disturbance. As to be expected, the roll and pitch measurements suffer more from the disturbance than the yaw measurement. The absolute error of the roll and pitch angles $< 25.2°$ Most noticeable, the SONAR measurements contain heavy oscillations that were not present during experimentation. For instance, Figure 5.16 show oscillations with an unrealistic amplitude of ± 1 m, which indicates that the SONAR might be susceptible to unaccountable non-linear behavior. The absolute errors of the height estimates are drastically large < 0.511 m. An explanation for these tremendous oscillation might be due to frequency shifting (Doppler effect). The receiver of the SONAR specifically listens for 40 kHz pulses, hence the sensor might miss pulses. Although, the EKF filters the oscillation to some extend, the estimate are a poor representation of state of the system. Surprisingly, the system deals better with these oscillations when in motion, as can be seen in Figure 5.15.

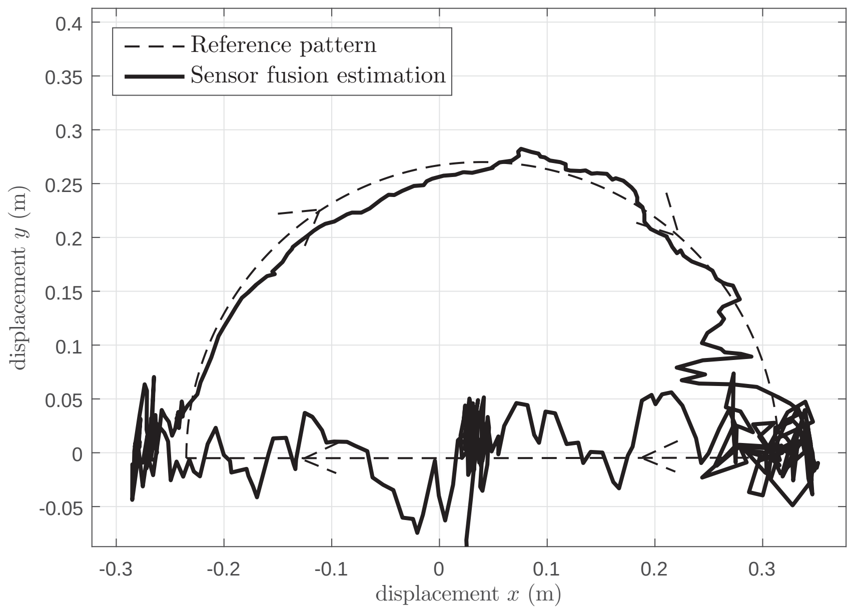


Figure 5.18: *Estimated and measured poses for the circle test setup, where the estimates (black) reference pattern (black dotted). The scaling parameters $\alpha = 10$ and $\beta = 1$.*

In order to illustrate the amount of perturbation in the position estimates, the planar displacement of the UAV (p_x^w and p_y^w) is plotted and shown in Figure 5.18. As can be seen, the oscillations are tremendously large and the system truly performs poorly with the additional rotor disturbance. The position estimates deviate drastically from the reference pattern. The reason is that the extended Kalman filter is provided with bad measurements from the ultrasonic rangefinder. In the extended Kalman filter the weighting of process noise is higher than the weighting of output noise, since the observations are deemed accurate. However, the ultrasonic rangefinder provide inaccurate measurements, and therefor the system works poorly.

In conclusion, the high frequent external disturbance has a huge impact on the ultra-rangefinders. Although, the sensor fusion between the inertial sensor and ultrasonic ranged finders suppresses a vast amount of oscillation, the estimates are a bad representation of the true states due to bad measurement provided by the ultrasonic rangefinder. Thus, solely sensor fusion is not a solution to counter these oscillations. Either other sensors for position measurements, or an different filter approach should be considered.

5.7 Observability test

The observability of the EKF implemented for the INS architecture can be assessed using the test discussed in Chapter 4.3. As previously stated, the determination of observability for LTV system is substantially more difficult, than for linear systems. To establish observability of a system, one checks the rank of the observability grammiam $\mathcal{O}^T\mathcal{O}$. To establish if the matrix $\mathcal{O}^T\mathcal{O}$ is full rank, one can check whether the matrix has a non-zero determinant. If the observability grammiam is full rank, then the system is observable. Figure 5.19 shows the determinant of the observability grammiam of multiple tests evaluated over a time range $t \in [t_0, t_N]$.

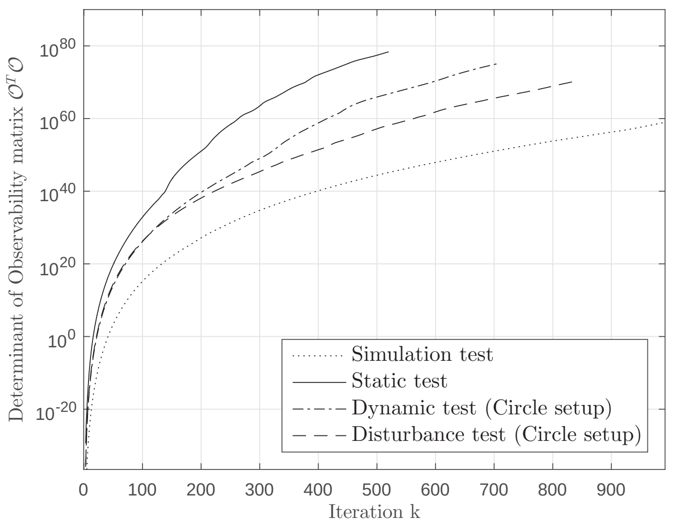


Figure 5.19: *The determinant of the matrix $\boldsymbol{\mathcal{O}}^T\boldsymbol{\mathcal{O}}$ against time for multiple test setups.*

As can be seen, the determinant is increases over time iterations. For the iterations $k > 2$ the determinant of the observability grammiam $\det(\boldsymbol{\mathcal{O}}^T\boldsymbol{\mathcal{O}}) \neq 0$, thus the grammiam has full rank. Hence, one can conclude that the system is observable at time t_N where the time $t_N < \infty$ and $t_N > t_2$. Additionally, stochastic observability is a necessary condition to ensure that the errors of the estimated state vectors are bounded. A system is stochastically observable if the state covariance matrix is bounded or less than a predefined threshold. Figure 5.20 shows the maximum singular value of the state covariance matrix for multiple test that are considered in this report.

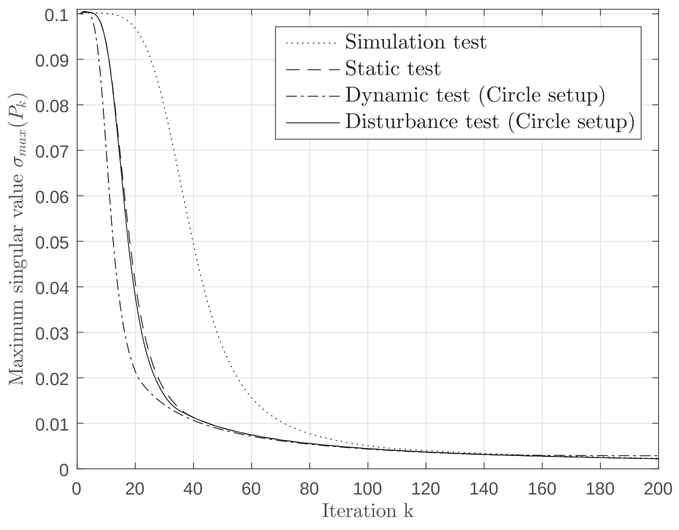


Figure 5.20: *Maximum singular values of the covariance matrix $\boldsymbol{P}_k$ against time for multiple test setups.*

As can be seen, the maximum singular value of the covariance matrix is initially high and convergence to a stable value. The convergence of the maximum singular value concludes that $\sigma_{max}(\boldsymbol{P}_k)$ can indeed be bounded by a constant T_v for iterations $k \geq N$, and therefor stochastic observability can be concluded.

6 Conclusion

A novel sensor fusion algorithm has been presented applicable to inertial measurements units (IMU) and ultrasonic sensor array. The approach uses an extended Kalman filter to fuse the data from different sensors together to produce an accurate estimate of the true system state. The performance of the EKF aided INS has been evaluated using simulations, static measurements, and dynamic measurements.

In the simulation setup, the system was subjected to an arbitrary acceleration and angular rate, where the gimbal lock boundaries were apprehended. The simulations indicated that the data fusion between the sensors improved the overall quality of the data. The results showed that the EKF is effective in noise reduction and enables significant accuracy improvement in orientation as position measurements. The noise reduction is most significant in the orientation estimates. The physical system showed similar promising results. In static state, the EKF reduced the noise in orientation of the x-, y- and z-axis to around 30.1%, 27.1% and 58.7%, respectively. Although the noise reduction for the orientation estimate around the z-axis are larger than for the x- and y-axis, the absolute error is larger since the yaw measurement is based on the roll and pitch estimates. In position estimates, the noise in the x-, y- and z-axis were reduced to around 6.49%, 7.45% and 7.25%, respectively. Although, the dynamic tests are not quantifiable they are deemed accurate and justify that the system could provide a solution to improve the navigation information in UAV platforms. Additionally, the system has been subject to high frequent periodic disturbance. One rotor of the UAV has been equipped with a broken rotor. All the six rotors are operated at a fixed speed of approximately 3000 RMP. The results indicated that the system works poorly with the additional rotor disturbance. The absolute errors of the roll and pitch angles $<25.2°$ and the absolute errors of the height estimates < 0.511 m. The addition had the most impact upon the SONAR measurements, where the data suffered from tremendous oscillations. An possible explanation could be a slight frequency shift of the echo pulse (Doppler effect) resulting in the SONAR missing the echo pulse.

In conclusion, data fusion between an IMU and ultrasonic sensor array has been successfully implemented that will improve the quality of the navigational information. Nevertheless, external rotor disturbance has a negative affect on the system and poses a limitation on the system for its practically in UAV platforms. Thus, further research should be invested into the dynamics of ultrasonic rangefinders.

7 Recommendation

Although, the sensor fusion setup worked properly and is fairly robust there is some room for further improvements to the system.

Quaternion

As stated in Chapter 3.1, the orientation of the UAV is bounded to prevent gimbal lock from occurring. An alternative way to express the orientation of the UAV, without the gimbal lock limitation, are quaternions. A quaternion is a four-dimensional complex number $\boldsymbol{q} = [q_1, q_2, q_3, q_4]^T$ that can be used to represent the orientation of a rigid body or coordinate frame in three-dimensional space [22].

Although, quaternions are less intuitive than Euler angles, the quaternion have the computational advantage in size and speed (quaternion multiplication is much faster than 3×3 matrix multiplication). Less computational load provides a more stable and effective platform. They are often used as internal states for orientation representation in computer implementation of navigation applications [23].

Adaptive Extended Kalman Filtering

To robustly handle uncertainty in the standard deviations of the sensors and process noises, an adaptive filter can applied to identify the values of $\boldsymbol{Q}$ or $\boldsymbol{R}$, unlike the EKF where the covariance matrices need to be chosen. There are many different adaptive filters in existence. The Method of Maximum Likelihood Estimation (MMLE) is a technique that could be applied to the current extended Kalman filters [25]. The basic premise is to use the measurement and state residuals to modify the parameter values for sensor and process noise.

Particle filter

Particle filtering is a general Monte Carlo method for performing inference in state-space model, where the state of a system propagates in time. The information is obtained via noise measurements made at each time step [26]. The system could predict its own location, based on sampled measurements from the environment, similar to a SLAM algorithm [21]. The technique provides an better localization algorithm then distance measurements provide from the ultrasonic sensors. However, the amount of ultrasonic sensors in the current setup cover to little range to operational with a particle filter. If one ultrasonic sensor covers approximately $40°$ (two times the beam-width) of range, atleast 9 ultrasonic sensors are required for full $360°$ detection.

References

[1] Antonino Catena, Carmelo Donato Melita, Automatic Tuning Architecture for the Navigation Control Loops of Unmanned Aerial Vehicles, Science and Bussiness Media, Dordrecht, 2013.

[2] Henry Hexmoor, Brian McLaughlan, Matt Baker, Swarm Control in Unmanned Aerial Vehicles, University of Arkansas, Arkansas.

[3] Nils Gageik, Michael Strohmeier, Sergio Montenegro, An Autonomous UAV with an Optical Flow Sensor for Positioning and Navigation, Aerospace Information Technology, University of Würzburg, Würzburg, 2013.

[4] Anoop Cherian, Jon Andersh, Autonomous Altitude Estimation Of A UAV Using A Single Onboard Camera, University of Minnesota, Minneapolis, 2009.

[5] Fei Wang, Jinqiang Cui, Swee King Phang, T.H. Lee A mono-camera and scanning laser range finder based UAV indoor navigation system University of Atlanta, Atlanta, 2013.

[6] Roy Edgar Hansen, Introduction to SONAR, University of Oslo, Oslo, 2012.

[7] Alex Cao, Johann Borenstein, Experimental Characterization of Polaroid Ultrasonic Sensors in Single and Phased Array Configuration, University of Michigan, Michigan, 2002.

[8] Ultrasonic sensor, HC-SR04 documentation, Micropik, retrieved from http://www.micropik.com/PDF/HCSR04.pdf,

[9] Nikolas Trawny, Stergios Roumeliotis, Indirect Kalman Filter for 3D Attitude Estimation, University of Minnesota, Minneapolis 2005.

[10] Nicholas Rotella, Sean Mason, Stefan Schaal, Ludovic Righetti, Inertial Sensor-Based Humanoid Joint State Estimation 2016.

[11] He Zhao, Zheyao Wang, Motion Measurement Using Inertial Sensors, Ultrasonic Sensors, and Magnetometers With Extended Kalman Filter for Data Fusion, IEEE Sensors Journal, 2012.

[12] Jonghyuk Kim, Lee Ling Ong, Eric Nettleton, Salah Sukkarieh. Decentralised approach to UAV navigation: without the use of GPS and preloaded maps, University of Sydney, Sydney, 2006.

[13] Fredrik Gustafsson, Statistical Sensor Fusion, Studentliteratur AB, Lund, 2nd edition, 2012.

[14] Talat Ozyagcilar., Implementing a Tilt-Compensated eCompass using Accelerometer and Magnetometer Sensors, Freescale Semiconductors, 2015, p3.

[15] Simon Julier, Jeffrey Uhlmann, A New Extension of the Kalman Filter to Nonlinear Systems, The University of Oxford, Oxford, 1997, p1-2.

[16] Kenneth Gustavsson, UAV Pose Estimation using Sensor Fusion of Inertial, Sonar and Satellite Signals, Uppsala Universitet, Uppsala, 2015.

[17] Chi Chiu, Error Reduction Techniques for a MEMS Accelerometer-based Digital Input Device, University of Hong Kong, Hong Kong, 2008.

[18] Tara Bracken, Multimodal Noncontact Tracking of Surgical Instruments, The University of Western Ontario, London, 2013.

[19] Martin Nilsson, Kalman Filtering with Unknown Noise Covariances Swedish Institute of Computer Science, Kista, 2006.

[20] Vibhor Lal Bageshwar, Quantifying Performance Limitations of Kalman Filters in State Vector Estimation Problems University of Minnesota, Minnesota, 2008.

[21] Jonghyuk Kim, Lee Ling Ong, Eric Nettleton, Salah Sukkarieh, Decentralised approach to UAV navigation: without the use of GPS and preloaded maps, University of Sydney, Sydney, 2006.

[22] Sebastian Madgwick, An efficient orientation filter for inertial and inertial/magnetic sensor arrays, 2010.

[23] Dave Zachariah, Estimation for Sensor Fusion and Sparse Signal Processing, KTH Royal Institute of Technology, Stockholm, 2013.

[24] Antoni Burguera, Yolanda González, Gabriel Oliver, Sonar sensor models and their application to mobile robot localization Universitat de les Illes Balears, Palma de Mallorca, 2009.

[25] M. Oussalah, J. De Schutter, Adaptive Kalman Filtering for Noise identification K.U. Leuven, Leuven, 2001.

[26] Emin Orhan, Bayesian Inference: Particle Filtering University of Rochester, Rochester, 2012.

A Appendix

A.1 Angular rate rotation matrix

The angular rate rotation matrix $\boldsymbol{C}_b^w$ translates angular rate from the fixed body frame $\{b\}$ to the world frame $\{w\}$. The angular rates in the world frame $\{w\}$ can be obtained from the gyroscope using the following equations.

$$\begin{bmatrix} \omega_x^b \\ \omega_y^b \\ \omega_z^b \end{bmatrix} = \begin{bmatrix} \omega_x^w \\ 0 \\ 0 \end{bmatrix} + R^T(x,\phi) \begin{bmatrix} 0 \\ \omega_x^w \\ 0 \end{bmatrix} + R^T(x,\phi) R^T(y,\theta) \begin{bmatrix} 0 \\ 0 \\ \omega_x^w \end{bmatrix}. \tag{A.1}$$

Therefore,

$$\begin{bmatrix} \omega_x^w \\ \omega_y^w \\ \omega_z^w \end{bmatrix} = \boldsymbol{C}_b^w \, \boldsymbol{\omega^b} \tag{A.2}$$

$$= \begin{bmatrix} 1 & s\phi\, t\theta & c\phi\, t\theta \\ 0 & c\phi & -s\phi \\ 0 & s\phi/c\theta & c\phi/c\theta \end{bmatrix}, \tag{A.3}$$

where $c\phi$ is defined as $cos(\phi)$, $s\phi$ is defined as $sin(\phi)$, and $t\phi$ is defined as $tan(\phi)$.

A.2 Rotation matrix derivatives

The Jacobian of the rotation matrices $\boldsymbol{R}_b^w$ and $\boldsymbol{C}_b^w$ with respect to the Euler angles (ϕ, θ, ψ) are three element matrices defined as

$$\hat{\boldsymbol{R}}_b^w = \frac{\partial \boldsymbol{R}_b^w}{\partial(\phi,\theta,\psi)} = \left[\frac{\partial \boldsymbol{R}_b^w}{\partial \phi}, \frac{\partial \boldsymbol{R}_b^w}{\partial \theta}, \frac{\partial \boldsymbol{R}_b^w}{\partial \psi},\right] \qquad \hat{\boldsymbol{C}}_b^w = \frac{\partial \boldsymbol{C}_b^w}{\partial(\phi,\theta,\psi)} = \left[\frac{\partial \boldsymbol{C}_b^w}{\partial \phi}, \frac{\partial \boldsymbol{C}_b^w}{\partial \theta}, \frac{\partial \boldsymbol{C}_b^w}{\partial \psi},\right] \tag{A.4}$$

$$\frac{\partial \boldsymbol{R}_b^w}{\partial \phi} = \begin{bmatrix} 0 & s\phi\, s\psi + c\phi\, s\theta\, c\psi & c\phi\, s\psi - s\phi\, s\theta\, c\psi \\ 0 & -s\phi\, c\psi + c\phi\, s\theta\, c\psi & -c\phi\, c\psi - s\phi\, s\theta\, s\psi \\ 0 & c\phi\, c\theta & -s\phi\, c\theta \end{bmatrix} \tag{A.5}$$

$$\frac{\partial \boldsymbol{R}_b^w}{\partial \theta} = \begin{bmatrix} -s\theta\, c\psi & s\phi\, c\theta\, c\psi & c\phi\, c\theta\, c\psi \\ -s\theta\, s\psi & s\phi\, c\theta\, c\psi & c\phi\, c\theta\, s\psi \\ -c\theta & -s\phi\, s\theta & -c\phi\, s\theta \end{bmatrix} \tag{A.6}$$

$$\frac{\partial \boldsymbol{R}_b^w}{\partial \psi} = \begin{bmatrix} -c\theta\, s\psi & -c\phi\, c\psi - s\phi\, s\theta\, s\psi & s\phi\, c\psi - c\phi\, s\theta\, s\psi \\ c\theta\, c\psi & -c\phi\, c\psi - s\phi\, s\theta\, s\psi & s\phi\, s\psi + c\phi\, s\theta\, c\psi \\ 0 & 0 & 0 \end{bmatrix} \tag{A.7}$$

$$\frac{\boldsymbol{C}_b^w}{\partial\phi} = \begin{bmatrix} 0 & c\phi\, s\theta/c\theta & -s\phi\, s\theta/c\theta \\ 0 & -s\phi & -c\phi \\ 0 & c\phi/c\theta & -s\phi/c\theta \end{bmatrix} \tag{A.8}$$

$$\frac{\boldsymbol{C}_b^w}{\partial\theta} = \begin{bmatrix} 0 & s\phi/c^2\theta & c\phi/c^2\theta \\ 0 & 0 & 0 \\ 0 & s\phi\, s\theta/c^2\theta & -c\phi\, s\theta/c^2\theta \end{bmatrix} \tag{A.9}$$

$$\frac{\boldsymbol{C}_b^w}{\partial\psi} = \mathbf{0}. \tag{A.10}$$

A.3 Sensor fusion circuit schematic

The schematic shown in Figure A.1 has been designed with EagleCAD. The circuit consist out of four HC-SR04 ultrasonic sensors, a LSM9DS0 IMU, an Arduino NANO, two 100 μF capacitors and one IC 7085 voltage regulator. The system can handle voltage upto 9-12v, with a recommended voltage of 9v. Additionally, the RX/TX connection of the Arduino NANO can be attached to an data-logger circuit for serial communication.

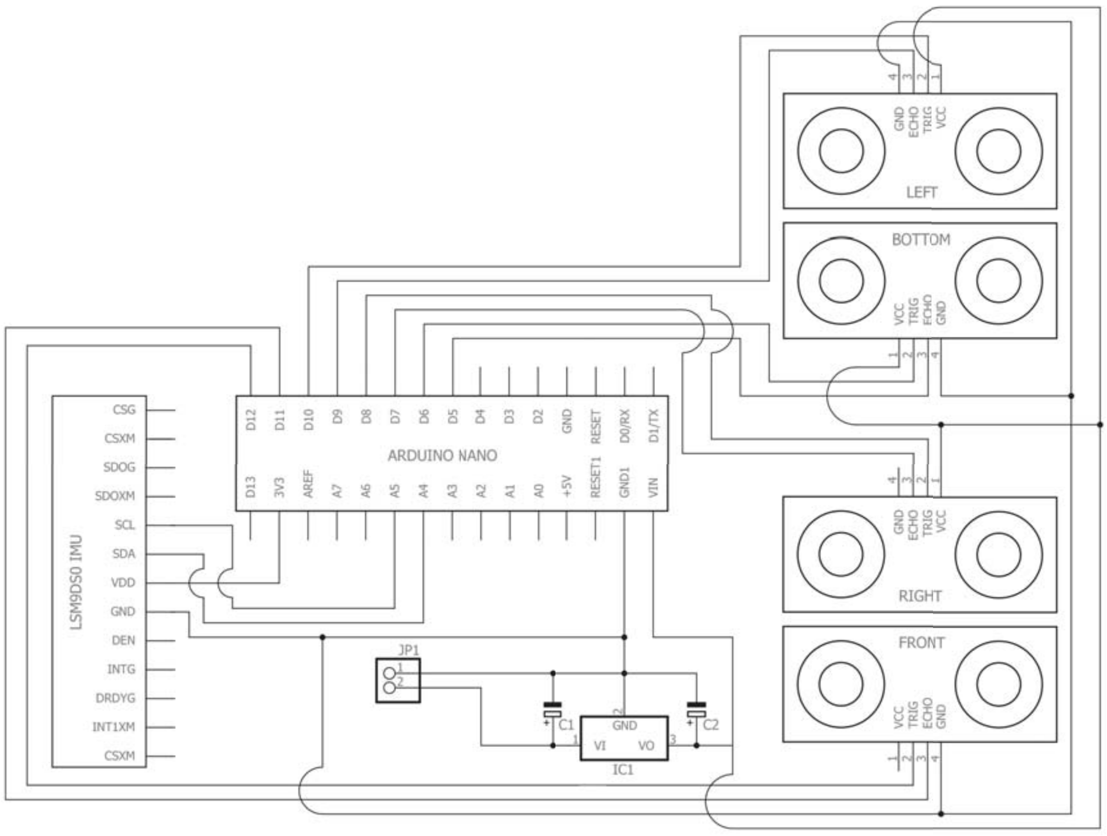

LEFT
BOTTOM
RIGHT
FRONT
GND ECHO TRIG VCC
VCC TRIG ECHO GND
ARDUINO NANO
D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 GND RESET D0/RX D1/TX
D13 3V3 AREF A7 A6 A5 A4 A3 A2 A1 A0 +5V RESET1 GND1 VIN
LSM9DS0 IMU
CSG CSXM SDOG SDOXM SCL SDA VDD GND DEN INTG DRDYG INT1XM CSXM
JP1
C1
C2
GND VI VO
IC1

Figure A.1: *Schematic of sensor fusion circuit board.*

A.4 MATLAB Sensor Fusion code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%— Extended Kalman Filter Algorithm for free body pose estimation —%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Requirements: file_name.mat — data file, with A = [fx,fy,fz] acceleration
% in (g), G = [wx,wy,wz] angular rate in (rad/s), Sx, Sy, Sz SONAR
% measurements in (m), O = [phi, theta, psi] orientation in (rad)
% and t time vector in (s).
%
% Provided: covariance_and_drift.mat — data file with covarainces and drift
% values of HC—SR04 SONAR unit and LSMD90F IMU. Values can be tweak with
% the parameters alpha, beta, and gamma.
%
% EKF_pose.m provides an algorithm that estimates the position p, velocity
% v, and orientation theta in 3D space. Based SONAR measurement in x,y, and
% z — direction, plus measured orientation roll, pitch and yaw.
```

```
%
% Moreover, if the sampling of the particular IMU measurement is sufficient
% , the algorithm is able to propagate displacement, velocity and
% orientatation quite well.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% initialize
clc; clear;% close all;

%% rough tweaking of Q and R

a = 20; b = 1; c = 10;
%% fine tweaking of Q and R

alpha = a*[1,1,1,...
           1,1,1];

beta = b*[0.1,0.1,0.5,...
           0.5,0.5,1];

gamma = c*[1,1,1,...
           1,1,1];
%% load data

clc;
load('data_3d_half_circle2');
disp('measurement data loaded');

load('covariance_and_drift');
disp('covariance and drift data loaded')

RAD_TO_DEG = 180/pi;
DEG_TO_RAD = pi/180;

I3 = eye(3,3);
O3 = zeros(3,3);
O6 = zeros(6,6);
g = 9.81;

%% data split

gx_b = g*A(:,1);                                                          % split ...
    measure accel and correct to m/s^2
gy_b = g*A(:,2);                                                          % ''
gz_b = g*A(:,3);                                                          % ''

omegax_b = G(:,1);                                                        % split ...
    measure gyro
omegay_b = G(:,2);                                                        % ''
omegaz_b = G(:,3);                                                        % ''

Phi = O(:,1);                                                             % split ...
    measured angle
Theta = O(:,2);                                                           % ''
```

```
Psi = (O(:,3));                                                    % ''

dt = diff(t);                                                      % ...
    determine all time steps

%% bias compensation
n = 10;                                                            % amount...
     of data points for drift determination

dfx_0 = mean(gx_b(1:n));                                           % bias ...
    compensation for acceleration
dfy_0 = mean(gy_b(1:n));                                           % ''
dfz_0 = mean(gz_b(1:n));                                           % ''


domegax_0 = mean(omegax_b(1:n));                                   % bias ...
    compensation for gyroscope
domegay_0 = mean(omegay_b(1:n));                                   % ''
domegaz_0 = mean(omegaz_b(1:n));                                   % ''

disp('IMU bias compensation done')

%% covariances

Qk{1} = diag(alpha)*[diag(cov_accel), O3;                          % fill Q...
     with std values
                     O3, diag(cov_gyro)];                          % ''

Rk{1} = diag(beta)*(diag([cov_sonar, cov_sonar, cov_sonar, cov_theta,...   % fill R...
     with std values
    cov_theta, cov_pzi]));                                         % ''

                       Qk{1} = Qk{1}.^2;                           % square...
                            the std
                       Rk{1} = Rk{1}.^2;                           % ''

DRIFT = diag(gamma)*diag([abs(drift_accel), abs(drift_gyro)]);

Qk{1} = [Qk{1}, O6;                                                % fill Q...
     with drift values
         O6, DRIFT];                                               % ''

%% initialize state vector x = [p, v, theta, ∆ f, ∆ omega]

xk{1} = 0.0*ones(15,1);                                            % create...
     zero vector state x

xk{1}(1) = Sx(1);                                                  % fill ...
    first SONAR measurement
xk{1}(2) = Sy(1);                                                  % ''
xk{1}(3) = 0*Sz(1);                                                  % ''

xk{1}(10) = dfx_0;                                                 % fill ...
    first drift measurement
xk{1}(11) = dfy_0;                                                 % ''
xk{1}(12) = dfz_0 - 9.81;                                          % ''
```

```

xk{1}(13) = domegax_0;                                                    % ''
xk{1}(14) = domegay_0;                                                    % ''
xk{1}(15) = domegaz_0;                                                    % ''

%% initialize covariance matrix

Pk{1} = 0.1*eye(15,15);                                                   % ...
    initialize covariance matrix

%% initialize measurement model y = Hx;

Hk{1} = [I3, O3, O3, O3, O3;                                              % form ...
    measurement model matrix H
         O3, O3, I3, O3, O3];

%% Extended Kalman Algorithm
i = 1; n = 1; k_ = 1;

disp('preforming EKF algorithm');

for k = 2:n:length(dt)                                                    % for 1...
    :end of measurement

    i = i + 1; steps = i;                                                 % update...
         count

    t_dis(i) = t(k_); dts = t(k) - t(k_);                                 % set ...
        time and time step

    IMU{i-1} = [gx_b(k_)-xk{k_}(10), gy_b(k_)-xk{k_}(11),...              % get ...
        IMU data, and correct for bias
        gz_b(k_)-xk{k_}(12), omegax_b(k_)-xk{k_}(13),...                  % ''
        omegay_b(k_)-xk{k_}(14), omegaz_b(k_)-xk{k_}(15)];                % ''

    xk{i} = zeros(9,1); xk{i}(10:15) = zeros(6,1);                        % ...
        initalize state k+1

    xk{i}(1:9) = propagation_function(xk{i-1}(1:9),IMU{i-1},dts);         % ...
        propagate states p,omega and theta

    xk{i}(10) = drift_accx*dts;                                           % ...
        propagate drift after f = f - Δ f;
    xk{i}(11) = drift_accy*dts;                                           % ''
    xk{i}(12) = drift_accz*dts;                                           % ''

    xk{i}(13) = drift_gyrox*dts;                                          % ...
        propagate drift after omega = omega - Δ omega;
    xk{i}(14) = drift_gyroy*dts;                                          % ''
    xk{i}(15) = drift_gyroz*dts;                                          % ''

    % MEASUREMENT UPDATE

    if isnan(Sx(k_)) == 0 || isnan(Sy(k_)) == 0 || isnan(Sz(k_)) == 0     % check ...
        if SONAR data is available

        z{i} = [Sx(k_); Sy(k_); Sz(k_); Phi(k_); Theta(k_); Psi(k_)];        % form ...
            measurement zk

        eps{i} = z{i} - Hk{1}*xk{i};                                          % form ...
            innovation epsilon

        Sk = Hk{1}*Pk{i-1}*transpose(Hk{1}) + Rk{1};                          % ...
            compute innovation covariance S

        Kk{i} = Pk{i-1}*transpose(Hk{1})*inv(Sk);                             % ...
            compute Kalman gain K

        dx{i} = Kk{i}*eps{i};                                                 % ...
            compute Kalman update dx

        Pk{i-1} = (eye(15,15) - Kk{i}*Hk{1})*Pk{i-1};                         % ...
            correct covariance P

        xk{i} = xk{i} + dx{i};                                                % ...
            correct states

    else

        eps{i} = [NaN; NaN; NaN; NaN; NaN; NaN].';                            % no - ...
            innovation

        dx{i} = zeros(15,1);                                                  % no - ...
            kalman update

        Pk{i} = Pk{i-1};                                                      % ...
            maintain covariance

        xk{i} = xk{i};                                                        % ...
            maintain states

    end

    [Fk{i}, Gk{i}] = update_jacobian(xk{i},IMU{i-1},dts);                     % update...
         Jacobian matrix

    Pk{i} = Fk{i}*Pk{i-1}*transpose(Fk{i}) + Gk{i}*Qk{1}*transpose(Gk{i});   % ...
        predict the new covariance matrix

    k_ = k;                                                                   % set ...
        the old time step

end

dx{1} = zeros(15,1); eps{1} = zeros(6,1); IMU{k} = IMU{k-1};                 % fill ...
    empty data

disp('- EKF algorithm done');

%% estimation splitting

i = 1; n = 1;
```

```
for k = 1:steps

    px(i) = xk{k}(1);
    py(i) = xk{k}(2);
    pz(i) = xk{k}(3);

    vx(i) = xk{k}(4);
    vy(i) = xk{k}(5);
    vz(i) = xk{k}(6);

    phie(i) = xk{k}(7);
    thetae(i) = xk{k}(8);
    psie(i) = xk{k}(9);

    dfx(i) = xk{k}(10);
    dfy(i) = xk{k}(11);
    dfz(i) = xk{k}(12);

    dwx(i) = xk{k}(13);
    dwy(i) = xk{k}(14);
    dwz(i) = xk{k}(15);

    i = i+1;
end

disp('plotting results')

figure('Name','Position');
subplot(3,1,1);
plot(t_dis,px); hold on; plot(t,Sx);
ylabel('$p^w_x$ (m)','Interpreter','Latex','Fontsize',14); grid on;
axis tight
subplot(3,1,2);
plot(t_dis,py); hold on; plot(t,Sy);
ylabel('$p^w_y$ (m)','Interpreter','Latex','Fontsize',14); grid on;
axis tight
subplot(3,1,3);
plot(t_dis,pz); hold on; plot(t,Sz);
ylabel('$p^w_z$ (m)','Interpreter','Latex','Fontsize',14); grid on;
axis tight
xlabel('time (s)','Interpreter','Latex','Fontsize',14);

figure('Name','Orientation');
subplot(3,1,1);
plot(t_dis,phie); hold on; plot(t,Phi);
ylabel('$\phi$ (rad)','Interpreter','Latex','Fontsize',14); grid on;
axis tight
subplot(3,1,2);
plot(t_dis,thetae); hold on; plot(t,Theta);
ylabel('$\theta$ (rad)','Interpreter','Latex','Fontsize',14); grid on;
axis tight
subplot(3,1,3);
plot(t_dis,psie); hold on; plot(t,Psi);
ylabel('$\psi$ (rad)','Interpreter','Latex','Fontsize',14); grid on;
axis tight
xlabel('time (s)','Interpreter','Latex','Fontsize',14);
```

```
function [Fk, Gk] = update_jacobian(xk_,IMU,dt)
px = xk_(1);
py = xk_(2);
pz = xk_(3);
vx = xk_(4);
vy = xk_(5);
vz = xk_(6);
phi = xk_(7);
theta = xk_(8);
pzi = xk_(9);

fx = IMU(1);
fy = IMU(2);
fz = IMU(3);

wx = IMU(4);
wy = IMU(5);
wz = IMU(6);

R = Rot([phi,theta,pzi]);
C = Rotdot([phi,theta,pzi]);

Fk(:,1) = [ 1, 0, 0, 0, 0, 0, 0, 0, 0].';
Fk(:,2) = [ 0, 1, 0, 0, 0, 0, 0, 0, 0].';
Fk(:,3) = [ 0, 0, 1, 0, 0, 0, 0, 0, 0].';
Fk(:,4) = [ dt, 0, 0, 1, 0, 0, 0, 0, 0].';
Fk(:,5) = [ 0, dt, 0, 0, 1, 0, 0, 0, 0].';
Fk(:,6) = [ 0, 0, dt, 0, 0, 1, 0, 0, 0].';
Fk(:,7) = [ 0, 0, 0, 0, dt*(fx*(sin(phi)*sin(pzi) + cos(phi)*cos(pzi)*sin(theta)) ...
    ...
- fy*(cos(pzi)*sin(phi) - cos(phi)*sin(pzi)*sin(theta)) + fz*cos(phi)*cos(theta)),...
    ...
 -dt*(fy*(cos(phi)*cos(pzi) + sin(phi)*sin(pzi)*sin(theta)) - fx*(cos(phi)*sin(pzi)...
     ...
 - cos(pzi)*sin(phi)*sin(theta)) + fz*cos(theta)*sin(phi)), dt*((wy*cos(phi)*sin(...
     theta))/cos(theta)....
 - (wz*sin(phi)*sin(theta))/cos(theta)) + 1, -dt*(wz*cos(phi) + wy*sin(phi)),...
 dt*((wy*cos(phi))/cos(theta) - (wz*sin(phi))/cos(theta))];
Fk(:,8) = [ 0, 0, 0, -dt*(fz*cos(theta) + fx*cos(pzi)*sin(theta) + fy*sin(pzi)*sin(...
    theta)),...
 dt*(fx*cos(pzi)*cos(theta)*sin(phi) - fz*sin(phi)*sin(theta) + fy*cos(theta)*sin(...
     phi)*sin(pzi)),...
 dt*(fx*cos(phi)*cos(pzi)*cos(theta) - fz*cos(phi)*sin(theta) + fy*cos(phi)*cos(...
     theta)*sin(pzi)),...
 dt*(wz*cos(phi) + wy*sin(phi) + (wz*cos(phi)*sin(theta)^2)/cos(theta)^2 + (wy*sin(...
     phi)*sin(theta)^2)/cos(theta)^2),...
 1, dt*((wz*cos(phi)*sin(theta))/cos(theta)^2 + (wy*sin(phi)*sin(theta))/cos(theta)...
     ^2)];
Fk(:,9) = [ 0, 0, 0, dt*(fy*cos(pzi)*cos(theta) - fx*cos(theta)*sin(pzi)),...
 -dt*(fx*(cos(phi)*cos(pzi) + sin(phi)*sin(pzi)*sin(theta)) + fy*(cos(phi)*sin(pzi)...
     ...
 - cos(pzi)*sin(phi)*sin(theta))), dt*(fx*(cos(pzi)*sin(phi) - cos(phi)*sin(pzi)*sin...
     (theta))...
 + fy*(sin(phi)*sin(pzi) + cos(phi)*cos(pzi)*sin(theta))), 0, 0, 1];

```

```
Fk(10,10) = 1.0; Fk(11,11) = 1.0; Fk(12,12) = 1.0; Fk(13,13) = 1.0; Fk(14,14) = 1.0;...
     Fk(15,15) = 1.0;

Fk(4:6,10:12) = dt*R;
Fk(7:9,13:15) = dt*C;

Gk(:,1) = [ 0, 0, 0, dt*cos(pzi)*cos(theta), -dt*(cos(phi)*sin(pzi) - cos(pzi)*sin(...
    phi)*sin(theta)), dt*(sin(phi)*sin(pzi) + cos(phi)*cos(pzi)*sin(theta)), 0, 0, ...
    0].';
Gk(:,2) = [ 0, 0, 0, dt*cos(theta)*sin(pzi), dt*(cos(phi)*cos(pzi) + sin(phi)*sin(...
    pzi)*sin(theta)), -dt*(cos(pzi)*sin(phi) - cos(phi)*sin(pzi)*sin(theta)), 0, 0, ...
    0].';
Gk(:,3) = [ 0, 0, 0, -dt*sin(theta), dt*cos(theta)*sin(phi), dt*cos(phi)*cos(theta),...
     0, 0, 0].';
Gk(:,4) = [ 0, 0, 0, 0, 0, 0, dt, 0, 0].';
Gk(:,5) = [ 0, 0, 0, 0, 0, 0, (dt*sin(phi)*sin(theta))/cos(theta), dt*cos(phi), (dt*...
    sin(phi))/cos(theta)].';
Gk(:,6) = [ 0, 0, 0, 0, 0, 0, (dt*cos(phi)*sin(theta))/cos(theta), -dt*sin(phi), (dt...
    *cos(phi))/cos(theta)].';

Gk(10,7) = dt; Gk(11,8) = dt; Gk(12,9) = dt; Gk(13,10) = dt; Gk(14,11) = dt; Gk...
    (15,12) = dt;
end
```

```

function R = skew(w)

  if 3 ≠ size(w,1),
    error('SCREWS:skew','vector must be 3x1')
  end

  if isnumeric(w),
    R = zeros(3,3);
  end

  R(1,2) = -w(3);
  R(1,3) =  w(2);
  R(2,3) = -w(1);

  R(2,1) =  w(3);
  R(3,1) = -w(2);
  R(3,2) =  w(1);

end
```

```

function DMC = Rot(x)

    phi = x(1);
    theta = x(2);
    pzi = x(3);

    DMC = [cos(theta)*cos(pzi), cos(theta)*sin(pzi), -sin(theta);
        sin(phi)*sin(theta)*cos(pzi)-cos(phi)*sin(pzi), sin(phi)*sin(theta)*sin(pzi)...
            +cos(phi)*cos(pzi), sin(phi)*cos(theta);
        cos(phi)*sin(theta)*cos(pzi)+sin(phi)*sin(pzi), cos(phi)*sin(theta)*sin(pzi)...
            −sin(phi)*cos(pzi), cos(phi)*cos(theta)];

end
```

```

function Racc = Rotdot(x)

    phi = x(1);
    theta = x(2);

    Racc = [1, (sin(phi)*sin(theta))/cos(theta), (cos(phi)*sin(theta))/cos(theta);
            0, cos(phi),    −sin(phi);
            0, sin(phi)/cos(theta), cos(phi)/cos(theta)];
end
```

A.5 SIMULINK Virtual Model

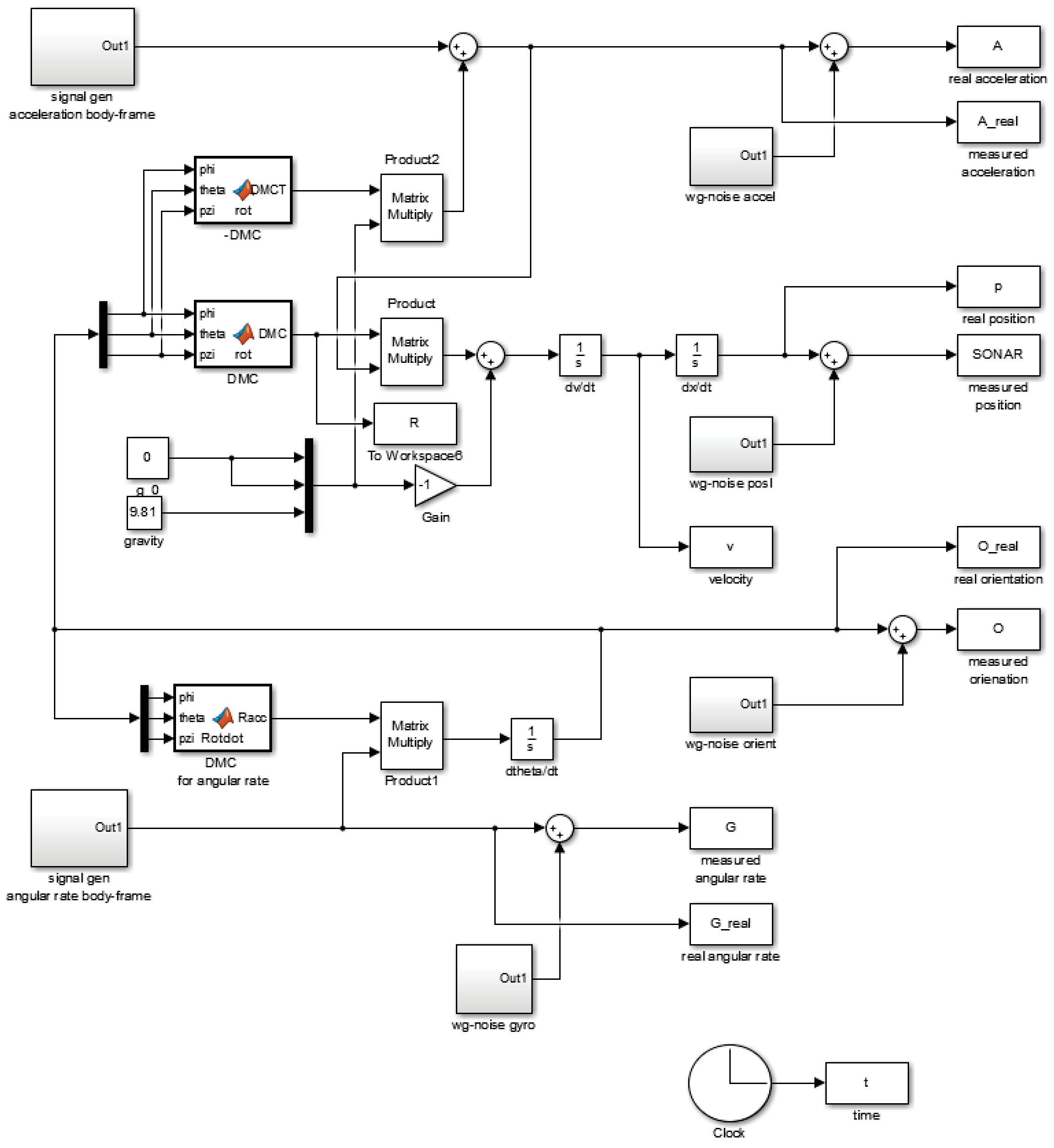


Figure A.2: *SIMULINK model that can simulate a virtual enviroment for UAV type movement.*