UPPSALA
UNIVERSITET

CANDIDATE PROJECT

GPS and IMU Sensor Fusion to Improve Velocity Accuracy

Independent Project Work in Electrical Engineering, 15hp

Adam Laurell

Erik Karlsson

Yousuf Naqqar

Supervisor(s):
Rasmus Hamrén
Mats Ekberg

Course co-ordinator:
Mats Ekberg

Examiner:
Mikael Bergkvist

ELEKTRO-E 22006

June, 2022

Preface and Acknowledgement (Adam, Erik)

First of all, we are especially grateful to Rasmus Hamrén, our supervisor at NEP (Nordic Electronic Partner), for coordinating us through the project and giving valuable insights on project management and the engineering industry.

We would also like to thank the course leader Mats Ekberg at Uppsala University (UU) for holding several informative seminars and arranging the course.

Special thanks goes to Mälardalens University (MDU), for lending us the Pixhawk 4 with GPS module, along with offering many other resources.

We also extend our gratitude to the company Qtagg for providing us with a workspace and technical insights. Of the staff at Qtagg, Andreas Domfors deserves an extra praise for giving us helpful technical information.

Finally we would like to thank Anders Söderbärg, CEO at NEP, for providing us with this fantastic opportunity. It has helped us improve in both technical aspects and in performing and managing project work.

Abstract (Erik, Adam)

The project explores the possibilities on how to improve the accuracy of GPS velocity data by using sensor fusion with an extended Kalman filter.

The proposed solution in this project is a sensor fusion between the GPS and IMU of the system, where the extended Kalman filter was used to estimate the velocity from the sensor data. The hardware used for the data acquisition to the proposed solution was a Pixhawk 4 (PX4), which has an IMU consisting of accelerometers, gyroscopes and magnetometers. The PX4:s corresponding GPS module was also used to collect accurate velocity data. The data was logged using Simulink and later processed with MATLAB.

The sensor fusion using the extended Kalman filter gave good estimates upon constant acceleration but had problems with estimating over varying acceleration. This was initially planned to be solved using smoothing filters, which is an essential part of the fusion process, but was never implemented due to time constraints. The constructed filter acts as a foundation towards future improvement.

Other methods such as unscented Kalman filter, particle filter and neural network could also be explored to improve the estimation of the velocity due to these filters being known to have better performance. However, most of these alternatives need more computing power and are generally harder to implement compared to the extended Kalman filter.

This project would be beneficial to QTAGG, since increasing the velocity resolution and accuracy of the system can provide possibilities of better optimization. It is also a commonly implemented solution where there are many state of the art implementations available.

Keywords: Sensor fusion, GPS navigation, Extended Kalman filter, Control systems, Velocity estimation, Data processing

Contents

Nomenclature

Terms

insfilterAsync	MATLAB function used to construct an asynchronous EKF
MATLAB	Programming language and numeric computing platform
Simulink	MATLAB-based graphical programming environment
uORB	Asynchronous API used for communication on board the Pixhawk 4

Acronyms

AHRS	Attitude and Heading Reference System
AI	Artificial intelligence
DOP	Dilution of Precision
EKF	Extended Kalman Filter
GPS	Global Positioning Unit
IMU	Inertial Measurement Unit
KF	Kalman Filter
NED	North, East, Down
PX4	Pixhawk 4, (flight controller used for the data acquisition)
RMS	Root Mean Square
SOG	Speed Over Ground
STW	Speed Through Water
UKF	Unscented Kalman filter

Variables

g	Gravitational acceleration at a local point on Earth's surface
a	Acceleration
θ	Roll angle
φ	Pitch angle
v_{NED}	Velocity represented within the NED-frame
v_{xyz}	Velocity represented within the xyz-frame
$\hat{v}_i$	Velocity estimate at the discrete time sample i
P	State error covariance matrix
b	Bias

1 Introduction

1.1 Problem description (Erik, Adam)

This project addresses the issue of improving velocity estimations through sensor fusion. The purpose of this project is to find a way to improve the resolution along with the accuracy of the velocity on various ships using the same control system.

Solving this problem would not only be beneficial towards route optimization but also to the customers, due to hardware replacement being more expensive than using a software implementation. To backup these claims, many companies that are in the industry for vehicles have already implemented sensor fusion in their systems, in many aspects the implementation of this solution should be done.

1.2 Background

This project was provided by the engineering agency NEP and focuses on providing Qtagg with improved velocity accuracy to be used in their system. Qtagg is a Swedish Company, which develops advanced control systems for the marine market. The purpose of their control systems are to reduce ship dynamics, which results in decreasing the fuel consumption of ships and increase the life expectancy for different parts of the ship such as rudder, propeller and engine.

It is achieved by using different sensors, that collects a variety of data. This data is then used in simulation where it selects the outcome that would reduce the ship dynamics the most, while also upholding the requirement of the desired time in which the vessel should arrive to its desired destination.

One of the key variables for Qtagg:s algorithm is velocity, which in its current state has a poor accuracy. This is because Qtagg is currently using consumer-level GPS modules to estimate the velocity of their ships. However, their velocity could be improved due to their control systems being equipped with an IMU. By combining the GPS and IMU through sensor fusion, then accuracy of the velocity could be improved. To our knowledge, it has currently not been implemented. If the accuracy, or more specifically the resolution, was improved then it would in turn raise the accuracy and resolution of the other dependent variables. With higher resolution and accuracy, then more precise optimizations can be determined.

Initially the group did not have sufficient knowledge about AI and machine learning. Therefore, it was not feasible for the group to perform sensor fusion using any complex machine learning algorithms such as constructing and training a neural network. Since the group have previously had courses in measurement technology and control engineering it was more suitable for the group to handle filter algorithms such as Kalman filters. Luckily the extended Kalman filter appeared to be a strong candidate for sensor fusion between GPS and IMU. There were also some time constraints present since the deadline was about two months from start. Since most of the applications was new to the group, a lot of time had to be spent on the initial studying phase until any technical implementation could be done.

2 Theory

2.1 GPS (Erik)

GPS (Global Positioning System) is a satellite-based navigation system, its primary purpose is to give a precise location to a GPS receiver. It is commonly known that GPS was originally intended for military uses. However, the application for using the GPS later became open for both industries and consumers usage. Today the usage of GPS is mainly used for tracking of both unmanned and manned vehicles. For the GPS to function it needs satellites and a GPS receiver. If one wanted to improve the accuracy then the usage of ground control stations can be used, which purpose is to act as an additional reference point for the GPS receiver.

The GPS can track the location of the GPS receiver by satellites that orbits the earth. These satellites send out different radio frequencies, which are used as reference points for GPS receiver to figure out its location. For the GPS receiver to work it needs at least access to 4 satellites. it is then able to determine parameters such as time, location, speed, and direction etc. The way a GPS receiver determines theses parameters is by using trilateration, time difference, doppler and signal decoding.

Trilateration and time difference is used to locate the location. The principle of trilateration is that the signals of at least three satellites will intersect with each other, this results in a point that allows one to determine the location. Time difference is also used to figure out the location, this is done by measuring the time it takes for the signal to arrive to the GPS receiver. Due to radio signals traveling at constant speed the receiver can then easily calculate the distance to each satellite [1].

The doppler effect is a known classical of physics, thus results in the law being quite well founded and accurate. The doppler method is rather straightforward as the doppler effect become stronger when being near the receiver and becomes weaker when the satellites recede from the receiver. With the doppler effect one can then determine the speed of an object [2]. The radio signals that the GPS satellites sends out also have specific codes. This code can be decoded to give information such as time and date [1].

This results in the GPS being a system that is quite accurate. However, the accuracy of the location also depends on the satellite's location. This is given in a parameter known as DOP (dilution of precision) and specifies the error relative to the GPS satellites, where low values of DOP are considered good and higher values considered as bad. The value of DOP depends on different conditions such as the geometrics of the satellites, weather conditions, jamming, radio emissions and the number of visible satellites [3].

2.2 IMU (Erik)

IMU (Inertial Measurement Unit) is an electrical device, which can be used to measure angular velocity and acceleration. It combines the sensors of accelerometers, gyroscopes, and magnetometers into one sensor. The accelerometer gives acceleration as a parameter and with time derivative can also be used to give parameters such as velocity and distance. The gyroscope gives the orientation and angular velocity of the sensor. The magnetometer is an optional sensor that one can include into the IMU, it is often implemented for speed detection and to eliminate the gyro offset.

With a IMU one can then calculate parameters such as linear velocity and position if one were to have a global reference point. IMU have applications in both industrial and consumer electronics.

Where in its applications it's often used for tracking movement and to stabilize vehicles.

2.3 Kalman Filters (Erik)

Kalman filtering (KF) is an algorithm that is used to estimates measurements observed over time. Kalman filtering is generally used in control systems and statistics. Since the development of the first Kalman filter, several different versions of Kalman filter have been created. Extensions and general concepts that have been developed are filters such as the extended Kalman filter and the unscented Kalman filter, which is used on nonlinear systems, where the Kalman filter is applied on linear systems [4].

The extended Kalman filter (EKF) is a version of the Kalman filter (KF) and is used for nonlinear application. The EKF is often used for more non linear applications in navigation systems, such as providing position or velocity estimations from the acceleration readings from an accelerometer.

There are three general differences that differentiates the EKF from the KF. The first one being that the EKF uses a nonlinear state update. The second is that the state Jacobian replaces the state transition matrix of the KF. The third difference is that the measurement Jacobian replaces the measurement matrix of the KF.

The extended Kalman filter has three stages. The first stage is the initialization stage, this is the initialization of the filter and is essential due to the system needing to have a starting point which it can start assuming from. The initialization stage will thus only be used for the initialization of the EKF, and then the other stages will enter a continuous loop. The second stage is the prediction stage, this stage is used to predict the next state estimate and covariance estimate. The third stage is the correction stage, which is used to updates the state estimate and covariance estimate [4].

The Mathematical model of the extended Kalman filter follows steps 0 to 8 once, steps 1 to 8 then repeat indefinitely

Initialization

(0) Initial estimates: $x_{0|0}, P_{0|0}$

Prediction

(1) Predicted state estimate: $x_{i+1|i} = f(x_{i|i}, u^i)$

(2) Predicted error covariance estimate: $P_{i+1|i} = F_i^{(x)} P_{i|i} F_i^{(x)T} + F^{(n)} Q_i F^{(n)T}$

(3) Measurement: $y_{i+1|i} = h(x_{i+1|i})$

Correction

(4) Innovation covariance: $S_{i+1} = H_{i+1}^{(x)} P_{i+1|i} H_{i+1}^{(x)T} + H^{(\omega)} R_{i+1} H^{(\omega)T}$

(5) Compute the Kalman gain: $K_{i+1} = P_{i+1|i} H_{i+1}^{(x)T} S_{i+1}^{-1}$

(6) Update the state estimate: $x_{i+1|i+1} = x_{i+1|i} + K_{i+1}(y_{i+1} - y_{i+1|i})$

(7) Update the error covariance: $P_{i+1|i+1} = P_{i+1|i} - K_{i+1} S_{i+1} K_{i+1}^T$

(8) New measurement: $y_{i+1|i+1} = h(x_{i+1|i+1})$

x is the state estimate, y is the measurement, P is the error covariance, F is the state Jacobian, H is the observation Jacobian, K is the Kalman gain [4], n is the measurement noise and ω is the random noise perturbation.

2.4 Sensor Fusion (Erik)

Sensor fusion (or data fusion) is the method of merging two or more sensors, this results in data that can increase parameters such as certainty, range, accuracy and reliable of the data. When using sensor fusion algorithms such as extended Kalman filter is also used in the process. It should be noted that the data that one fuses should contain the same type of information.

The resulting data which one gets for the sensor fusion depends on the weight that is put on each sensor. The weight of the sensors is usually dependent on parameters, such as the sensors environmental workability, resolution, sensitivity, sample frequency, range, etc. For example, if a car were to have a proximity warner, it could fuse a long-range sensor with low precision, and a short-range sensor with high precision to create a sensor that is high range and high precision. The weight of which sensors should be used then depends on the distance [5].

The advantages of using sensor fusion is that it can use the strength of different sensors and if one sensor were to fall through or act up, then the other sensor/sensors can be used as a fail-safe to ensure control and that the whole system doesn't break down [6].

However, the usage of sensor fusion can come with hurdles such as that the complexity of the system increases. This implies that if data from a sensor have imperfections or inconsistency, then it could ruin the sensor fusion. Thus, data must be handled carefully when implementing sensor fusion [6].

3 Methods

3.1 Data acquisition (Erik, Adam)

The hardware used for the project were a Pixhawk 4 with its designated GPS module. The Pixhawk 4 is an open-source flight controller for drones other unmanned vehicles. The Pixhawk 4 is equipped with various sensors such as accelerometer, gyroscope magnetometer, and is coupled with a GPS module that provides location and speed measurement.

By looking at the datasheets of the components of the Pixhawk 4, performance specifications such as noise and bias can be assumed. The IMU in the PX4 module is a BMI055, the datasheet [7] gives values on the bias and noise for both the accelerometer and gyroscope in the BMI055. The accelerometer noise and bias are $150\mu g/\sqrt{Hz}$ and $\pm 70mg$. The gyroscope noise and bias are $0.014^{\circ}/s/\sqrt{Hz}$ and $\pm 1^{\circ}/s$. The magnetometer used in the PX4 is a IST8310 [8], with a noise value of $\pm 1600\mu T$ (XY), $\pm 2500\mu T$ (Z) and a bias of $\pm 0.3\mu T$. The given values represent the specifications at room temperature ($T = 25C^{\circ}$). Note that the datasheets often vary between squared values or absolute values, it is also normal to refer to the bias as zero Gauss offset.

The data was collected using Simulink. The toolbox *UAV Toolbox Support Package for PX4 Autopilots*, developed for communicating with the PX4 module, was used in Simulink to read and log the sensors. Simulink provides many parameters that can be customized, such as the sampling frequency for each sensor. The data was collected in a higher sampling frequency such as 10hz or 30hz compared to the designated 1hz.

The Simulink program acquires the data through uORB, which is a messaging API used for inter-thread/inter-process communication. This can provide data that is already estimated internally through the Pixhawk 4 internal software, such as quaternions [9].

To ensure that the data were not affected by environmental disturbance, a board was constructed to ensure the stability of the sensors. This was done by placing the Pixhawk 4 and GPS module on a plexiglass board with some Velcro between the components and the board for easy mounting. A plastic covering over the ports of the Pixhawk 4 was also applied to avoid damages such as moisture. The board was then tapered to the roof of a car with the component's direction heading towards the bonnet of the car. This was done to ensure that the GPS module would have clear clearance to access the satellites and thus get the best accuracy.

Figure 1: PX4 module mounted with GPS sensor.

3.2 Physical model and general approach (Adam)

The pose of a moving object within a reference frame can be determined through fundamental kinematics. In a one-dimensional frame the velocity of an object in a relative moment in time can be determined through the following equation:

$$v_{i+1} = v_i + a_i \cdot \Delta t \tag{1}$$

Where i represents the initial state, $i + 1$ is the momentaneous state after the time that elapsed from the initial state, which is represented as Δt. Note that a is assumed as constant acceleration. When expanded to three dimensions, represented in the xyz-frame, the following equations are obtained:

$$\begin{aligned} v_{x,i+1} &= v_{x,i} + a_{x,i} \cdot \Delta t \\ v_{y,i+1} &= v_{y,i} + a_{y,i} \cdot \Delta t \\ v_{z,i+1} &= v_{z,i} + a_{z,i} \cdot \Delta t \end{aligned} \tag{2}$$

However, this is not applicable in a real-life scenario as there are more dynamics introduced. The data that is acquired from the sensors will have errors and noise present, thus providing an uncertainty in the measurements. The systematic error in the measurements is seen as a bias and the noise can be approximated as white noise [10]. If the bias from the noise present in the acceleration data is considered, the following three-axis states are obtained:

$$\begin{aligned} v_{x,i+1} &= v_{x,i} + a_{x,i} \cdot \Delta t - b_{x,i} \\ v_{y,i+1} &= v_{y,i} + a_{y,i} \cdot \Delta t - b_{y,i} \\ v_{z,i+1} &= v_{z,i} + a_{z,i} \cdot \Delta t - b_{z,i} \end{aligned} \tag{3}$$

Where $b_{x,y,z}$ is the bias.

The introduced xyz-frame represents the sensors measurements through its own reference frame. By introducing gyroscope measurements along with a magnetometer, the objects orientation can be obtained in form of quaternions. With the rotation matrix given by the quaternions the values in the sensors frame can be transformed into a frame relative to the earth, such as NED. To determine quaternions from the IMU readings it can be fused with an AHRS filter. In MATLAB the toolbox *Sensor Fusion and Tracking Toolbox* provides an easy way of applying an AHRS filter to GPS, accelerometer and magnetometer data [11]. The function provided with the toolbox that is used for an AHRS filter is called *ahrsfilter*, which inputs gyro, accelerometer and magnetometer readings to output angular velocity and orientation in form of quaternions or a rotation matrix [12]. To test the reliability of the orientation estimate produced with the gathered data the AHRS filter is used on a smaller log of IMU data, where the orientation through the log is known. The estimated orientation is viewed in 3D with the function *poseplot*, to be compared with the real orientation. For a more precise error measurement, the quaternions that are obtained from the PX4 are used as a reference to the estimations. Note that the PX4 also uses an AHRS to predict quaternions. The quality of the estimations reflects that of the gathered data from the IMU, where unsynchronized data or bad noise assumptions can lead to bad estimations of attitude.

To estimate the relative velocity of an object on the earth's surface it is important to consider the gravitational acceleration that is present on the downward (in reference to earth) direction of the

object. The raw accelerometer readings will have the gravitational acceleration distributed over its three-axis depending on its orientation relative to the earth's reference frame. Given that ships operate on sea-level it is safe to assume the gravitational acceleration as constant, chosen according to the local gravitational constant g (in this case $g = 9.82m/s^2$). The gyroscopes Euler angles of pitch and roll can be used to roughly eliminate the gravitational acceleration, as they tell the angle of the sensors from its own reference frame to the Earth's. Given according to Figure 2 that roll (θ) describes the rotation around the x-axis and pitch (φ) describes the rotation around the y-axis, the following equations as seen in (4) are obtained.

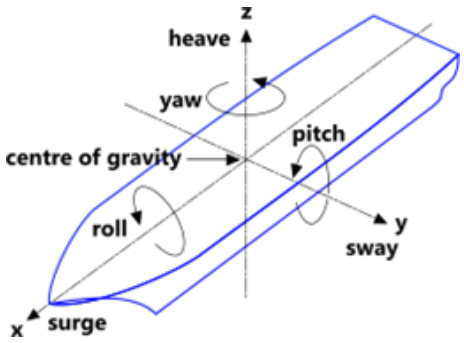

Figure 2: Euler angles in ship reference frame.

$$\begin{aligned} v_{x,i} &= v_{x,i-1} + (a_{x,i-1} - g \cdot \sin(\varphi))\Delta t - b_i \\ v_{y,i} &= v_{y,i-1} + (a_{y,i-1} - g \cdot \sin(\theta))\Delta t - b_i \\ v_{z,i} &= v_{z,i-1} + (a_{z,i-1} - g \cdot \cos(\varphi)\cos(\theta))\Delta t - b_i \end{aligned} \tag{4}$$

However, this is only a direct approach at eliminating the gravitational acceleration from given roll and pitch measurements. The AHRS filter provides a more efficient way of transformation as it includes estimations, through a Kalman filter, of the bias and noise in the IMU measurements along with the Earth's rotation, which affect roll and pitch [13, p. 20]. It is also worth noting that the gravitation is only present in the down direction/axis on the NED frame, as stated in [13].

When the acceleration from the IMU is expressed in NED it can be fused together with the GPS data, which is expressed in the NED frame. To express acceleration in NED through the IMU, the raw acceleration in the xyz-frame can be multiplied by the inverse of the rotation matrix [14] that is obtained through the AHRS filter. This transforms the acceleration to the NED frame and enables fusion with the GPS data.

$$a^{NED} = R^{-1}a^{xyz} = R^T a^{xyz} \tag{5}$$

Equation (5) shows the transformation between coordinate frames using the rotation matrix R. For information on how the rotation matrix is constructed using quaternions, refer to [13, p. 14].

3.3 Sensor fusion of GPS and IMU (Adam)

The resolution of the raw velocity data from the GPS will be improved by fusing it with the velocity data from the IMU. The fusion will theoretically combine each sensors strengths and eliminate their weaknesses. The resolution of acceleration given by an accelerometer will often surpass that of a GPS. However, the velocity data that is integrated from the acceleration will have a significant bias to the true value because of the noise present in the accelerometer readings. One can say that the error from the noise is converted into a bias when integrated, thus drifting the velocity data from the true value over time [15, p. 590]. A positive note is that if the bias is fully removed then the velocity will theoretically have an even higher accuracy as less noise is present after integration. Therefore, the accelerometer can provide a high resolution and accuracy if the noise and bias are being precisely estimated.

The combination of an accelerometer and a gyro in an IMU will further simplify the bias estimation as the correlation of the sensors will help in detecting noise and bias. The magnetometer will also help in the noise and bias estimations, along with providing a way to referencing the velocity inside the NED-frame.

Realistically the noise can't be fully estimated, thus it will always be a bias present over time. Given the time, the bias can drift to any size. The velocity data from the GPS, that is determined through doppler shift [16], have no bias drift present which provides a more absolute value of the velocity. Hence, the GPS will act as a drift-detecting reference over the velocity data from the IMU.

The following figure presents an overview of the sensor fusion with the previously stated strengths and weaknesses of the sensors:

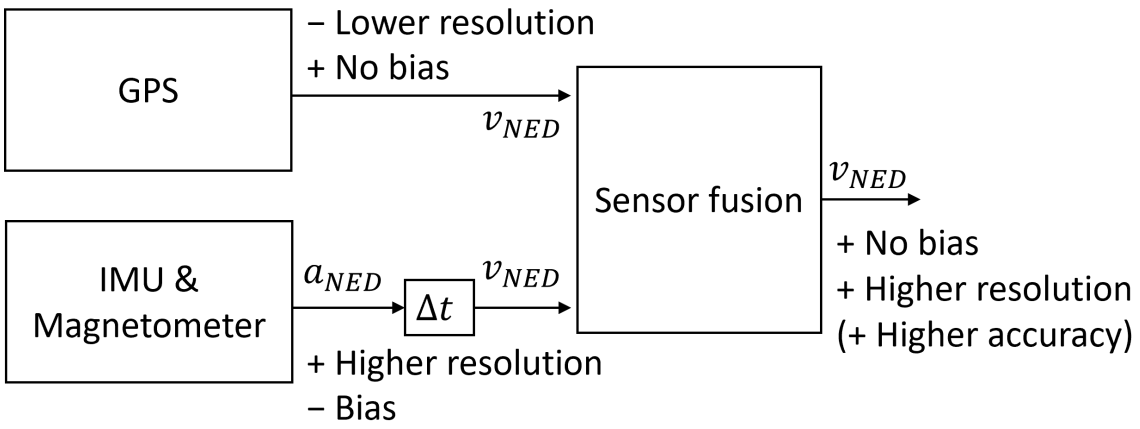

Figure 3: Block diagram of the sensor fusion process.

3.4 Extended Kalman Filter (Adam)

Since we are dealing with a dynamic real system there will be nonlinearities over the states covariances. For such systems, the EKF is a common and optimal choice of filter. The filter will be tested on the acquired data from the IMU and GPS.

The MATLAB toolbox *Sensor Fusion and Tracking toolbox* was used when fusing the IMU and GPS data. The function *insfilterAsync* provides a prebuilt continuous-discrete extended Kalman filter that is optimal for fusion between asynchronous IMU and GPS sensor inputs [17], it also provides an easy way of monitoring the different matrices of the filter.

The function fuses raw GPS, gyro, accelerometer and magnetometer data to estimate pose (position (m), velocity and orientation (quaternions)) in the NED frame (or ENU if specified, East North Down). The state vector of the filter has 28 elements that consists of orientation, angular velocity (rad/s), position (m), velocity (m/s), acceleration (m/s^2), accelerometer bias (m/s^2), gyroscope bias (rad/s), geomagnetic field vector (magnetometer, μT), magnetometer bias (μT). As seen in [17], the filter created by the function *insfilterAsync* has the following state matrix:

$$x = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \\ angVel_X \\ angVel_Y \\ angVel_Z \\ position_N \\ position_E \\ position_D \\ v_N \\ v_E \\ v_D \\ accel_N \\ accel_E \\ accel_D \\ accelbias_X \\ accelbias_Y \\ accelbias_Z \\ gyrobias_X \\ gyrobias_Y \\ gyrobias_Z \\ geomagneticFieldVector_N \\ geomagneticFieldVector_E \\ geomagneticFieldVector_D \\ magbias_X \\ magbias_Y \\ magbias_Z \end{bmatrix} \tag{6}$$

Given that the filter has the process equation $\dot{x} = f(x) + w$, where w is the process noise and $\dot{x}$ is the derivative of x, the transition function $f(x)$ is the following:

$$f(x) = \begin{bmatrix} \frac{-(q_1)(angVel_X)-(q_2)(angVel_Y)-(q_3)(angVel_Z)}{2} \\ \frac{(q_0)(angVel_X)-(q_3)(angVel_Y)+(q_1)(angVel_Z)}{2} \\ \frac{(q_3)(angVel_X)+(q_0)(angVel_Y)-(q_1)(angVel_Z)}{2} \\ \frac{(q_1)(angVel_X)-(q_2)(angVel_Y)+(q_0)(angVel_Z)}{2} \\ 0 \\ 0 \\ 0 \\ v_N \\ v_E \\ v_D \\ accel_N \\ accel_E \\ accel_D \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{7}$$

For more information on the state and transition matrices, refer to [17].

The noise and bias for the different parameters can be specified in the filter, the values used are the corresponding values for the PX4 module's sensors as discussed in Section 3.1. It is vital that as much as possible is known about the different dynamics of the sensors to be able to estimate a parameter more accurately.

For a stable filter, to be able to converge the estimation errors, it needs a good initial state estimation on the different parameters of the state vector [14]. The PX4 module that was used already have an estimated output on quaternions. The estimated quaternions from the PX4 module uses the same approach as the AHRS filter to estimate the quaternions. The raw accelerometer data that is in the sensors own xyz-frame is converted into the NED frame within the function using the provided quaternion values. The first measurements in the data log are used to initiate the filter so that the initial state won't be too far off, otherwise it can't converge as the covariance matrix has to operate on too high or low numbers as it diverges [18].

The GPS-data that is obtained from the PX4-module have a sampling frequency of $5Hz$ and a resolution of $0.001m/s$, with an accuracy of $0.05m/s$ [19]. A less accurate version of the data was constructed, so that it could then be improved again through the fusion process. The original data was rounded such that the new resolution was $0.01m/s$. The sampling frequency was reduced from $5Hz$ to $1Hz$ by iterating over the data and selecting the corresponding samples.

The code for the EKF filter consists of an outer loop that predicts the model state forward by the sample time of the IMU on every iteration. The predictions are based on the accelerometer and gyro readings where the bias and noise estimations are being considered, as well as the variance of the estimated parameter. Inside the loop there are if statements that will fuse in magnetometer and GPS velocity on the iteration corresponding with the sampling frequency of the parameter. In this test the GPS velocity is set at $1Hz$ and the magnetometer to $15Hz$, thus the predictions will be based on only accelerometer and gyro readings until the GPS or magnetometer sample is available. The GPS will act as a correction when fused and is seen as a reference point where the drift of the predictions is removed. This will repeat over the data and the EKF will update the state covariance matrix to perform better estimates over the parameters. Figure 4 shows a simplified overview of the filtering process, where it is visualized using block diagrams.

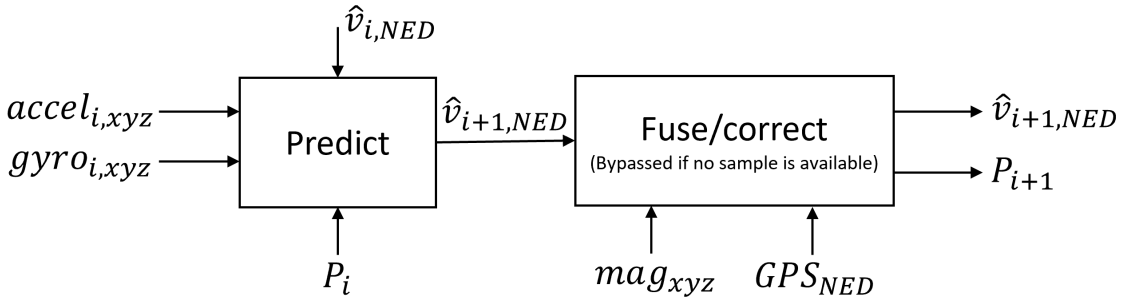

Figure 4: Block diagram of the EKF process.

The filtering process will be manually tuned by adjusting different hyperparameters such as the noise and bias as well as the covariance of the parameters that are fused together. The optimal outcome of the filtering would be that the estimations are accurate and precise enough to be able to give a new and improved value of SOG, such that the resolution of the velocity is increased from 0.01 to 0.001 knots.

4 Results

4.1 Quaternion verification (AHRS) (Erik, Adam)

The data log used in the verification was taken while the module was positioned through different observed poses. The different poses are represented in Figure 5 and are ordered in chronological order going from top left to bottom right.

Figure 5: Photographs depicting the different poses.

The pose given by the quaternions was visualized in MATLAB with the function *poseplot*, which is included in the *Sensor Fusion and Tracking toolbox*. Figure 6 represents the pose of the quaternions given by the PX4 module and are estimations made in the uORB and are not based on the IMU data that was aqcuired for the AHRS filter. Figure 7 shows the pose given by the quaternions that was estimated through the AHRS filter with the acquired data from the IMU.

Note: Figure 6 and 7 are found on the next page. The larger size of the figures is meant to simplify the visual comparison between the poses.

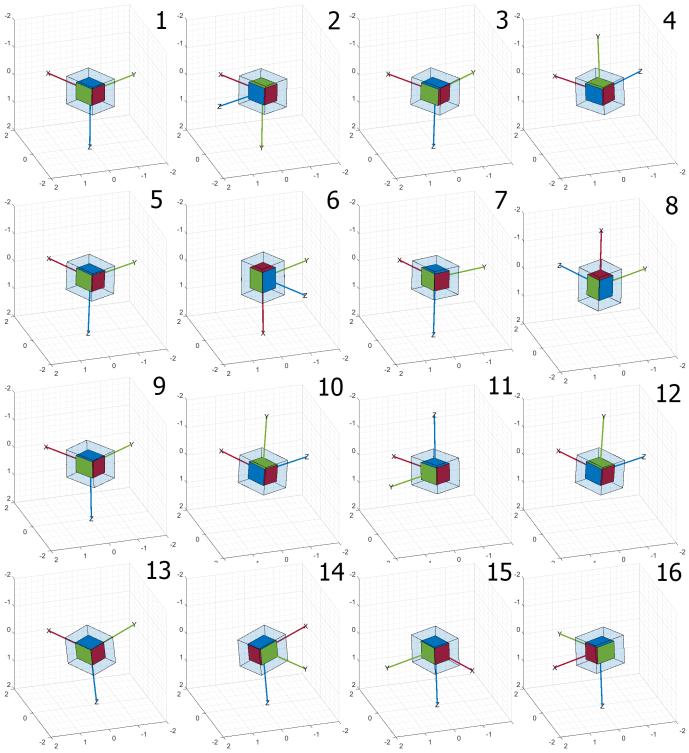

Figure 6: Pose from the PX4 module.

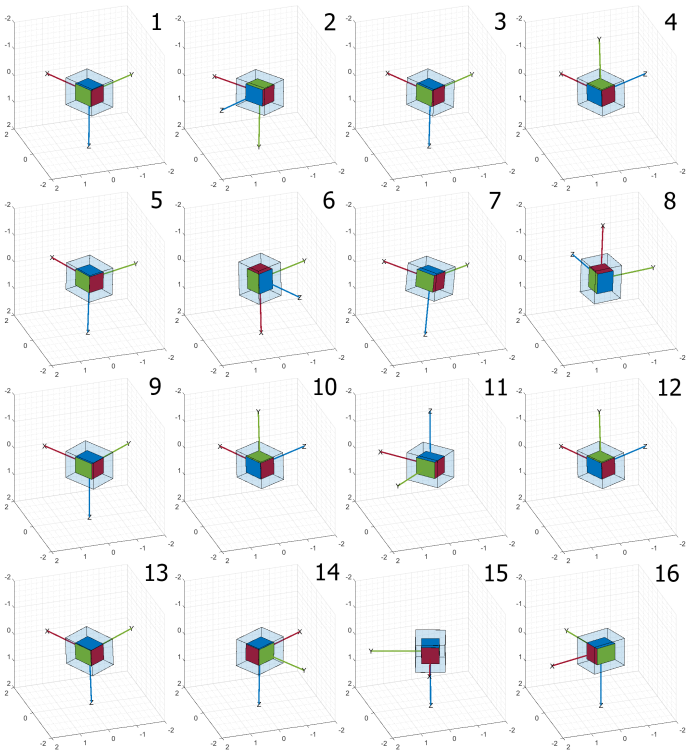

Figure 7: Estimated pose through AHRS with logged data.

4.2 Extended Kalman Filter (Adam)

The following tests shows the EKF velocity estimations along with the low-resolution GPS data that was used in the fusion process and the high-resolution GPS data that is obtained from the PX4 module. As described in Section 3.4, the low-resolution data was constructed from the original high-resolution data. The absolute error between the estimated and the high-resolution GPS velocity is also shown. The bottom axis of the corresponding velocity and error plots display the same values, so that it is easier to determine about a specific time-point in the two plots.

4.2.1 No Delay Compensation

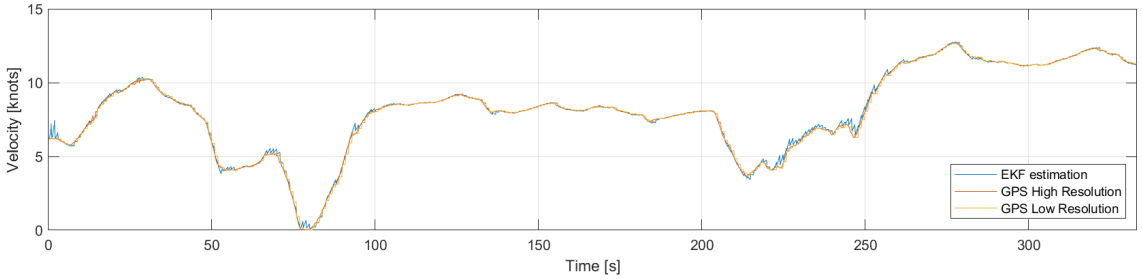

Figure 8: Velocity values over full data set.

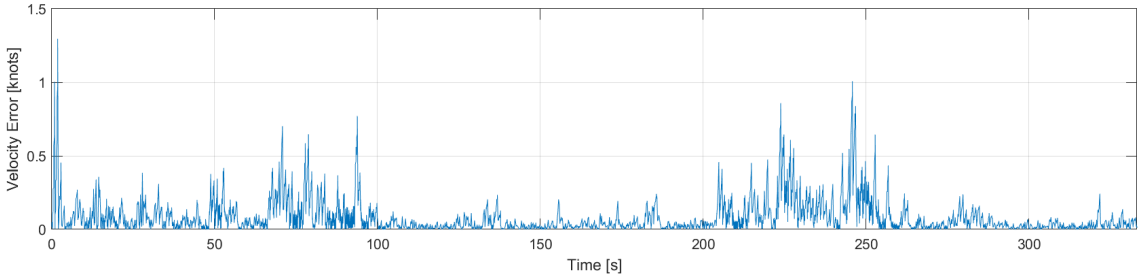

Figure 9: Absolute error of the estimated velocity over full data set.

Figure 8 shows a full-scale overview of the filters result when no delay compensation was made. Figure 9 shows the errors of the estimations in Figure 8. As shown in the figures, the error is higher where there are fast fluctuations in acceleration and lower where the acceleration is more constant. The error RMS value is about 0.1384 knots.

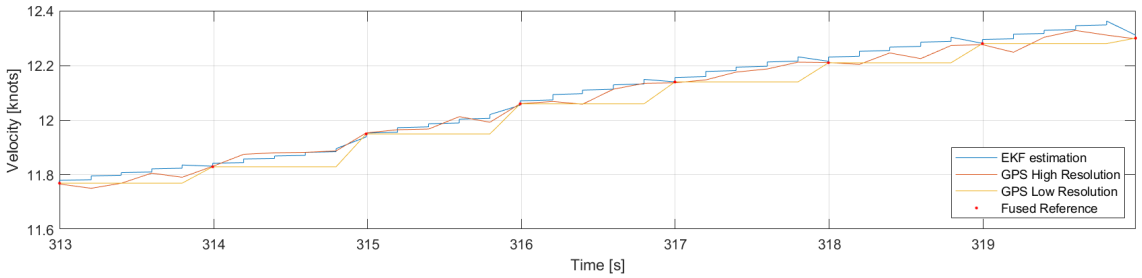

Figure 10: Velocity values at constant acceleration.

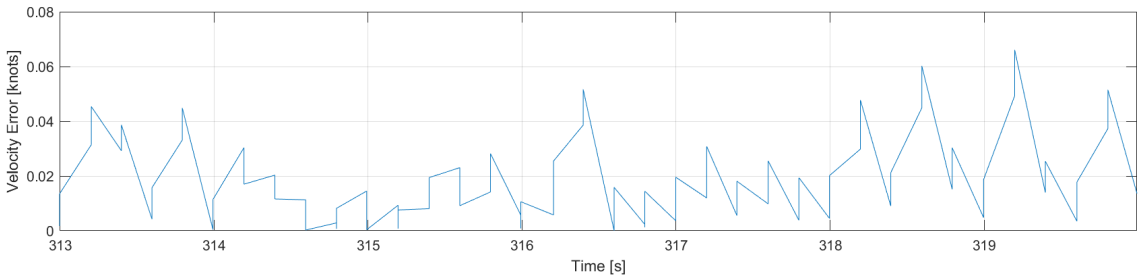

Figure 11: Absolute error of the estimated velocity at constant acceleration.

Figure 10 and 11 provides a closer look at the previous plots, it is a small segment where the acceleration is more constant. In figure 10, the blue line which shows the estimated velocity appears to be improving the accuracy and resolution of the yellow line that shows the low-resolution velocity data from the GPS. Note, the red dots represent the points of reference to where the GPS was fused in the EKF. In Figure 11 we can see that the absolute error of the segment is low, with an error RMS near 0.025 knots.

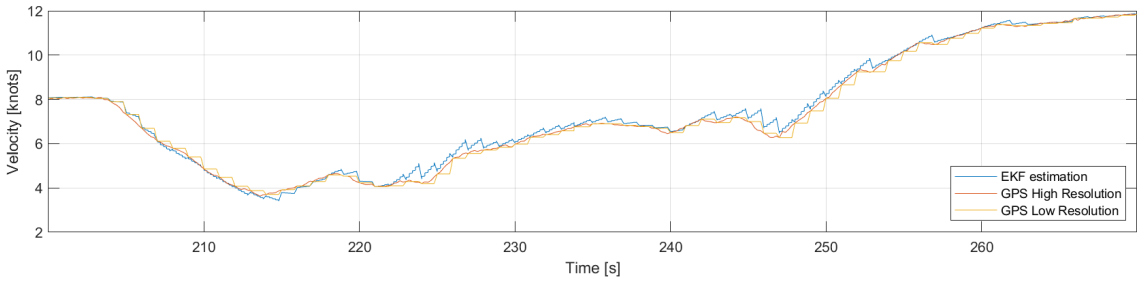

Figure 12: Velocity values at varying acceleration.

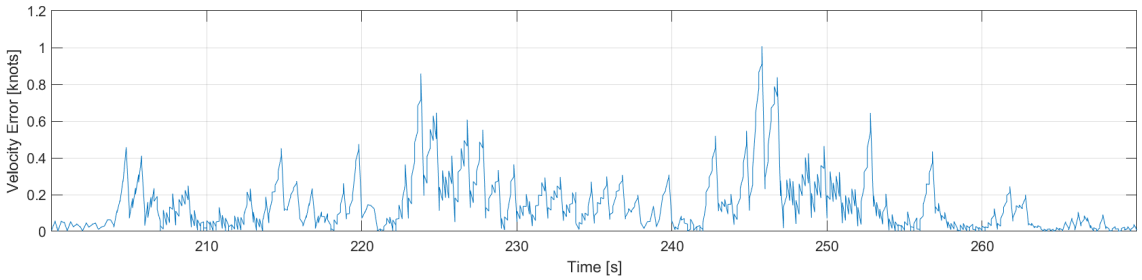

Figure 13: Absolute error of the estimated velocity at varying acceleration.

Figure 12 and 13 shows a segment where there are variations in acceleration. There are several overshoots in the estimations where there is a fluctuation in acceleration. The errors are peaking at these fluctuations and we can see that the error RMS of this section is about 0.2 knots, which is higher than that of the segment with more constant acceleration.

4.2.2 Delay Compensation

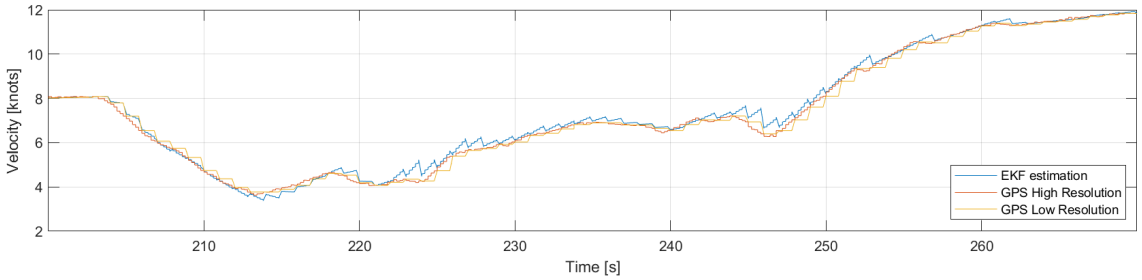

Figure 14: Velocity values at varying acceleration, with GPS delay compensation of 0.333 seconds.

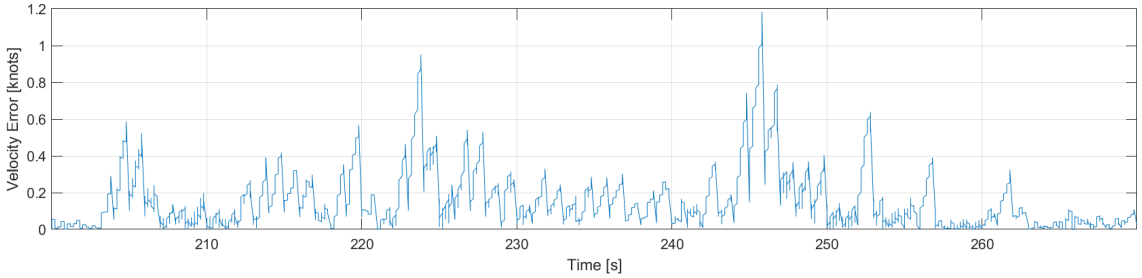

Figure 15: Absolute error of the estimated velocity at varying acceleration, with GPS delay compensation of 0.333 seconds.

Figure 14 and 15 shows the segment with varying acceleration when the GPS data was compensated for 0.333 seconds delay. As seen there was no significant improvement in performance after delay compensation. Tests with different amounts of delay was performed but gave no further improvements.

4.2.3 Larger Dataset

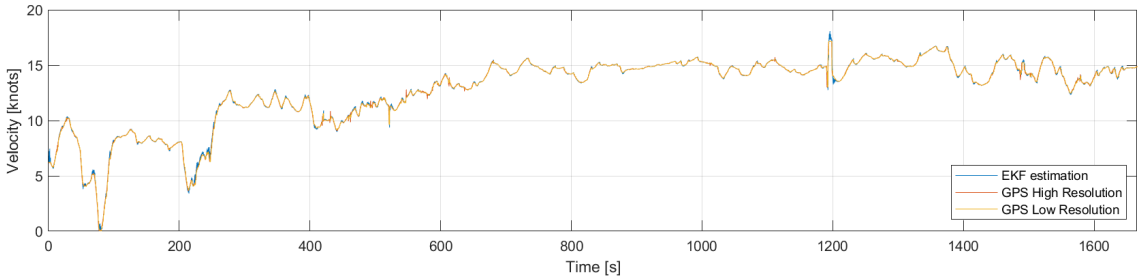

Figure 16: Velocity values over larger data set.

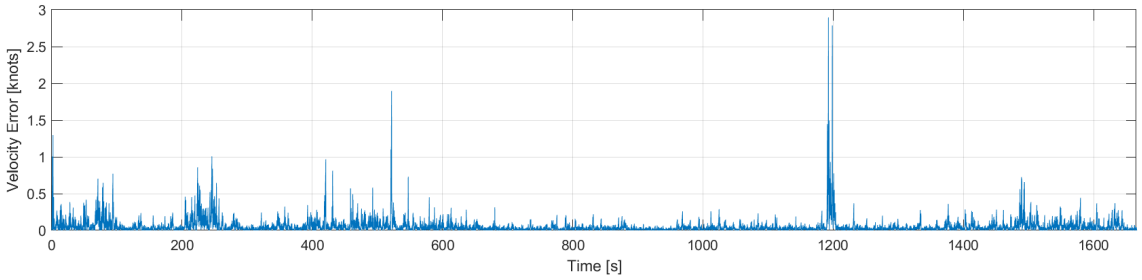

Figure 17: Absolute error of the estimated velocity over larger data set.

The larger data set showed only smaller improvements, but nothing significant. Note that at time 1200 there is a clear outliner present in the data.

5 Discussion

5.1 Quaternion verification (Erik)

The quaternion verification was done to verify that the quaternion estimations were accurate to the real pose of the PX4 system, so that the quality of the IMU data used for the estimations could be assessed. The movements of the logged quaternions and the quaternions estimated by the AHRS filter can be seen in Section 4.1. The results of the logged quaternion and the calculated quaternion from the AHRS filter seems to correspond quite well, however this can be deceiving due to it being still pictures. If one were to simulate the logged quaternions and quaternions from the AHRS filter. It is clear from the visual simulation that the movements of the logged quaternions have less noise and its direction corresponds better with the physical movements than the AHRS filter.

To improve the AHRS filter, it would require more linear relations and better estimation. The reason of why the logged data is better than the calculated data is due to the PX4 already having processed and implemented linear relations and estimations before sending the data through uORB.

Due to the logged quaternion from the PX4 being better than the data from the AHRS filter, the logged data was used in initiating the EKF. The logged quaternions from the PX4 was used in the initiation of the EKF, since it helped the filter to stabilize the estimates and not diverge. The initiation should not have any larger affects on the estimations over a larger data set, since the filter is constantly updating the estimations from the provided raw sensor data to the GPS.

5.2 Sensor fusion with EKF (Adam)

From the results in Section 4.2.1 it is stated that the estimations made with the EKF vary in quality over different conditions. When the velocity is more constant the estimations are accurate, with an effective error that approximately corresponds to the expected accuracy of the reference signal (0.025 m/s = 0.049 knots). Whenever there are faster fluctuations in the velocity, which implies a varying acceleration, the estimations are overshooting the reference. The overshoots have a higher effective error to the reference and implies a lower accuracy over fluctuations in velocity.

By visually examining the overshoots in Figure 12 one aspect of the appearance could be that the estimated velocity fits better with the direction of the previous velocity gradient. The underlying causes are uncertain, where the more feasible explanations are the quality of the data acquisition or bad specifications in the properties of the EKF. An initial thought was that the IMU and GPS data are not synchronized to each other, where the GPS data is delayed. A delay in the GPS data would mean that the predictions made between the corrections would be based on acceleration from the IMU, that is of a more previous moment than at the initially referenced value from the GPS. This would also explain the higher accuracy at the sections where the acceleration is constant, as a delay to the acceleration would not be noticeable if the acceleration remained constant over the iterations.

Delay compensation was implemented by shifting the elements of the GPS dataset, where a shift in one element corresponds to a delay of the sample time (inverse of sampling frequency). Compensation was made for a delay of 1/3 second and 1 second, that with a sampling frequency of 30 Hz corresponds to shifting 10 and 30 elements respectively. However, as stated in Section 4.2.2 and seen in Figure 12 it was not effective in solving the overshoots. More combinations of delay compensation in both GPS, IMU and Magnetometer data was tested, where none gave a significant difference on the outcome.

To prevent any larger misalignment in the data it can be synchronized during the acquisition. This could be done by having a global time frame in Simulink that records the specific moment that every datapoint from each sensor was acquired on. The data points can then be synchronized by interpolating over the time frame. This was not fully implemented due to time constraints. It is worth noting that the function *insfilterAsync* that was used in MATLAB is meant for asynchronous data and is likely the reason to why the delay compensations made in the datasets had no larger impacts on the estimations.

Apart from unaligned data, let us consider the other plausible cause of the overshoots being that the sensor specifications could have been poorly specified in the EKF properties. As previously discussed in Section 5.1 the estimations for the quaternions indicated that the data or specifications of the IMU was not optimal. Since the EKF relies on having a good representation of the system, it is crucial that the noise and bias of the components are known. As stated in Section 3.1 the noise and bias for the sensors were chosen according to the corresponding preprinted datasheets. It is worth noting that any fluctuations in noise because of operational conditions, such as temperature, was not considered as the measurements were taken under rather nominal and constant conditions. It would be of interest to implement corrections to the parameters for such conditions in more extreme environments or if higher precision is required, which could be relevant in the scenario of a ship application. It is important to realize that the characteristics of the noise and bias can vary between different conditions and the relation is often non-linear for the cheaper IMUs. If the noise deviates from its specifications and predictions, then the quality of the estimations will be affected. The extended Kalman filter algorithm can detect and update the covariance between the parameters but if the system is poorly defined then it follows that the predictions will differ from the true state.

The longer data set in Figure 16, Section 4.2.3, did not show any significant increase in performance of the system. There were some small improvements, but it never converged into making optimal estimations during varying acceleration. It is known that the EKF continuously updates its covariance matrix to provide better estimations, but the overshoots still abide after the larger dataset. When repeating the EKF over the same dataset without clearing the properties of the filter, there was still no significant improvement regarding the overshoots.

5.3 Further Improvements (Adam)

The overall performance of the filter is something that must be improved to be relied upon as a new and more accurate velocity reading. The verification of the velocity currently relies on comparing it to accurate GPS data and is something that is not meant to be required upon application. Since any introduced GPS can vary in quality, it is essential that the estimations can be based as much as possible on the IMU so that it would perform with the same quality over different systems. It would be helpful to only require basic information on the introduced GPS, such as the GPS noise from the datasheet. The noise parameter of the GPS helps the EKF to determine the accuracy of the GPS signal and is useful when correcting the estimations to it. It corrects the estimated value to a point weighted between the estimation and the GPS value. The new point of velocity can provide a more accurate value of velocity if the estimations from the IMU are dependable. It is also not certain that the velocity from the GPS is as accurate during the velocity fluctuations. Therefore, the true value where there are overshoots could be at a weighted point between the estimated value and the GPS value.

An initial goal was to apply a smoothing filter on the estimations from the EKF, but was not

implemented due to the time constraints. The addition of a smoothing filter could help with eliminating the overshoots. A simple approach would be to interpolate with a spline, however it can be important to weigh between more parameters than just the velocity data points. There are smoothing filters that weighs in the covariance between the predictions and corrections such that the fusion process could capture more of the potentially higher accuracy of the IMU. A commonly used smoothing filter for EKF is the RTS (Rauch-Tung-Striebel) algorithm, which is a fixed-interval filter that combines for- and backwards filtering with an updating covariance matrix similar to that of the EKF [13]. The implementation of the RTS is further discussed in [13].

Apart from improving the estimations by building on the currently constructed EKF there are also reasons as to go back to the more initial phases such as the data acquisition. Data synchronisation is something that needs to be troubleshooted further. It would also be beneficial to ensure that all of the parameters in the EKF are specified correctly and to further tune were needed. The MATLAB function *tune* from *Navigation toolbox* provides a way of tuning the noise parameters to reduce the estimation errors. Note that the process involves having a reference value to tune the filter against, which the GPS high-resolution signal could be used as.

Further validation techniques will have to be performed to verify that the estimations have achieved a good accuracy and precision, along with further verifying the resolution of the readings in relation to the accuracy.

5.4 Other approaches and implementation (Erik)

Besides the use of an extended Kalman filter, other approaches can be made to improve the velocity estimation. This is by using other methods such as unscented Kalman filter, particle filter and neural network.

Unscented Kalman filter (UKF) is a filter used for nonlinear estimation. It is a variant of the Kalman filter and is also a filter that is deemed to have better performance than the EKF. The differences are that the EKF uses first order approximation and linearizes the covariance and the mean. Where the UKF uses unscented transformation to pick out sample points also known as sigma points. The sigma points are then used to calculate the covariance and the mean. The UKF has lower error and better accuracy compared to the EKF. However, the main advantages of the EKF are that the EKF is easier to implement than the UKF as the Jacobian is very easy to derive analytically. The EKF is more computationally efficient, thus being faster and more energy effective compared to the unscented Kalman filter [20].

A neural network can also be used to improve the estimation of velocity. Neural network is an iterative process, meaning that for the system to improve it must be trained. This is done by feeding the network with data, which it can learn from. From the training, the neural network learns and find the best parameters for the system. The neural network is shown to have lower error and higher accuracy compared to the EKF. However, the EKF is faster to implement and doesn't need the huge amount of training that the neural network needs [21].

A particle filter uses a set of Monte Carlo algorithms to solve filtering problems. Particle filtering can be used for velocity estimation. For more information on particle filters, read *"Particle Filters for Positioning, Navigation and Tracking"* in [22].

5.5 Validation and correlation between parameters (Yousef)

There are different ways to validate different boat parameters and what the different models for each parameter might look like. We will also see the correlation between the parameters.

In addition, the various parameters can help to understand how everything is connected and if something goes wrong, it can affect for example the SOG but not enough to affect the entire result. Parameters that have built the SOG are IMU and GPS sensors, but there are several parameters that can affect the result. An example of other parameters is fuel consumption, STW, speed and wind resistance.

A specific model for estimating the speed through water (STW) signal can be constructed:

$$STW = SOG + K \cdot W \tag{8}$$

We can see that speed over ground (SOG) can be rewritten as:

$$SOG = STW - K \cdot W \tag{9}$$

SOG is the signal that is taken over the ground, STW is the signal for the speed through the water and W is the reactive wind signal which is our data set. Should the wind signal be noisy, it would give the wrong value to SOG, but not enough that we would get an incorrect result.

We can implement an extended kalman filter (EKF) to estimate SOG. The estimation is made either with the help of information about individual signals or with the help of redundancy to correct a signal with other signals that are more correct. It can remove noise and incorrect values. In this case, we estimate SOG with a GPS signal (PX4) [23].

To understand the relationship between the dependent and independent parameters, we can look at the correlation. The correlation between SOG and other parameters is not so high because the ship's speed is primarily determined by the torque and speed of the ship. In addition, other weather-related properties have a relatively low correlation. We could see this in the equation (9), where W is the wind signal and if we have received noise, it would not affect the SOG [24].

Let's check the model for fuel consumption and how it can affect the SOG. Ships fuel consumption may depend on certain technical factors such as propeller condition and engine efficiency. Fuel consumption may also depend on environmental conditions that affect ships parameters, such as sea currents, waves, and wind. In addition, the acceleration of the ship has also impacted fuel consumption. The model for fuel consumption can depend on various parameters:

Parameter	**Unit**
Main engine RPM (RPM)	Revolution/min
Speed Over Ground (SOG)	Knot
Speed Through Water (STW)	Knot
Relative Wind Direction (RWD)	°
Rudder angle (RUD)	°
The Displacement (DIS)	Ton
Wetted surface area (WSA)	m^2

Table 1: Available parameters from the ship control system

With the correlation between the fuel consumption and some of these parameters, we can get the following regression model [25]:

$$f_c = a_0 + a_1 \cdot speed^2 + a_2 \cdot wind \tag{10}$$

It is a good method to combine many parameters when it comes to correlation and regression for fuel consumption because if any parameter showed errors or if we were able to remove a parameter, it would not affect the fuel consumption much, it is positive for SOG. If we look further at more correlation for fuel consumption, we can see it is exponential with ship's speed, so fuel consumption depends a lot on speed. It is the biggest factor that can affect fuel consumption, if we increase the speed then the fuel consumption would also increase.

Finally, we can say the more parameters we used, the more reliable and more precise results we obtain because small disturbances will not affect the result [26].

6 Conclusion (Adam)

The constructed EKF can estimate the velocity between GPS readings where the covariance of the measurements weighs the velocity value upon correction such that the IMU and GPS precision and accuracy can be fused. The performance of the EKF was lacking during variations in acceleration, where it would overshoot from the true value of velocity. This was determined to likely be because of non-homogenous relations in the noise of IMU or bad data acquisition. The result from this report currently acts as a foundation on implementing the EKF to the available ships. For more accurate estimations it is determined that further implementations are required for the EKF, such as applying the RTS smoothing filter over the prediction part of the EKF.

There are other filters that can be implemented such as the UKF, particle filter or a neural network. However, the EKF is by far the most computational efficient of the filters. Since the goal is to improve the resolution of velocity to be used in optimizing the energy consumption of a ship, it is vital that the energy required for the computations does not exceed that of the benefited route optimization. There are many applications of sensor fusion that can be further expanded to a multi-sensor network, that would be able to use several of the ship-parameters to give robust estimations on various parameters of the ship. It is hard to conclude how much that can be gained, but it is worth noting that every small change in optimization would have larger effects in the long run.

If expanded and implemented, the addition of an extended Kalman filter would provide Qtagg with a lucrative offer to their costumers. Where they can obtain higher accuracy velocity data without having to pay for a more expensive high-grade GPS sensor. The opportunity to be able to use a consumer-level IMU and GPS to its full potential, where it would rival a standalone high grade GPS sensor, is an offer that is hard to turn down.

7 MATLAB Code

Simplified version of the EKF fusion loop used in MATLAB

```
%% EKF

p = zeros(n,3);
q = zeros(n,1,'quaternion');
v = zeros(n,3);

    j = 0;
    in = 1;

for i = 1:n
    predict(filt,1/fs);

    fuseaccel(filt,accel(i+j,:),Racc);
    fusegyro(filt,gyro(i+j,:),Rgyro);


    if ~mod(i,fix(fs/2))
        fusemag(filt,mag(i+j,:),Rmag);
    end

    if ~mod(i,fs)

        fusegps(filt,lla(i+j,:),Rpos,gpsvel_lowres(i+j,:),Rvel);
        idx(in) = i+j;
        in = in+1;
        end

        [p(i+j,:),q(i+j),v(i+j,:)] = pose(filt);
    end
end
```

References

[1] G. R. G. a. UNAVCO, *Decoding the gps signal*, Dec. 2019. [Online]. Available: https://spotlight.unavco.org/how-gps-works/gps-basics/decoding-the-gps-signal.html (visited on 05/13/2022).

[2] D. G. King-Hele, "Review of tracking methods," *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, vol. 262, no. 1124, pp. 5–13, 1967, ISSN: 00804614. [Online]. Available: http://www.jstor.org/stable/73457 (visited on 05/13/2022).

[3] B. Pattanayak and L. Moharana, "Analyzing the effect of dilution of precision on the performance of gps system," in *2021 1st Odisha International Conference on Electrical Power Engineering, Communication and Computing Technology(ODICON)*, 2021, pp. 1–5. DOI: 10.1109/ODICON50556.2021.9428982. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9428982 (visited on 05/10/2022).

[4] Mathworks, *Extended kalman filters*. [Online]. Available: https://se.mathworks.com/help/driving/ug/extended-kalman-filters.html (visited on 05/06/2022).

[5] Altium, *Unmanned autonomous vehicles: Pros and cons of multiple sensor fusion*, May 2017. [Online]. Available: https://resources.altium.com/p/unmanned-autonomous-vehicles-multiple-sensor-fusion-pros-cons (visited on 05/12/2022).

[6] RF Wireless World, *Advantages of data fusion - disadvantages of data fusion*. [Online]. Available: https://www.rfwireless-world.com/Terminology/Advantages-and-Disadvantages-of-Data-Fusion.html (visited on 05/12/2022).

[7] *Bmi055 - small, versatile 6dof sensor module*, Datasheet, Bosch Sensortec, Nov. 2021. [Online]. Available: https://www.bosch-sensortec.com/products/motion-sensors/imus/bmi055/ (visited on 05/11/2022).

[8] *Ist8310 - 3d magnetometer*, Datasheet, Isentek, Motion Sensor Provider. [Online]. Available: https://www.isentek.com/userfiles/files/IST8310%20Datasheet%20v1_4%20Brief.pdf (visited on 05/11/2022).

[9] PX4 Autopilot, *#uorb messaging*, Feb. 2021. [Online]. Available: https://docs.px4.io/v1.12/en/middleware/uorb.html (visited on 05/12/2022).

[10] C. Jonsson, "Velocity estimation in land vehicle applications - sensor fusion using gps, imu and output-shaft," Royal Institute of Technology, Dissertation, 2016. [Online]. Available: http://kth.diva-portal.org/smash/get/diva2:939065/FULLTEXT01 (visited on 05/06/2022).

[11] Mathworks, *Sensor fusion and tracking toolbox*. [Online]. Available: https://se.mathworks.com/help/fusion/index.html?s_tid=CRUX_lftnav (visited on 05/06/2022).

[12] ——, *Ahrsfilter*. [Online]. Available: https://se.mathworks.com/help/fusion/ref/ahrsfilter-system-object.html (visited on 05/07/2022).

[13] E. Shin, "Estimation techniques for low-cost inertial navigation," University of Calgary, Dissertation, 2005. [Online]. Available: https://www.ucalgary.ca/engo_webdocs/NES/05.20219.EHShin.pdf (visited on 05/14/2022).

[14] P. SJÖBERG, "Design and implementation of an exogenous kalman filter for uavs," Royal Institute of Technology, Dissertation, 2018. [Online]. Available: https://kth.diva-portal.org/smash/get/diva2:1232900/FULLTEXT01.pdf (visited on 05/14/2022).

[15] A. S. Morris and R. Langari, *Measurement and Instrumentation*, 3rd ed. London Wall, London: Katey Birtcher, 2021.

[16] VBOX, *Gps accuracy*. [Online]. Available: https://www.vboxautomotive.co.uk/index.php/en/how-does-it-work-gps-accuracy (visited on 05/14/2022).

[17] Mathworks, *Insfilterasync*. [Online]. Available: https://se.mathworks.com/help/fusion/ref/insfilterasync.html (visited on 05/07/2022).

[18] S. Zhao and B. Huang, "On initialization of the kalman filter," in *2017 6th International Symposium on Advanced Control of Industrial Processes (AdCONIP)*, 2017, pp. 565–570. DOI: 10.1109/ADCONIP.2017.7983842. [Online]. Available: https://folk.ntnu.no/skoge/prost/proceedings/adconip-2017/media/files/0118.pdf (visited on 05/13/2022).

[19] PX4 Autopilot, *Pixhawk 4*, Jun. 2021. [Online]. Available: https://docs.px4.io/v1.12/en/flight_controller/pixhawk4.html (visited on 05/19/2022).

[20] E. Wan and R. Van Der Merwe, "The unscented kalman filter for nonlinear estimation," in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, 2000, pp. 153–158. DOI: 10.1109/ASSPCC.2000.882463. (visited on 05/17/2022).

[21] A. AbdulMajuid, O. Mohamady, M. Draz, and G. El-bayoumi, *Gps-denied navigation using low-cost inertial sensors and recurrent neural networks*, arXiv, 2021. DOI: 10.48550/ARXIV.2109.04861. [Online]. Available: https://arxiv.org/abs/2109.04861 (visited on 05/10/2022).

[22] P. Djuric, J. Kotecha, J. Zhang, *et al.*, "Particle filtering," *IEEE Signal Processing Magazine*, vol. 20, no. 5, pp. 19–38, 2003. DOI: 10.1109/MSP.2003.1236770. (visited on 05/18/2022).

[23] A. El Ouardi, *Applying autonomous methods for signal analysis and correction with applications in the ship industry*, Mälardalen University, 2018. [Online]. Available: http://www.diva-portal.org/smash/get/diva2:1214553/FULLTEXT01.pdf (visited on 05/11/2022).

[24] M. Abebe, Y. Shin, Y. Noh, S. Lee, and I. Lee, "Machine learning approaches for ship speed prediction towards energy efficient shipping," *Applied Sciences*, vol. 10, no. 7, p. 2325, 2020. [Online]. Available: https://www.mdpi.com/2076-3417/10/7/2325/htm (visited on 05/18/2022).

[25] N. Bialystocki and D. Konovessis, "On the estimation of ship's fuel consumption and speed curve: A statistical approach," *Journal of Ocean Engineering and Science*, vol. 1, no. 2, pp. 157–166, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2468013315300127 (visited on 05/18/2022).

[26] L. Jimenez Blazquez, *Mathematical methods for maritime signal curation in noisy environments*, Mälardalens University, 2019. [Online]. Available: http://mdh.diva-portal.org/smash/get/diva2:1321388/FULLTEXT01.pdf (visited on 05/12/2022).