Net-relative localization algorithm for fish cage inspection operation

Leonard O. Cheri

Masteroppgave ved Institutt for teknologisystemer
(30 studiepoeng)
UNIVERSITETET I OSLO

9. januar 2018

Net-relative localization algorithm for fish cage inspection operation

Leonard O. Cheri,

9. januar 2018

Summary

The aim of this project was to investigate the abilities to track the traveled path of a ROV relative to a net pen, using a DVL system, a single axis gyroscope, a GPS and a pressure sensor. The project was performed by simulating the path of a ROV in the context of a net inspection routine in the aquaculture industry. The ROV's position was estimated by using different variants of the Kalman Filter. The filter design process went through different steps, implementing different filters and sensor fusion levels. After each step the results were assessed and compared in order to identify the possible challenges and improvements. Four different filters were implemented. The expected performance improvement at each level of sensor fusion was met. The results of the simulation illustrates that the proposed smoother unscented Kalman Filter manages to track the ROV's trajectory and gives the best position estimate. This implies that the sensor configuration used in the project is applicable for aquaculture net inspections.

Foreword

This assignment was written as a master thesis in the master's program Cybernetics at the Department of Technology Systems (ITS) at the University of Oslo (UiO). The project has been executed in collaboration with SINTEF and ITS.

I would like to thank my supervisor Oddvar Hallingstad at ITS, for guidance and two very inspiring years as your student. Further, I would like to thank Magnus Bjerkeng at SINTEF, for his guidance and for giving me the opportunity to be a part of one of SINTEF's current projects. Lastly, I would like to thank my wife Sofie, for her support and patience.

Table of content

Figures

1.Introduction

1.1 The assignment

Guidance, Navigation and motion Control (GNC) of underwater vehicles for aquaculture operations are in many aspects particularly complex due to the challenges the navigated environment poses. Gravity net cages used by modern aquaculture are exposed to many risks, holes in the net and other types of net failure present a challenge with respect to fish escapes. One of the effective measures established to deal with this issue in Norway, is a mandatory net inspection after each net related operation. Earlier, these inspections were mostly performed by divers doing manual inspection. However, Remotely Operated Vehicles (ROVs) provide a safer, more robust and cost efficient solution.

The purpose of this thesis is to investigate the abilities of a system consisting of a Doppler Velocity Log (DVL), a Global Positioning System (GPS), a pressure sensor and a gyroscope, to localize a ROV relative to a net during the following motions: starting on-the-surface, diving down to the bottom, going up to the surface. The ROVs used for net inspections are usually equipped with a camera facing the net, used to navigate and detect holes. The proposed algorithm in the thesis is tested in a simulated environment with different sensor data configurations.

In the following, to further elaborate the purpose of this assignment and the importance of optimizing net inspections, a brief introduction to the fish escapes issue is given. Then, a short review of some of the previous and state of the art solutions to underwater navigation and localization is given, before presenting the work of the assignment.

1.2 Fish escapes

The Norwegian marine aquaculture industry started in the early 1970's and has since then been a successful industry with continuous growth. As the oil prices have been dropping in the later years, the aquaculture industry is also becoming more and more interesting to investors. Today, Norway is considered one of the leaders in the culture of salmon in sea-cages[1].

As the industry is growing and becoming more and more appealing financially, the interest to optimize the industry and reduce losses is getting bigger. One of the biggest issues in modern Norwegian aquaculture is fish escapes. In fact, Norway has the most comprehensive national record of fish escapes in the world[2].

Fish escapes can be caused by a variety of issues related to farming equipment. Figure (1) illustrates a report by fish farming companies to the Norwegian Directorate of Fisheries during the period September 2006 and December 2009. As shown in the figure, the Atlantic salmon escapes are almost 70% due to structural failures of equipment. Structural failures may occur during severe environmental conditions, from strong winds to waves and water currents, especially in combination with equipment fatigue or human installation error[3].

[1] Jensen, Ø., Dempster, T., Thorstad, E., Uglem, I., Fredheim, A. Escapes of fishes from Norwegian sea-cage aquaculture, consequences and prevention. (2010).
[2] Jensen, Ø., Dempster, T., Thorstad, E., Uglem, I., Fredheim, A. Escapes of fishes from Norwegian sea-cage aquaculture, consequences and prevention. (2010).
[3] Jensen, Ø., Dempster, T., Thorstad, E., Uglem, I., Fredheim, A. Escapes of fishes from Norwegian sea-cage aquaculture, consequences and prevention. (2010).

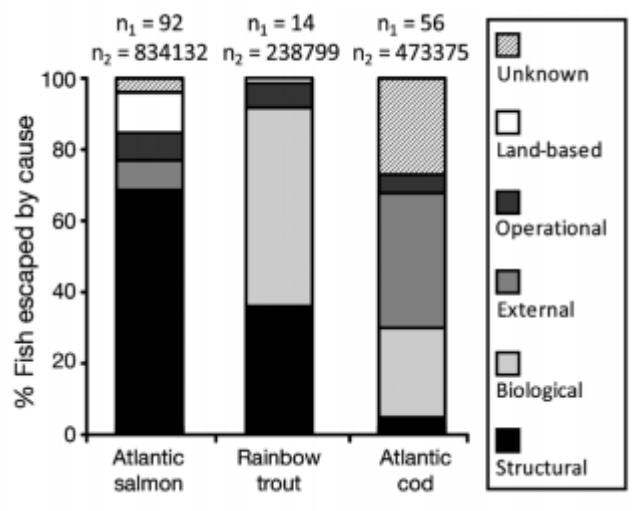
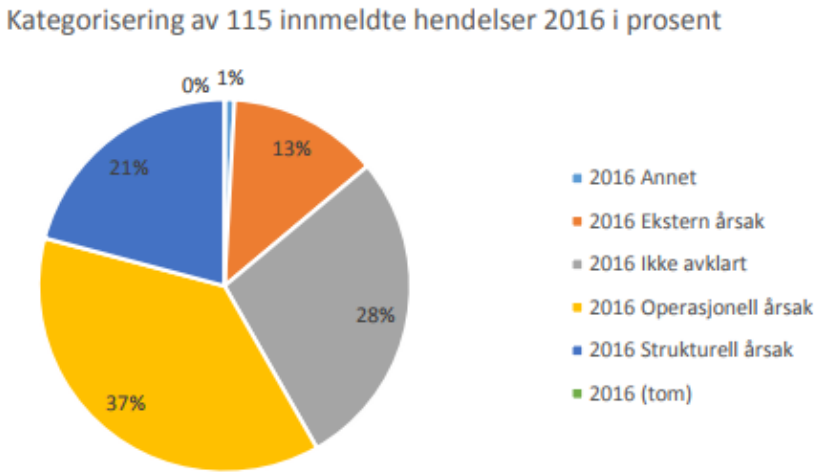


Figure 1: Causes of escape from reports by fish farming companies to the Norwegian Fisheries Directorate from 1 September 2006 to 31 December 2009. n1: total number of reported escape incidents upon which the % of fish escaped by cause is based. n2: total number of escaped fish reported from 1 September 2006 to 31 December 2009 upon which the % of fish escaped by cause is based[4].

Large-scale structural-related escapes represent a small percentage of the events reported, but when looking at the number of escaped fish, these incidents seem to be the main cause for most of the loss. In 2009, only 19% of the incidents reported were structural related, but 91% of escaped fish were caused by this type of incidents[5]. Fish escapes are not only an economical problem; it can also have an environmental impact on the population of wild salmon.

Thus, focus on preventing this type of not frequent structural incidents, will have a big impact on the consequences of fish escapes. Figure (2) illustrates incidents reports to the Norwegian Directorate of Fisheries in 2016. The structural incidents are still the cause of 52% of the total number of fish escaped.

[4] Figure is taken from: Jensen, Ø., Dempster, T., Thorstad, E., Uglem, I., Fredheim, A. Escapes of fishes from Norwegian sea-cage aquaculture, consequences and prevention. (2010).
[5] Jensen, Ø., Dempster, T., Thorstad, E., Uglem, I., Fredheim, A. Escapes of fishes from Norwegian sea-cage aquaculture, consequences and prevention, page 74. (2010),



Numbers of fish for every category:

- External: 13 705
- Not identified: 18 435
- Operational: 58 911
- Structural: 99 738
- Others: 20

Figure 2: Categories of fish escapes causes and Numbers of fish for each category from 115 reports by fish farming companies to the Norwegian Fisheries Directorate in 2016[6].

1.3 Underwater localization

Autonomous underwater vehicle (AUV) navigation and localization in underwater is a challenging and complex task, due mainly to the unstructured and unpredictable conditions underwater environment impels, from currents and waves to flexible structures changing in an undetermined pattern. It's not possible to use GPS, and underwater communications are low bandwidth and unreliable. In the past, solving localization problems was done by using expensive inertial sensors, periodic surfacing to be able to use GPS and improve radio frequency signals or even installing beacons in the studied area. The progress made in underwater communications and the use of Simultaneous Localization and Mapping (SLAM) in underwater offers new possibilities in the field[7].

Accurate localization is essential to be able to use the gathered data. Above water, most solutions use radio or spread-spectrum communications and GPS. Underwater, acoustic-based sensors and communications perform better while still suffering from certain issues as:

[6] Fiskeridirektoratet. Kategorisering av rømmingshendelser i 2016. (2017)
[7] Paull, L., Saeedi, S., Seto, M., Li, H. AUV navigation and localization: a review. (2014)

• Small bandwidth.

• Low data rate.

• High latency due to sound speed in water.

• Variable sound speed due to fluctuating water temperature and salinity.

• Unreliability.[8]

There are three common categories of underwater navigation and localization systems[9]:

1.3.1 Inertial

Dead reckoning (DR) uses knowledge about the last position and orientation and velocity or acceleration vectors to estimate the next position. The biggest disadvantage of DR is that errors are cumulative due to integration. Despite that, DR is still widely used in modern navigation systems.

The performance of an Inertial Navigation System (INS) is dependent on the quality of the inertial measurement units used. Inertial sensors are the basis of a navigation scheme, and are usually combined with other techniques and sensors to overcome each other's disadvantages. In certain situations using inertial sensors is the only option, like in extreme depths for example.

1.3.2 Acoustic

In acoustic navigation systems, localization is performed by measuring the Time of Flight (TOF) of acoustic signals. There are several methods depending on the geometry of the setup:

Ultra-short/Short Baseline (USBL)

The transducers on the transceiver are spaced with the approximated baseline on the order of less than 10cm. USBL navigation allows for relative localization to the surface ship as

[8] Paull, L., Saeedi, S., Seto, M., Li, H. AUV navigation and localization: a review. (2014)
[9] The information on the following methods is taken from Paull, L., Saeedi, S., Seto, M., Li, H. AUV navigation and localization: a review. (2014)

illustrated in figure (3). Relative range and bearing are calculated from TOF and phase differencing across the array of transceivers. The main limitation is the range.

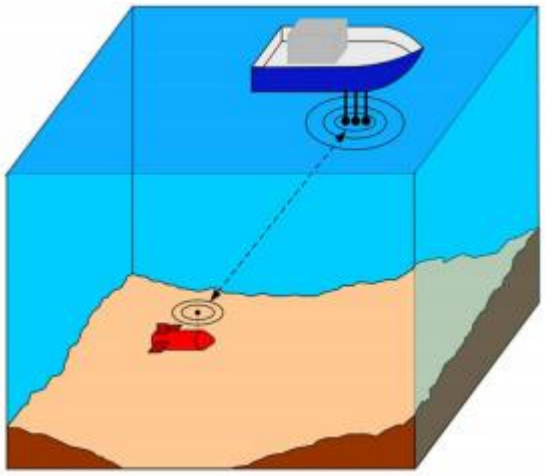


Figure 3: USBL system illustration[10].

Short Baseline (SBL)

The beacons are placed at opposite ends of a ship's hull as shown in figure (4). Triangulation of acoustic signals is used for localization. The position accuracy is highly dependent on the size of the baseline (i.e. the size of the ship).

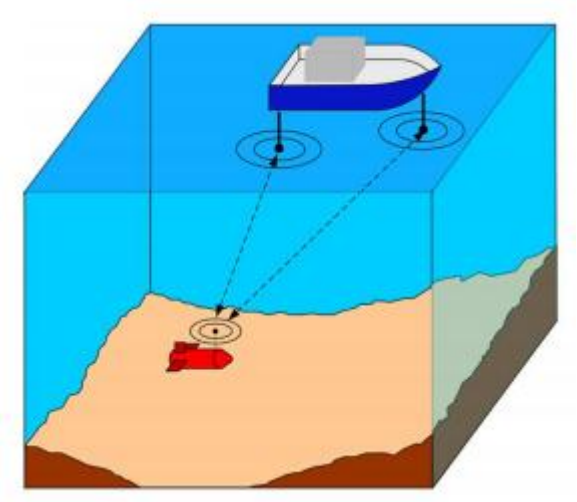


Figure 4: SBL system illustration[11].

[10] Paull, L., Saeedi, S., Seto, M., Li, H. AUV navigation and localization: a review, page 139. (2014)
[11] Paull, L., Saeedi, S., Seto, M., Li, H. AUV navigation and localization: a review, page 139. (2014)

Long Baseline (LBL) and GPS Intelligent Buoys (GIBs)

The beacons are widely placed over the region of interest. A basic system configuration is illustrated in figure (5).Triangulation of acoustic signals is used. In most scenarios the beacons are globally referenced. GIBs remove the need of installing the beacons on the seafloor. In the case of GIBs, the beacons are at the surface, whereas for LBL they are installed on the seabed. The main drawback of LBL is the finite range imposed by the range of beacons and reliance on local knowledge about the environment such as temperature, salinity, conductivity. Nevertheless, LBL is one of the most reliable and accurate localization techniques.

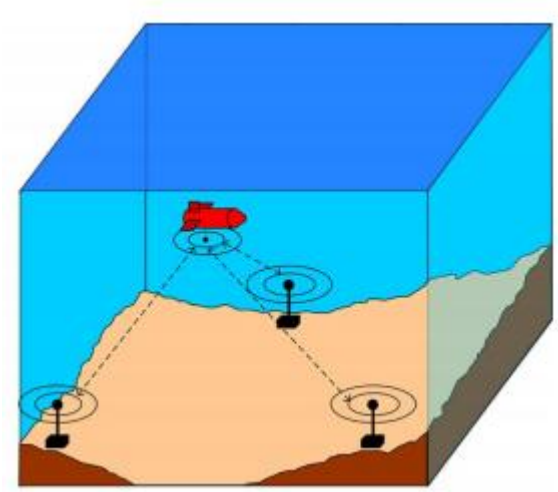


Figure 5: LBL system illustration[12].

A single beacon

In a single beacon setup only one fixed beacon is used. The baseline is simulated by propagating the ranges from the beacon forward in time until the next update is received. This technique has been referred to as virtual LBL. The main issue here is that the AUV movements toward or away from the beacon will cause unbounded growth in the position error.

[12] Paull, L., Saeedi, S., Seto, M., Li, H. AUV navigation and localization: a review, page 139. (2014)

Acoustic modem

Progress in the field of acoustic communications has had a major effect on underwater navigation capabilities. The acoustic modem allows simultaneous communication of small packets and ranging based on TOF. Acoustic modems development offers two important benefits over the discussed methods above. It removes the need for beacons to be fixed or localized prior to the mission and also allows the beacons to move. This can save time and money and also extend the range. Several methods exploit these two advantages. The concept of Moving Long Base Line (MLBL) is a good example; two manned surface vehicles are used to localize the target. An example is given in the paper "Experimental validation of the moving long base-line navigation concept"[13]. This concept has further extended to a single moving source. As will be further elaborated below, a single moving source is the method investigated in this thesis.

1.3.3 Geophysical

Geophysical Navigation uses external environmental features for localization. SLAM is used in almost all methods in this category. Some of the most used methods are:

Magnetic:

The idea is to use magnetic field maps for localization.

Optical:

Visual odometry is the process of estimating the robot's pose by analyzing subsequent camera images. The method is based on using monocular or stereo camera to get images of the seabed and then match these to navigate. Optical systems in underwater can suffer from certain issues such as the range of camera, lighting and susceptibility to scattering.

Sonar:

This method is based on acoustic detection then identification and classification of features in the environment that could be used as navigation landmarks and references.

[13] Vaganay, J., Leonard, J., Curcio, J., Willcox, J. Experimental validation of the moving long base-line navigation concept. (2004)

1.4 Nomenclature

AUV Autonomous Underwater Vehicle

DR Dead Reckoning

DVL Doppler Velocity Log

EKF Extended Kalman Filter

GPS Global Positioning System

GIBS GPS Intelligent Buoys

INS Inertial Navigation System

KF Kalman Filter

ROV Remotely Operated Vehicle

TOF Time of Flight

SBL Short base Line

SLAM Simultaneous Localization and Mapping

SSBL Super Short Base Line

UKF Unscented Kalman Filter

USBL Ultra Short Base Line

UT Unscented Transformation

1.5 Notations

Symbol	Explanation
X_k	The state variable vector at time k
Z_k	The measurement vector at time k
U_k	The control input at time k
w_k	The process model white noise
v_k	The measurement model white noise
$\sim(0, R)$	Multivariate normal distribution with mean 0 and covariance R
X_0	The initial state vector
$\hat{X}_k$	The estimate of X at time k
$\bar{X}_k$	The predicted estimate of X at time k
$\hat{P}_k$	The estimated covariance of X at time k
$\bar{P}_k$	The predicted covariance of X at time k
Q	The measurement noise covariance matrix
R	The process noise covariance matrix
E(X)	The expectation of X
Cov(X)	The covariance of X
δ_{kl}	Kronecker δ_{kl} = 1 for k = l and δ_{kl}=0 for k ≠ l
X^i	Sigma point
K	Kalman gain
N_x	The size of variable x
P_{zz}	The innovation matrix
X_a	The augmented state vector
X^T	The transpose of X

Symbol	Explanation
M^k	Vector M represented in frame (k)
R_1^0	Transformation matrix from frame (0) to frame (1)
SO(n)	The special orthogonal group of order n
P_{xz}	The cross covariance
Det(R)	The determinant of matrix R
$V_{A/B}$	The relative velocity of A with respect to B
V_n^m	The velocity of frame(n) represented in frame (m)
S	Skew matrix

2 The theoretical background

2.1 The Kalman filter

The Kalman filter offers a recursive solution to the linear optimal filtering problem. Consider a discrete-time linear dynamic system with state space representation[14]:

$$X_{k+1} = A * X_k + B * U_k + w_k \quad (1)$$

$$Z_{k+1} = H * X_{k+1} + v_{k+1} \quad (2)$$

X represents the state vector. U is the input at time k, and Z the output measurement vector. The matrix A describes the dynamics of the system, how the system state's changes over time. The matrix B describes how the input is coupled to the system. The matrix H describes a mapping from the system state to the measured or observed outputs.

To account for errors in the system model, $w_k \sim(0,R)$ and $v_k \sim(0,Q)$ are respectively the process and measurement noise. They are both assumed to be additive, white and Gaussian with zero mean and with covariance R and Q respectively. R and Q are both symmetric and positive definite matrices. w_k and v_k are also assumed uncorrelated with each other and with the initial state vector X_0 (Eq. 3).

$$E(X_0) = \bar{X}_0; \quad E(w_k) = 0; \quad E(v_k) = 0 \quad (3)$$

$$E(X_0 w_k^T) = \bar{X}_0; \quad E(X_0 v_k^T) = 0; \quad E(v_k w_k^T) = 0$$

$$cov(X_0) = P_0; \quad cov(w_k w_k^T) = \delta_{kl} * R; \quad cov(v_k v_k^T) = \delta_{kl} * Q$$

Given a system model (A, B and H), the input U and measurements Z, we are able to find a minimum variance estimate of the state of the system.

The Kalman filter (KF) is an optimal estimator when the process and measurement noise are zero mean Gaussian noise.

[14] The Kalman filter content is taken from Corke, Peter. Robotics vision and control. Fundamental algorithms in Matlab. Kalman Filter Appendix H. (2013)

The filter has two steps, first the prediction step based on the previous step and input.

$$\bar{X}_{k+1} = \mathrm{A} * \hat{X}_k \ + \ \mathrm{B} * U_{\mathrm{k}} \tag{4}$$

$$\bar{P}_{k+1} = \mathrm{A} * \hat{P}_k * A^T \ + \ \mathrm{R} \tag{5}$$

$\hat{X}$ is the estimate of the state and $\hat{P}$ is the estimate of the covariance of $\hat{X}$.

The second step is the update step.

$$\mathrm{K} = \ \bar{P}_{k+1} \ * \ H^T \ * \ (\mathrm{H} \ * \ \bar{P}_{k+1} \ * \ H^T \ + \ \mathrm{Q})^{-1} \tag{6}$$

$$\hat{X}_{k+1} \ = \ \bar{X}_{k+1} \ + \ \mathrm{K} \ * \ (Z_{\mathrm{k+1}} - \ \mathrm{H} \ * \ \bar{X}_{k+1}) \tag{7}$$

$$\hat{P}_{k+1} = \bar{P}_{k+1} \ - \mathrm{K} * \mathrm{H} * \bar{P}_{k+1} \tag{8}$$

2.2 The unscented Kalman filter

Estimation in nonlinear systems is very important because most of the practical systems tend to have some nonlinearity. Estimation in nonlinear systems can be difficult and any practical estimator needs to use an approximation of some kind. This opened up the door to a lot of types of approximations to be used to solve the problem. The extended Kalman filter (EKF) have been the most used solution to apply the KF to nonlinear systems. The EKF linearizes the nonlinear transformation and uses Jacobian matrices for the linear transformation in the KF equations. The EKF suffers from a number of issues and limitations. Linearized transformations are only reliable if the error propagation can be well approximated by a linear function. To use Jacobians, they must exist and this is not always the case. Calculating Jacobian matrices is not easy and can be complicated, making room for more errors that are difficult to debug. To overcome these limitations, the Unscented Transformation (UT) propagates mean and covariance information through the nonlinear transformation. It is inspired from the idea that it is easier to approximate a probability distribution than a nonlinear function. A set of carefully chosen sigma points is propagated through the nonlinear

function. The statistics of the transformed points can then be calculated to produce an estimate of the nonlinearly transformed mean and covariance[15]. An illustration is shown in figure (6).

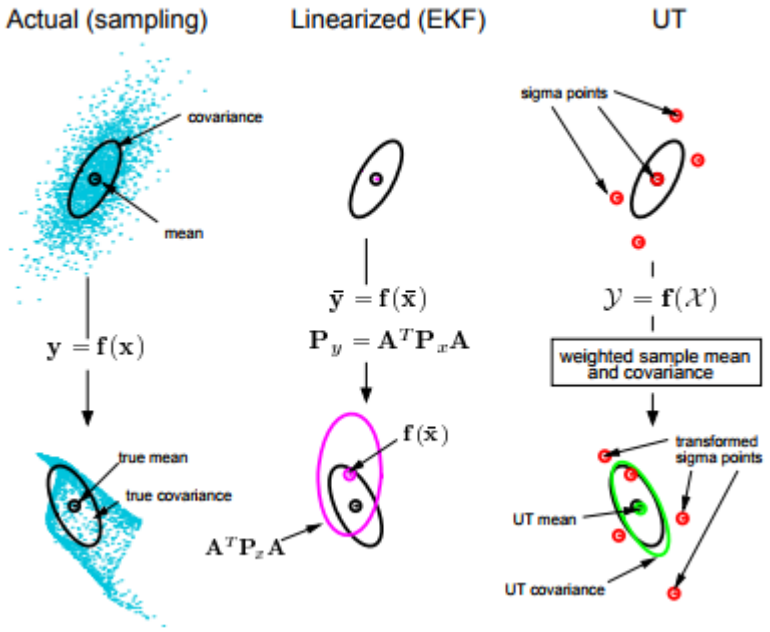


Figure 6: Example of the UT for mean and covariance propagation. a) Actual, b) First-order linearization (EKF), c) UT.[16]

The set of sigma points is chosen in order to satisfy certain properties. This set consist of a p+1 with their assigned weights { $x^{(i)}$ $W^{(i)}$ For i=0,1…,p}. The weights must satisfy:

$$\sum_{i=0}^{p} W^i = 1 \tag{9}$$

Each point is propagated through the function. The mean of the set of transformed points is calculated by the weighted average. The covariance is the weighted outer product of the transformed set points.

[15] The unscented Kalman filter content is taken from Julier, S.J., Uhlman, J.K. Unscented Filtering and Nonlinear Estimation.(2004)
[16] Wan, A., Eric, Van der Merwe, Rudolph. The unscented Kalman filter. (2010)

$$Z^i = h(X^i) \tag{10}$$

$$\bar{Z} = \sum_{i=0}^{p} W^i * Z^i \tag{11}$$

$$P_{zz} = \sum_{i=0}^{p} W^i * \left(Z^i - \bar{Z}\right) * (Z^i - \bar{Z})^T \tag{12}$$

2.2.1 Sigma points frame work

There are several ways to choose the set of sigma points[17]. It is usually a symmetric set. But because the UT offers flexibility to get information beyond the covariance and mean, it is possible to choose a set that uses additional information about the error distribution. Two sigma sets are presented below, Sigma-set1 and Sigma-set3:

Sigma-set 1

The first sigma set is a symmetric set

$$p = 2 * N_x \tag{13}$$

$$W^i = \frac{1}{2N_x} \tag{14}$$

$$P = U * U^T \tag{15}$$

Where $U=[\underline{u}^1\ \underline{u}^2 \cdots \underline{u}^{Nx}]$

$$X^i = \bar{X} + \sqrt{N_x} * u^i \tag{16}$$

[17] Julier, S.J., Uhlman, J.K. Unscented Filtering and Nonlinear Estimation. (2004)

$$X^{i+N_x} = \bar{X} - \sqrt{N_x} * u^i \tag{17}$$

For i=1,2,…,2N_x

$(\sqrt{N_x * P_x})_i$ is the i-th row or column of the matrix square root of $N_x * P_x$. If the matrix square root A of P is of the form $P = A^T * A$, then the sigma points are formed from the rows of A. However, if the matrix square root is of the form $P = A*A^T$, the columns of A are used.

Sigma-set 3

The set of sigma points can be extended with another point, the mean $\bar{x}$ and its weight W^0. Adding this point provides a parameter for controlling some aspects of the higher moments of the distribution of sigma points without affecting the mean and covariance[18].

$$p = 2 * N_x + 1 \tag{18}$$

$$X^0 = \bar{X} \tag{19}$$

$$W^i = \frac{1 - W^0}{2 * N_x} \tag{20}$$

$$X^i = \bar{X} + \sqrt{P_x * \frac{N_x}{(1 - W^i)}} \tag{21}$$

$$X^{i+N_x} = \bar{X} - \sqrt{P_x * \frac{N_x}{(1 - W^i)}} \tag{22}$$

For i=1,2,…,2N_x

[18] Both sigma-sets methods are taken from Julier, S.J., Uhlman, J.K. Unscented Filtering and Nonlinear Estimation. (2004)

2.2.2 The UKF algorithm

Given a system:

$$X_{\mathrm{k+1}} = \mathrm{f}(X_{\mathrm{k}}, U_{\mathrm{k}}, w_{\mathrm{k}}) \tag{23}$$

$$Z_{\mathrm{k+1}} = g(X_{\mathrm{k+1}}, v_{\mathrm{k+1}}) \tag{24}$$

$$E(X_0) = \bar{X}_0; \quad E(w_{\mathrm{k}}) = 0; \quad E(v_{\mathrm{k}}) = 0 \tag{25}$$

$$E(X_0 w_k^T) = \bar{X}_0; \quad E(X_0 v_k^T) = 0; \quad E(v_k w_k^T) = 0$$

$$cov(X_0) = P_0; \quad cov(w_k w_k^T) = \delta_{kl} * R; \quad cov(v_k v_k^T) = \delta_{kl} * Q$$

The most general formulation for the UKF augments the state vector with the process and observation noise terms:

$$X_a = \begin{bmatrix} X \\ w_k \\ v_k \end{bmatrix} \tag{26}$$

The augmented state is a $(N_x+N_w+N_v)*1$ vector.

The UKF algorithm is summarized as:

1) The set of sigma points is generated using a selection algorithm like the examples in section 2.2.1 from the augmented $\hat{X}_a$ and $\hat{P}_a$:

$$\hat{X}_a = \begin{bmatrix} \hat{X} \\ 0_{N_w*1} \\ 0_{N_v*1} \end{bmatrix} \tag{27}$$

$$\hat{P}_a = \begin{bmatrix} \hat{P}_x & 0 & 0 \\ 0 & R & 0 \\ 0 & 0 & Q \end{bmatrix} \tag{28}$$

2) The sigma points are propagated through the function:

$$\bar{X}_{k+1}^{i} = \mathrm{f}(X_k^i, U_\mathrm{k}) \tag{29}$$

3) The predicted mean is computed from the weighted average of the transformed points:

$$\bar{X}_{k+1} = \sum_{i=0}^{p} W^i \, \bar{X}_{k+1}^{i} \tag{30}$$

4) The predicted covariance is computed as:

$$\bar{P}_{k+1} = \sum_{i=0}^{p} W^i * (\bar{X}_{k+1}^{i} - \bar{X}_{k+1})(\bar{X}_{k+1}^{i} - \bar{X}_{k+1})^T \tag{31}$$

5) The prediction points are propagated through the observation model:

$$Z_{k+1}^{i} = \mathrm{g}(\bar{X}_{k+1}^{i}, U_\mathrm{k}) \tag{32}$$

6) The predicted observation is calculated as:

$$\bar{Z}_{k+1} = \sum_{i=0}^{p} W^i \, Z_{k+1}^{i} \tag{33}$$

7) The innovation matrix is calculated as:

$$\bar{P}_{zz}(k+1) = \sum_{i=0}^{p} W^i * (Z_{k+1}^{i} - \bar{Z}_{k+1})(Z_{k+1}^{i} - \bar{Z}_{k+1})^T \tag{34}$$

8) The cross covariance matrix is calculated as:

$$\bar{P}_{xz}(k+1) = \sum_{i=0}^{p} W^i * (\bar{X}^i_{k+1} - \bar{X}_{k+1})(Z^i_{k+1} - \bar{Z}_{k+1})^T \tag{35}$$

9) The last update can be performed using the normal Kalman equations:

$$K = \bar{P}_{xz}(k+1) * \bar{P}_{zz}(k+1)^{-1} \tag{36}$$

$$\hat{X}_{k+1} = \bar{X}_{k+1} + \mathrm{K} * (Z_{\mathrm{k+1}} - \bar{Z}_{k+1}) \tag{37}$$

$$\hat{P}_{k+1} = \bar{P}_{k+1} - K * \bar{P}_{zz}(k+1) * K^T \tag{38}$$

2.3 The smoother unscented Kalman

The KF is a recursive algorithm providing the conditional expectation of the state x_k given all the observations up to the current time k. The smoother KF estimates the state using all the observations past and future[19]. The general idea is to run KF forward in time to estimate the mean and covariance ($\hat{x}_{forw}$,$\hat{P}_{forw}$) given past data, then a second KF is run backwards in time to provide a backward time predicted mean and covariance ($\hat{x}_{back}$,$\hat{P}_{back}$) given the future data. These two estimates are then used to produce the smoothed ($\hat{x}_{smooth}$,$\hat{P}_{smooth}$) as follows:

$$\hat{P}^{-1}_{smooth} = \hat{P}^{-1}_{forw} + \hat{P}^{-1}_{back} \tag{39}$$

$$\hat{X}_{smooth} = \hat{P}_{smooth} * (\hat{P}^{-1}_{back} * \hat{X}_{back} + \hat{P}^{-1}_{forw} * \hat{X}_{forw}) \tag{40}$$

[19] Wan, A., Eric, Van der Merwe, Rudolph. The unscented Kalman filter. (2010)

2.4 Position and orientation representation

2.4.1 Rotation in a plane

The coordinates of the point P in the XY coordinate system are (x_1,y_1). The coordinates system X'Y' is related to the XY system by a rotation by an angle α as shown in figure (7). The coordinates of the point P in the X'Y' system, are given by:

$$x'_1 = x_1 * cos\alpha + y_1 * sin\alpha \quad (41)$$

$$y'_1 = -x_1 * sin\alpha + y_1 * cos\alpha \quad (42)$$

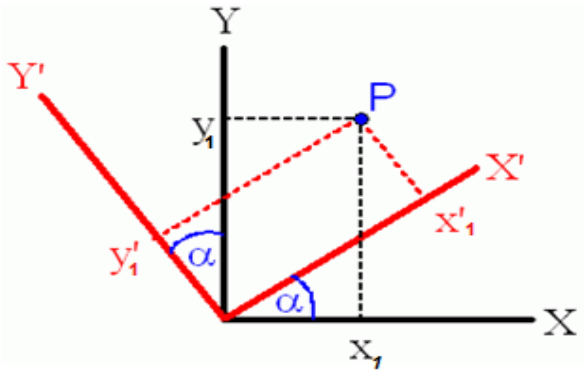


Figure 7: illustration of the the coordinates of P in both coordinate systems XY and X'Y'

The set of equations (Eq. 41,42) can be written as[20]:

$$P^1 = R_0^1 * P^0 \quad (43)$$

Where R is defined as:

$$R_0^1 = \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix} \quad (44)$$

[20] Spong, Mark W., Hutchinson, Seth, Vidyasagar, M., Robot Modeling and Control, pages 35-59. (2006)

P^1 is represented in frame (1), and P^0 is represented in frame (0).

The matrix R entries are derived in terms of the angle α, an alternative to obtain R is to project the axis of the X'Y' system into the XY system[21].

$$R_0^1 = \begin{bmatrix} \mathrm{X}'.\mathrm{X} & \mathrm{Y}'.\mathrm{X} \\ X'.\mathrm{Y} & \mathrm{Y}'.\mathrm{X} \end{bmatrix} \tag{45}$$

2.4.2 Rotation in 3D space

In the three dimensional case, the resulting matrix of the projection of the axis of the system X'Y'Z' into the system XYZ is[22]:

$$R_0^1 = \begin{bmatrix} \mathrm{X}'.\mathrm{X} & \mathrm{Y}'.\mathrm{X} & \mathrm{Z}'.\mathrm{X} \\ \mathrm{X}'.\mathrm{Y} & \mathrm{Y}'.\mathrm{Y} & \mathrm{Z}'.\mathrm{Y} \\ \mathrm{X}'.\mathrm{Z} & \mathrm{Y}'.\mathrm{Z} & \mathrm{Z}'.\mathrm{Z} \end{bmatrix} \tag{46}$$

$$R_1^0 = \begin{bmatrix} \mathrm{X}.\mathrm{X}' & \mathrm{Y}.\mathrm{X}' & \mathrm{Z}.\mathrm{X}' \\ X.\mathrm{Y}' & \mathrm{Y}.\mathrm{Y}' & \mathrm{Z}.\mathrm{Y}' \\ X.\mathrm{Z}' & \mathrm{Y}.\mathrm{Z}' & \mathrm{Z}.\mathrm{Z}' \end{bmatrix} \tag{47}$$

Since the dot product is commutative:

$$R_0^1 = (R_1^0)^T \tag{48}$$

[21] Spong, Mark W., Hutchinson, Seth, Vidyasagar, M., Robot Modeling and Control, pages 35-59. (2006)
[22] Spong, Mark W., Hutchinson, Seth, Vidyasagar, M., Robot Modeling and Control, pages 35-59. (2006)

Geometrically the orientation of the XYZ with respect to the frame X'Y'Z' is the inverse of the X'Y'Z' orientation with respect to the XYZ.

$$(R_0^1)^{-1} = (R_0^1)^T \tag{49}$$

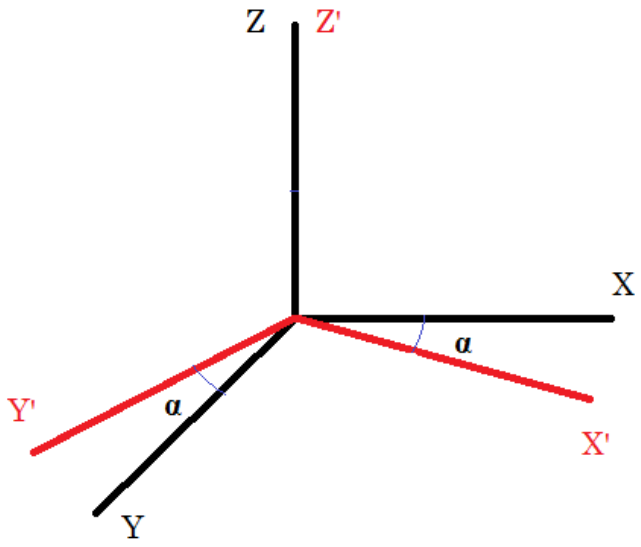


Figure 8: Basic rotation about the Z axis with an angle α

The basic rotation R_Z (figure (8)) about the Z axis is given by[23]:

$$R_0^1 = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{50}$$

The basic rotation R_X about the x-axis with angle α is given by[24]:

$$R_0^1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{bmatrix} \tag{51}$$

[23] Spong, Mark W., Hutchinson, Seth, Vidyasagar, M., Robot Modeling and Control, pages 35-59. (2006)
[24] Spong, Mark W., Hutchinson, Seth, Vidyasagar, M., Robot Modeling and Control, pages 35-59. (2006)

The basic rotation R_Y about the y-axis with angle α is given by[25]:

$$R_0^1 = \begin{bmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \\ \sin\alpha & 0 & \cos\alpha \end{bmatrix} \quad (52)$$

The matrix R is called a rotation matrix, and belongs to a group of matrices with a number of special properties. The set of n*n rotation matrices is known as the special orthogonal group of order n, and is denoted SO(n). For every R ϵ SO(n) the following properties hold[26]:

$$R^{-1} = R^T \quad (53)$$

$$Det(R) = 1 \quad (54)$$

2.4.3 Parameterization of R

The nine elements in a general rotation matrix R are not independent quantities. A rigid body has at most three rotational degrees of freedom, and therefor three quantities at most are necessary to form or describe this rotation matrix. A common method to specify the orientation of the XY coordinates system relative to the system X'Y' is Euler angles (ϕ, θ, ψ)[27]. First a rotation about the z axis by the angle ϕ, then a rotation about the current y-axis with an angle θ and finally a rotation about the current z-axis with an angle ψ as illustrated in figure(9).

[25] Spong, Mark W., Hutchinson, Seth, Vidyasagar, M., Robot Modeling and Control, pages 35-59. (2006)
[26] Spong, Mark W., Hutchinson, Seth, Vidyasagar, M., Robot Modeling and Control, pages 35-59. (2006)
[27] Spong, Mark W., Hutchinson, Seth, Vidyasagar, M., Robot Modeling and Control, pages 35-59. (2006)

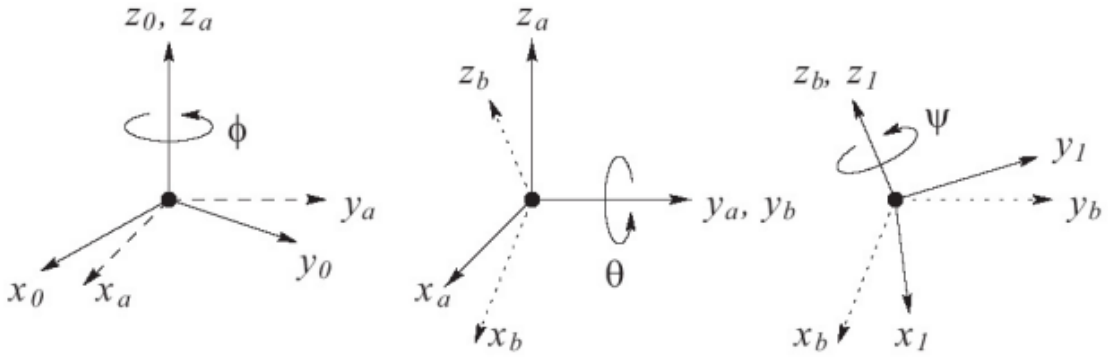


Figure 9: consecutive rotations about current axis[28]

The resulting transformation matrix can be generated from the product:

$$R_{zyz} = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} * \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (55)$$

Another method of representing a rotation matrix is the roll, pitch and yaw angles. A rotation matrix can be described as the product of successive rotation about the fixed principal XYZ axis in a specific order.

$$R_{zyx} = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{bmatrix} \quad (56)$$

2.4.4 Deriving a rotation matrix

Computing the relative velocity transformation between coordinate frames involve calculating the derivative of the rotation matrix. Introducing the concept of a skew matrix simplifies many of the computations required.

[28] Spong, Mark W., Hutchinson, Seth, Vidyasagar, M., Robot Modeling and Control, page 50. (2006)

The skew symmetric

An n*n matrix S is skew symmetric if and only if:

$$S^T + S = 0 \tag{57}$$

If S is a 3*3 skew symmetric matrix with components $S_{ij,}$ i,j=1,2,3 then (Eq. 57) is equivalent to:

$$S_{ij} + S_{ji} = 0 \tag{58}$$

For i,j=1,2,3

From (Eq. 58) the skew symmetric matrix S has the form:

$$S = \begin{bmatrix} 0 & -S_3 & S_2 \\ S_3 & 0 & -S_1 \\ -S_2 & S_1 & 1 \end{bmatrix} \tag{59}$$

For a vector $w=[w_x,w_y,w_z]^T$ the skew symmetric matrix S(w) is defined[29] as:

$$S(w) = \begin{bmatrix} 0 & -w_z & w_y \\ w_z & 0 & -w_x \\ -w_y & w_x & 1 \end{bmatrix} \tag{60}$$

[29] Spong, Mark W., Hutchinson, Seth, Vidyasagar, M., Robot Modeling and Control, pages 121-122. (2006).

Considering the general case of angular velocity about a possibly moving axis. For $R = R(t) \epsilon SO(3)$, assuming that R is continuously differentiable as a function of time. The derivative of R can be written[30] as:

$$\dot{R}(t) = S\big(w(t)\big) * R(t) \tag{61}$$

S(w(t)) is a skew symmetric matrix. The vector w(t) is the angular velocity of the rotating frame with respect to the fixed frame.

[30] Spong, Mark W., Hutchinson, Seth, Vidyasagar, M., Robot Modeling and Control, page 125. (2006)

3 The ROV setup

3.1 Introduction

The ROV is operated by a ROV pilot using gyroscope readings for heading, and a camera for navigation. The ROV is equipped with 2 vertical and 4 horizontal thrusters, and actuated in 4 degrees of freedom (surge, sway, heave and yaw). The DVL system is installed in the front facing the net pen. All the readings from the instrument are represented in the body reference frame. The sensors used are presented below with emphasis on the DVL system.

3.2 The Pressure sensor

For measuring depth, it is very common to measure the pressure and use hydrostatic pressure formula:

$$P = \rho * g * h \tag{62}$$

ρ is the fluid density and g is the gravitational acceleration and h the height

These sensors are usually very accurate and have small errors.

3.3 The gyroscope

A single axis gyroscope (yaw) is used to measure the angular rate of the ROV.

3.4 GPS

A GPS is used to get position measurements when at the surface. It is used mainly to measure the initial and final position for the travelled path.

3.5 The Doppler Velocity Log (DVL)

3.5.1 The Doppler effect

The Doppler effect can be observed when a source of waves is in relative motion to the observer. It can be described for a sound source as an upward shift in frequency for the observer towards whom the source approaches and a downward shift in frequency for the observer from whom the source recedes. Measuring the pitch change allow for measuring the speed of the source relative to the observer[31].

Sound consists of pressure waves in air, water or solids. The speed of sound is related to the frequency as follows:

$$C = f * \lambda \tag{63}$$

C: The speed of sound.

f: The frequency of sound.

λ: The wave length.

The equation for the Doppler shift in this situation[32] is

$$F_d = F_s * \frac{V}{C} \tag{64}$$

F_d: The Doppler shift frequency.

F_s: The frequency of sound when everything is still

V: The relative velocity between the observer and the source.

C: the speed of sound.

[31] Teledyne RD Instruments. Acoustic Doppler Current Profiler Principles of Operation: A Practical Primer.(2011)
[32] Teledyne RD Instruments. Acoustic Doppler Current Profiler Principles of Operation: A Practical Primer.(2011)

3.5.2 DVL

The Doppler velocity log is a hydro-acoustic Doppler instrument. It has four acoustic beams Oriented in a convex diverging configuration with a known beam angle, typically 20 or 30º. The beams send out 4 beam pings and measure the resulting response as frequency shift or Doppler shift. Because the DVL both transmits and receives sound the Doppler shift is doubled. The DVL sees the reflecting surface or object as a sound source[33].

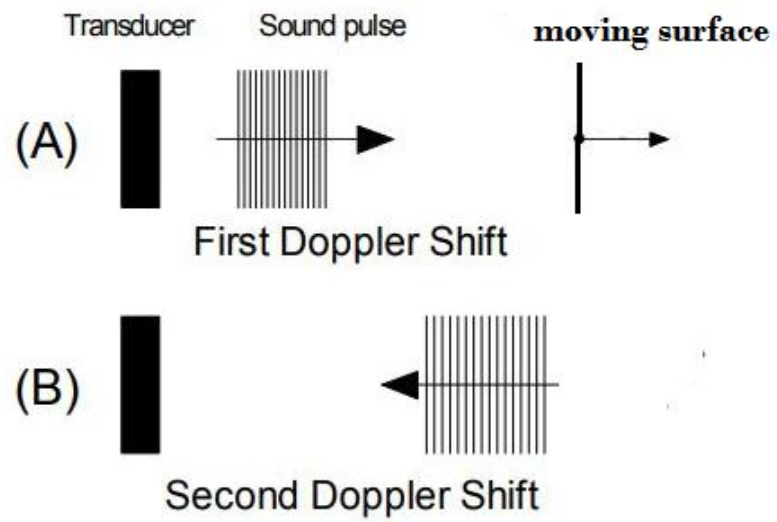


Figure 10: Backscattered sound includes two Doppler shifts, (A) one on the way to the surface, and (B) on the way back.[34]

The reflected sound consists of two Doppler shifts as shown in figure (10). The one on the way to the surface (A), and the second on the way back after reflection (B).[35]

$$F_d = 2 * F_s * \frac{V}{C} \tag{65}$$

F_d: The Doppler shift frequency.

F_s: The frequency of sound when everything is still

V: The relative velocity between the observer and the source.

C: the speed of sound.

[33] Teledyne RD Instruments. Acoustic Doppler Current Profiler Principles of Operation: A Practical Primer. (2011)
[34] Teledyne RD Instruments. Acoustic Doppler Current Profiler Principles of Operation: A Practical Primer. (2011)
[35] Teledyne RD Instruments. Acoustic Doppler Current Profiler Principles of Operation: A Practical Primer. (2011)

The Doppler shift works only when sound sources and receivers get closer or further from each other. The DVL measures radial motion, then converts it to relative velocity with respect to the reflecting surface, represented in the instrument frame. Among a long list of measurements a DVL system can provide ranging from temperature and depth to acoustic echo intensity and current profiling, this project uses only the relative velocity and the beam distances.

An example of DVL sensors is shown in figure (11).

Figure (12) illustrate beams' placement.

Figure 11: Teledyne DVL.[36]

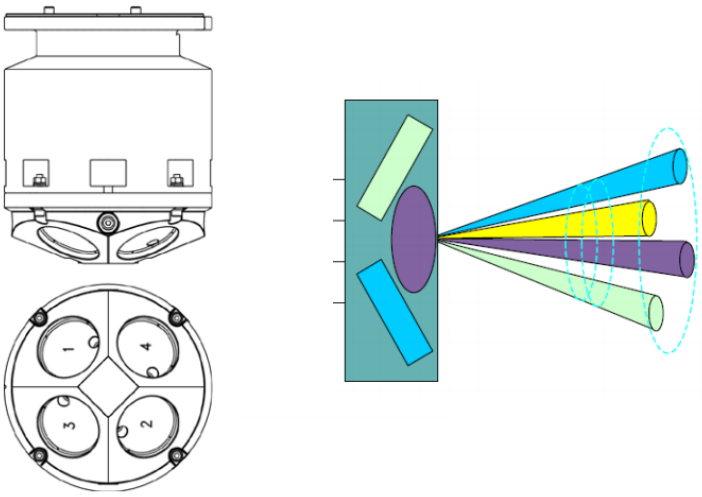


Figure 12: Illustration of the beam placement[37]

[36] Teledyne RD Instruments. Acoustic Doppler Current Profiler Principles of Operation: A Practical Primer. (2011)
[37] Teledyne RD Instruments. Acoustic Doppler Current Profiler Principles of Operation: A Practical Primer. (2011)

4 Design and Method

4.1 Introduction

The procedure is divided into four main tasks:

- Defining coordinate systems.
- Generating the ROV true path.
- Modeling sensors and generating measurement data.
- Position estimation.

The assumptions and considerations made to implement each task are presented below and further elaborated in connection with each task:

- The ROV has no pitch or roll motion.
- The net coincides with the world earth fixed frame.
- The net is a vertical plane.
- Additive process and measurement white Gaussian noise.
- The DVL manages to see the net as a seabed.

All simulations are performed in Matlab.

4.2 Coordinate systems

In order to represent the relative position and orientation of a rigid body with respect to another, it is possible to attach a coordinate frame to each of them and then study the geometric relationship between these two frames[38]. For this assignment, three coordinate systems are considered due to the nature of the sensors used for measurements. The gyroscope and pressure sensor outputs are inertial, while the DVL outputs are relative to the net pen.

The coordinate systems are shown in figure (13). The world coordinate frame is earth fixed, the z axis is pointing down, following the net rope used for navigation by the pilot. The net reference frame is attached to the net pen. The x-axis is perpendicular to the net, the net is on the plane (yz). The third frame is the body reference frame and it is attached to the moving ROV. The x-axis points from aft to fore, y-axis directed from port to starboard and z-axis pointing from top to bottom.

Two assumptions are made before moving forward:

The net pen frame coincides with the world coordinate frame at the beginning of the simulation and the displacement and orientation changes of the net relative to the world frame during the motion sequence is ignored. This assumption allows using the inertial measurement as if they were represented in the net frame.

Not making this this assumption will include an additional term to the calculations. Assuming it is possible to obtain from measurements, the transformation matrix from the world coordinates frame to the net frame, which describes the position and pose of the net relative to the world coordinate frame. In this case, since the pressure sensor and gyroscope readings are represented in the world coordinate frame, they need to be transferred to the net frame as described in section 2.4.2.

The second assumption is that the DVL system and the ROV share the same body reference frame. The ROV and sensors mounting spots are rigidly attached to each other. Ignoring the distance between the center of the DVL instrument coordinate frame and the center of gravity of the ROV simplifies kinematic calculations.

[38] Spong, Mark W., Hutchinson, Seth, Vidyasagar, M., Robot Modeling and Control, page 38. (2006)

Not making this assumption adds a transformation matrix that be can be used to transform the DVL readings to the ROV/body reference frame.

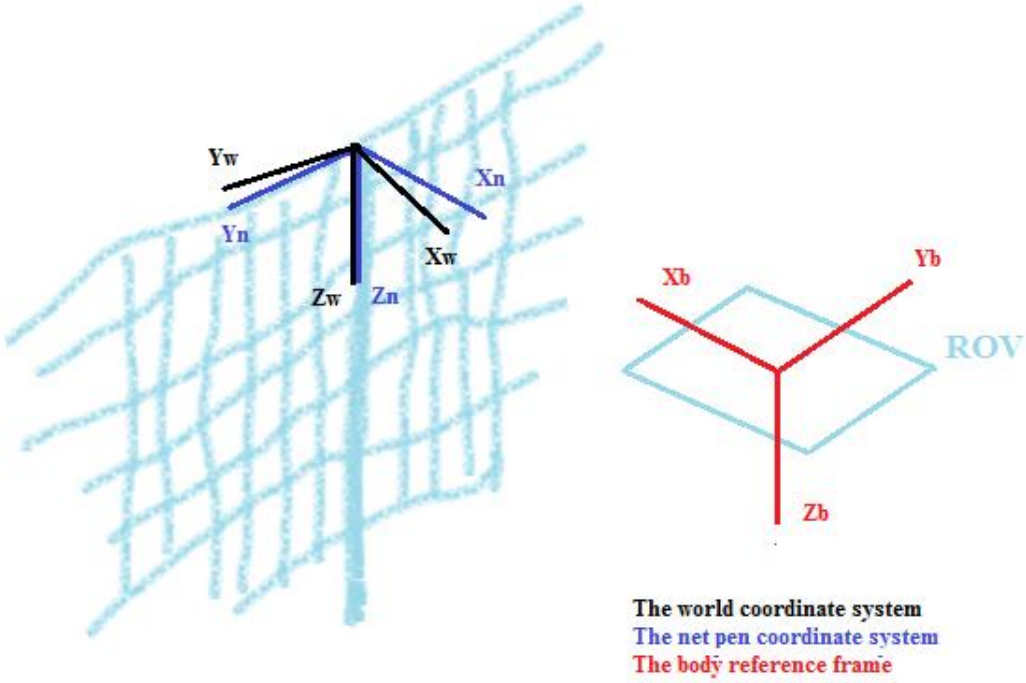


Figure 13: Coordinate systems used and the initial position of the ROV. The world coordinates frame and the net frame do not coincide here just for illustration.

4.3 True path generator

The ROV undergoes the following sequence of motion: on-the surface, dive down the bottom, go up the surface. The pilot dives down the bottom following a rope (also the net frame z-axis). The ROV is moving parallel to the net (The (yz) plane of the net frame) until it finds a second rope and then goes up to the surface. The input from the ROV operator is assumed to be some form of torque that results in a velocity component U. The ROV is assumed to follow the sequence at a constant velocity of 1m/s. The total covered distance is 600m, 200 meters down, followed by 200m in the direction from port to starboard and finally 200m up. The meter is used here as a virtual unit of distance. ROVs used for aquaculture net inspection tasks don't need to go that deep.

In order to model all the external factors like waves, water currents, operator mistakes and the fishnet unpredictable shape, that may affect the predicted noise-free path, a sinus function is

used for simulating the true ROV trajectory during the sequence of motion as shown in figure(14).

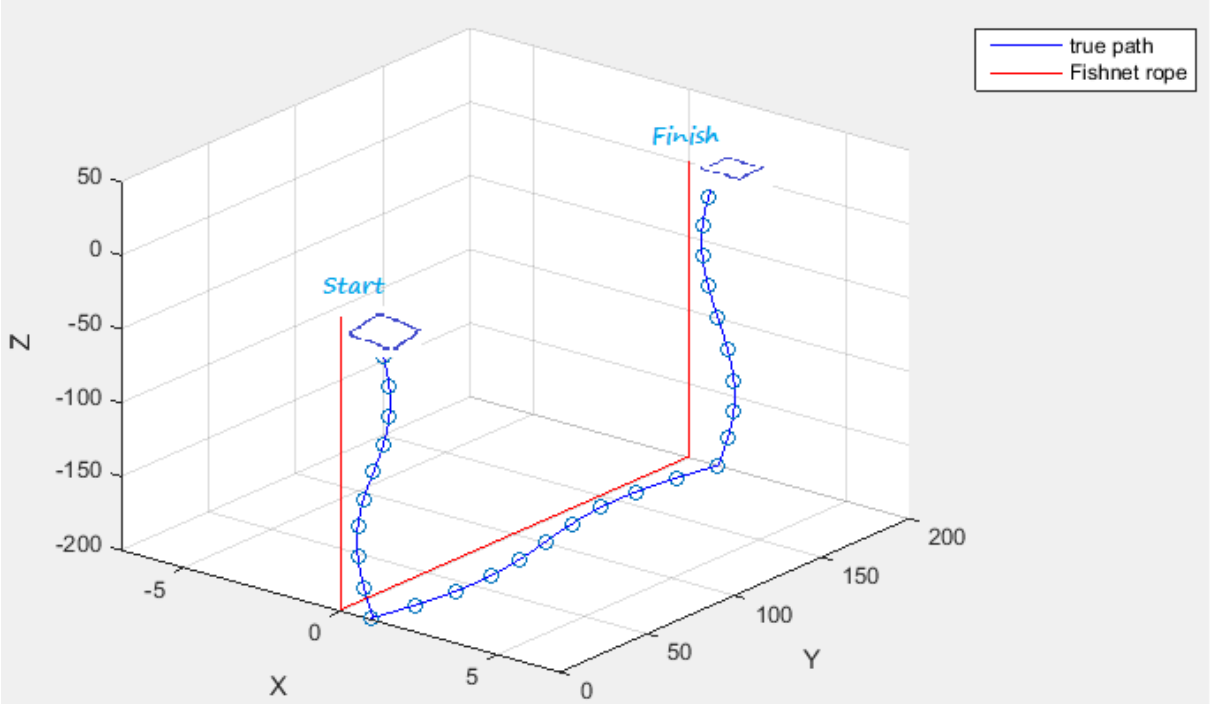


Figure 14: The ROV true path during the sequence of motion.

The ROV operator tries to keep the ROV heading neutral (yaw angle=0) using the input video from the camera facing the rope and a gyroscope. The yaw angle is smoothly varying around neutral heading. The true yaw angle (heading) is modeled using a sinus function with amplitude varying between 0.3 and -0.3 rad (±17 degrees).

The generated path and yaw angles excite all the four degrees of freedom the ROV can achieve.

The ROV true yaw and path generator are implemented in the Main function, Appendix A (8.2).

4.4 Modeling the sensors

The data from the true path generator were used to generate sensor measurements. Although most sensors can run at higher frequency, due to data loss and biomass interaction, acoustic instruments can suffer from, the sensors are assumed to take measurements each second.

The ROV is assumed to not have any pitch or roll motions. Only a single axis gyro is available, which means there is no measurements available for other rotations. This assumption also simplifies the transformation matrix between the net and the body frames.

The DVL system manages to see the net pen as a sea bed. This assumption has been validated by practical experiences like in the experiment done by Rundtop and Frank. [39]

The fish cages tend to utilize different geometries. For this assignment, the net is assumed to be a vertical plane that can represent a partition of a bigger design.

All measurements noises are modeled as additive white noise (Gaussian).

4.4.1 Gyroscope

The gyroscope angular rate is calculated from the generated true yaw angle. The measurement model contains two types of sensor errors, an additive white noise with rapid fluctuation and mean 0 and a stable sensor bias. The angular velocity is calculated as follow:

$$\dot{\phi}^{true}(k) = \phi(k+1) - \phi(k) \tag{66}$$

$$\dot{\phi}^{meas}(k) = \dot{\phi}^{true}(k) + bias + v(k) \tag{67}$$

$\dot{\phi}^{meas}$ stands for the measured angular rate, $\dot{\phi}^{true}$ stands for the true angular rate, ϕ for the true yaw angle and v(k) stands additive white noise.

The sensor model is given in Appendix B (8.3).

[39] Rundtop, Per, Frank, Kevin. Experimental evaluation of hydroacoustic instruments for ROV navigation along aquaculture net pens. (2016)

4.4.2 Pressure

The pressure sensor uses the data from the path generator to calculate the depth or the z component. This type of sensors tends to be pretty accurate. The measurement model contains an additive white noise element.

$$z^{meas}(k) = z^{true}(k) + v(k) \tag{68}$$

z^{meas} is measured depth, z^{true} is the true depth and v(k) is Gaussian white noise.

The sensor model is given in Appendix C (8.4).

4.4.3 The DVL relative velocity

The DVL calculates the net velocity relative to the ROV, represented in the body frame. The DVL sensor model uses the data from the path generator in addition to the yaw angle and angular rate to transform the velocity components into the body frame.

The relative velocity of A with respect to B is related to the relative velocity of B with respect to A as follow:

$$V_{A/B} = -V_{B/A} \tag{69}$$

The velocity of the ROV relative to the net can be calculated from the generated path data, as it is simply the velocity of the ROV represented in the net frame. The true yaw angle is used to transform this velocity from the net frame (0) to the ROV/body frame (1).

The rotation matrix between the net frame and the ROV/body frame is R_1^0. The number (0) is used for the net frame and the number (1) is used for the ROV/Body frame.

From the initial frame positions and orientations shown in figure(13), the body frame has an angle offset of π. Since only yaw rotation is taken into consideration the rotation matrix R_1^0 can be written as a basic rotation about the z axis from section 2.4.2.

$$R_1^0 = \begin{bmatrix} \cos(\phi + \pi) & -\sin(\phi + \pi) & 0 \\ \sin(\phi + \pi) & \cos(\phi + \pi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (70)$$

$$= \begin{bmatrix} -\cos\phi & \sin\phi & 0 \\ -\sin\phi & -\cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

O_0 and O_1 are the origins of the coordinate frames (0) and (1) respectively. Applying (Eq. 43) to the vector $\overrightarrow{O_0O_1}$ =P gives:

$$P^0 = R_1^0 * P^1 \quad (71)$$

Differentiating both sides of (Eq. 71) with respect to time using the product rule gives:

$$\dot{P}^0 = \dot{R} * P^1 + R * \dot{P}^1 \quad (72)$$

Applying the formula for the rotation matrix derivative (Eq. 61) gives:

$$\dot{P}^0 = S(\dot{\phi}) * R_1^0 * P^1 + R_1^0 * \dot{P}^1 \quad (73)$$

Using (Eq. 71) gives:

$$\dot{P}^0 = S(\dot{\phi}) * P^0 + R_1^0 * \dot{P}^1 \quad (74)$$

The vector $\dot{\phi}$ here is the angular velocity of the moving ROV frame (1) with respect to the net frame (0) and represented in frame (1). Since there is no pitch and roll motions, the vector $\dot{\phi}=[0,0, \dot{\phi}_z]^T$ contains only the yaw angular rate.

Substituting in (Eq. 60) gives:

$$S(\dot{\phi}) = \begin{bmatrix} 0 & -\dot{\phi}z & 0 \\ \dot{\phi}_z & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (75)$$

$\dot{P}^0$ is the time derivative of the position vector $\overrightarrow{O_0O_1}$=P represented in frame (0). Since O_0 is the origin of frame (0) and O_1 is the origin of frame (1), this vector represents the velocity of the ROV represented in the net frame (0), noted V_1^0.

$\dot{P}^1$ is the time derivative of the position vector $\overrightarrow{O_0O_1}$=P = -$\overrightarrow{O_1O_0}$ represented in frame (1). This vector represents the velocity of the net represented in the ROV/body frame, noted V_0^1.

Note there is a minus sign:

$$\dot{P}^1 = -V_0^1 \tag{76}$$

(Eq. 74) can now be written as:

$$V_1^0 = S(w) * P^0 - R_1^0 * V_0^1 \tag{77}$$

Finally the formula for calculating the relative speed of the net represented in the ROV/body frame and generating the relative velocity measurements from DVL is:

$$V_0^1(k) = R_0^1(k) * S\big(w(k)\big) * P^0(k) - R_0^1(k) * V_1^0(k) + v(k) \tag{78}$$

v(k) is additive Gaussian white noise.

4.4.4 The DVL beam distances

The DVL is also capable of measuring the beam distances (b_1,..,b_4) to the reflecting surface, the net. The beams readings are only dependent on the distance and angle between the ROV and the net. The beams don't see any change when the ROV is moving parallel to the net, i.e. in the y and z directions (sway and heave). The beam distances are calculated using simple trigonometry and relative position to the Body frame illustrated in figure (15) as follows:

$$b_1(k) = \frac{x}{cos\frac{\pi}{6} * \cos\left(\frac{\pi}{6} - \phi(k)\right)} + v_1(k) \quad (79)$$

$$b_2(k) = \frac{x}{cos\frac{\pi}{6} * \cos\left(\frac{\pi}{6} + \phi(k)\right)} + v_2(k)$$

$$b_3(k) = \frac{x}{cos\frac{\pi}{6} * \cos\left(\frac{\pi}{6} + \phi(k)\right)} + v_3(k)$$

$$b_4(k) = \frac{x}{cos\frac{\pi}{6} * \cos\left(\frac{\pi}{6} - \phi(k)\right)} + v_4(k)$$

x is the x coordinate of the ROV in the net frame.

The measurements error is modeled as an additive white noise parameter $v_i(k)$.

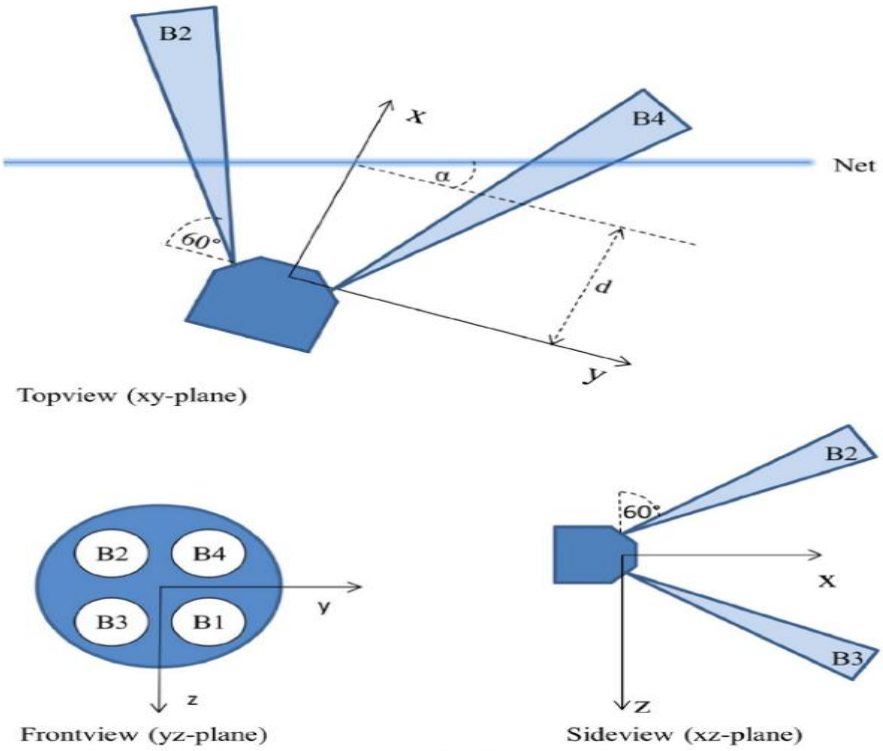


Figure 15: illustration of positioning of the beams and the body reference frame used.[40]

[40] Rundtop, Per, Frank, Kevin. Experimental evaluation of hydroacoustic instruments for ROV navigation along aquaculture net pens. (2016)

From the distances b_i it is possible to calculate the angle α shown in figure (16) as follow[41]:

$$a_1 = \frac{b_2 + b_3}{2} \tag{80}$$

$$a_2 = \frac{b_4 + b_1}{2} \tag{81}$$

Then,

$$AB = \frac{a_1}{\tan\frac{\pi}{3}} + \frac{a_2}{\tan\frac{\pi}{3}} \tag{82}$$

$$BC = a_2 - a_1 \tag{83}$$

$$\alpha = \text{atan}(\frac{BC}{AB}) \tag{84}$$

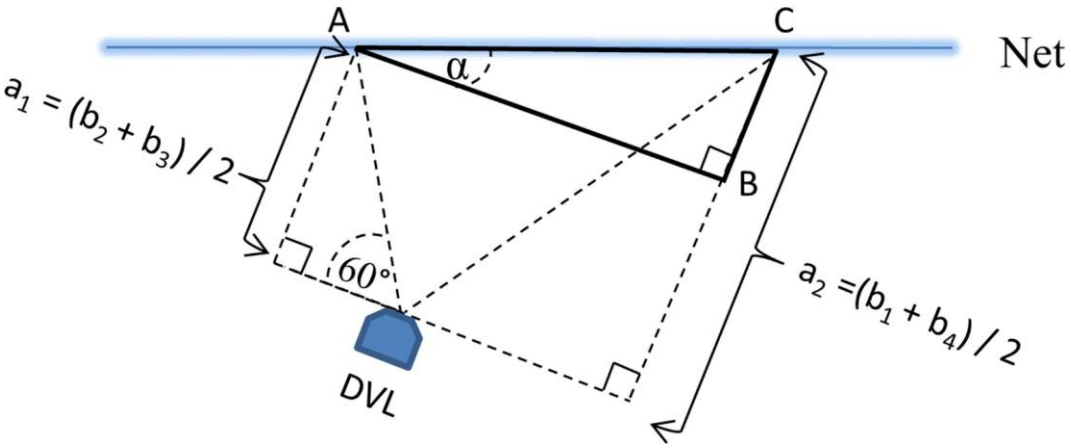


Figure 16: illustration of calculation of the angle α.[42]

The sensor model for the DVL measurement is given in Appendix D (8.5).

[41] Rundtop, Per, Frank, Kevin. Experimental evaluation of hydroacoustic instruments for ROV navigation along aquaculture net pens. (2016)
[42] Rundtop, Per, Frank, Kevin. Experimental evaluation of hydroacoustic instruments for ROV navigation along aquaculture net pens. (2016)

4.5 Position estimation

The filter's state space model consist of the position Pos=[x,y,z]^T of the ROV, its heading ϕ (yaw angle) and the angular rate $\dot{\phi}$, represented in the net frame (0):

The 5*1 state vector is defined as X=[Pos, ϕ , $\dot{\phi}$]^T

The filter process model is defined as:

$$Pos_{k+1} = A * Pos_k + B * U_k + w_{1,k} \tag{85}$$

$$\phi_{k+1} = \phi_k + \dot{\phi}_k * dt + w_{2,k} \tag{86}$$

$$\dot{\phi}_{k+1} = \dot{\phi}_k + w_{3,k} \tag{87}$$

A and B are identity matrices of size 3. w=[w_1 ,w_2 ,w_3]' is the 5x1 process white noise vector.

4.5.1 Filter 1: DVL relative velocity, gyroscope and pressure

The motion of the ROV is measured with a pressure sensor, the DVL relative velocity V_0^1=[V_x,V_y,V_z]^T and a gyroscope. The GPS provides initial position.

The measurement vector is defined as Z=[z^{meas}, V_0^1 , $\dot{\phi}^{meas}$]^T and the measurement model for the Kalman filter is defined as:

$$z_k^{meas} = z_k + v_{1,k} \tag{88}$$

$$V_0^1(k) = R_0^1(k) * S\left(\dot{\phi}(k)\right) * Pos(k) - R_0^1(k) * V_1^0(k) + v_{2,k} \tag{89}$$

$$\dot{\phi}_k^{meas} = \dot{\phi}_k + v_{3,k} \tag{90}$$

z^{meas} is represented in the net frame (0),z is z-component of the vector Pos=[x,y,z]^T, V_0^1 is the relative velocity of the net pen with respect to the ROV represented in the body frame (1), V_1^0 is the velocity vector of the ROV represented in the net frame (0) $V_1^0=\frac{d(Pos)}{dt}$, $\dot{\phi}^{meas}$ is represented in the net frame (0). v=[v_1 ,v_2, v_3]^T is the measurement white noise 5x1 vector.

From (Eq. 70) and (Eq. 48):

$$R_0^1 = (R_1^0)^T = \begin{bmatrix} -\cos\phi & -\sin\phi & 0 \\ \sin\phi & -\cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{91}$$

The ROV initial position and orientation at the surface is measured with the GPS and assumed known.

The initial state variable and covariance matrix are defined as:

$$X_0 = \begin{bmatrix} 2 \\ 2 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{92}$$

$$P_0 = \begin{bmatrix} 10^{-3} & 0 & 0 & 0 & 0 \\ 0 & 10^{-3} & 0 & 0 & 0 \\ 0 & 0 & 10^{-3} & 0 & 0 \\ 0 & 0 & 0 & 10^{-3} & 0 \\ 0 & 0 & 0 & 0 & 10^{-3} \end{bmatrix}$$

In order to include an implementation of the linear Kalman filter, the KF was first applied to the reduced system defined as follow:

Process model:

$$\phi_{k+1} = \phi_k + \dot{\phi}_k * dt + w_{2,k} \tag{93}$$

$$\dot{\phi}_{k+1} = \dot{\phi}_k + w_{3,k} \tag{94}$$

Measurement model

$$\dot{\phi}_k^{meas} = \dot{\phi}_k + v_{3,k} \tag{95}$$

The estimated yaw angle and angular rate are then fused into the system defined as follow:

Process model

$$Pos_{k+1} = A * Pos_k + B * U_k + w_{1,k} \tag{96}$$

Measurement model

$$V_0^1(k) = R_0^1(k) * S\left(\dot{\phi}(k)\right) * Pos(k) - R_0^1(k) * V_1^0(k) + v_{2,k} \tag{97}$$

As mentioned in section 2.2 above, the UKF has more advantages than the EKF and will be used for all filters in this assignment. The UKF was applied as described in section 2.2.2 using the sigma-set 3, described in section 2.2.1, to the reduced system ((Eq. 96), (Eq. 97)).The sigma points are computed from the augmented 9*1 state vector X_a and 9*9 covariance matrix $P_a$as follows:

$$X_a = \begin{bmatrix} x \\ y \\ z \\ 0_{N_w * 3} \\ 0_{N_v * 3} \end{bmatrix} \tag{98}$$

$$P_a = \begin{bmatrix} P_x & 0 & 0 \\ 0 & R & 0 \\ 0 & 0 & Q \end{bmatrix} \tag{99}$$

The linear KF filter is given in Appendix F (8.7).

Filter1 uses the linear KF yaw and angular rate estimates and is given in Appendix E (8.6).

4.5.2 Filter 2: DVL relative velocity, gyroscope, pressure and angle α

The ROV state dynamics filter model is the same as in section 4.5. The angle α is fused into the measurements vector.

The filter measurement model is defined as:

$$z_k^{meas} = z_k + v_{1,k} \tag{100}$$

$$V_0^1(k) = R_0^1(k) * S\left(\dot{\phi}(k)\right) * Pos(k) - R_0^1(k) * V_1^0(k) + v_{2,k} \tag{101}$$

$$\dot{\phi}_k^{meas} = \dot{\phi}_k + v_{3,k} \tag{102}$$

$$\alpha_k = \phi_k + v_{4,k} \tag{103}$$

The same initial conditions yield here.

The unscented Kalman filter was applied using sigma-sett3 as described in sections 2.2.1 and 2.2.2. The sigma points are computed from the augmented 16*1 state vector, and 16*16 augmented covariance matrix.

$$X_a = \begin{bmatrix} x \\ y \\ z \\ \phi \\ \dot{\phi} \\ 0_{N_w * 1} \\ 0_{N_v * 1} \end{bmatrix} \tag{104}$$

$$P_a = \begin{bmatrix} P_x & 0 & 0 \\ 0 & R & 0 \\ 0 & 0 & Q \end{bmatrix} \tag{105}$$

N_w=5 and N_v=6

Filter 2 is given in Appendix G (8.8).

4.5.3 Filter 3: DVL relative velocity, gyroscope, pressure, α and the beam distances.

A new set of measurements is fed into the UKF implementation done in section 4.5.2. The beams distances [b_1, b_2, b_3, b_4] are now added to the measurement vector [z_{meas}, V_0^1 , $\dot{\phi}_{meas}$, α]^T as follows:

$$Z = \begin{bmatrix} z_{meas} \\ V_x \\ V_y \\ V_z \\ \dot{\phi}_{meas} \\ \alpha \\ b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} \tag{106}$$

The filter process model is the same as defined in section 4.5.

The filter measurement model is:

$$z_k^{meas} = z_k + v_{1,k} \tag{107}$$

$$V_0^1(k) = R_0^1(k) * S\left(\dot{\phi}(k)\right) * Pos(k) - R_0^1(k) * V_1^0(k) + v_{2,k} \tag{108}$$

$$\dot{\phi}_k^{meas} = \dot{\phi}_k + v_{3,k} \tag{109}$$

$$\alpha_k = \phi_k + v_{4,k} \tag{110}$$

$$b_1(k) = \frac{x}{cos\frac{\pi}{6} * \cos\left(\frac{\pi}{6} - \phi(k)\right)} + v_5(k) \tag{111}$$

$$b_2(k) = \frac{x}{cos\frac{\pi}{6} * \cos\left(\frac{\pi}{6} + \phi(k)\right)} + v_6(k) \tag{112}$$

$$b_3(k) = \frac{x}{cos\frac{\pi}{6} * \cos\left(\frac{\pi}{6} + \phi(k)\right)} + v_7(k) \tag{113}$$

$$b_4(k) = \frac{x}{cos\frac{\pi}{6} * \cos\left(\frac{\pi}{6} - \phi(k)\right)} + v_8(k) \tag{114}$$

The initial conditions by the filter are the same as in section 4.5.3.

The Unscented Kalman filter was applied as described in section 2.2.2 using both the sigma-set 3 and sigma-set1 described in section 2.2.1. The sigma points are computed from the augmented 20x1 state vector:

$$X_a = \begin{bmatrix} x \\ y \\ z \\ \phi \\ \dot{\phi} \\ 0_{N_w * 1} \\ 0_{N_v * 1} \end{bmatrix} \tag{115}$$

$$P_a = \begin{bmatrix} P_x & 0 & 0 \\ 0 & R & 0 \\ 0 & 0 & Q \end{bmatrix} \tag{116}$$

N_w=5 and N_v=10

Filter 3 using both sigma sets (1 and 3) is given in Appendix H (8.9).

4.5.4 Filter 4: The smoother Kalman filter

The same unscented Kalman filter used in section 4.5.3 was run forward and backward in time. Since the dynamic state model is linear, the implementation was achieved by reversing the input U from the ROV operator used for the forward sequence and reversing all the velocity components measurements, in this case the relative velocity components from the DVL and the angular rate from the gyroscope.

The GPS works at the surface of the water and can provide very accurate position estimation. The final ROV position from GPS is used for the initial estimate of the smoother KF. The measurements are supposed to be noisy near the finishing position, the backward KF was

tuned to expect very noisy measurements in the end of the motion sequence, due to the degradation quality of the measurements, especially the DVL relative velocity.

State estimation was implemented using both forward and backward runs as described in section 2.3.

Filter 4 is implemented in Main, Appendix A (8.2) and the backward filter is given in Appendix I (8.10).

4.5.5 Summary

A summary of the all the filters and sensor fusion levels is presented in figure (17).

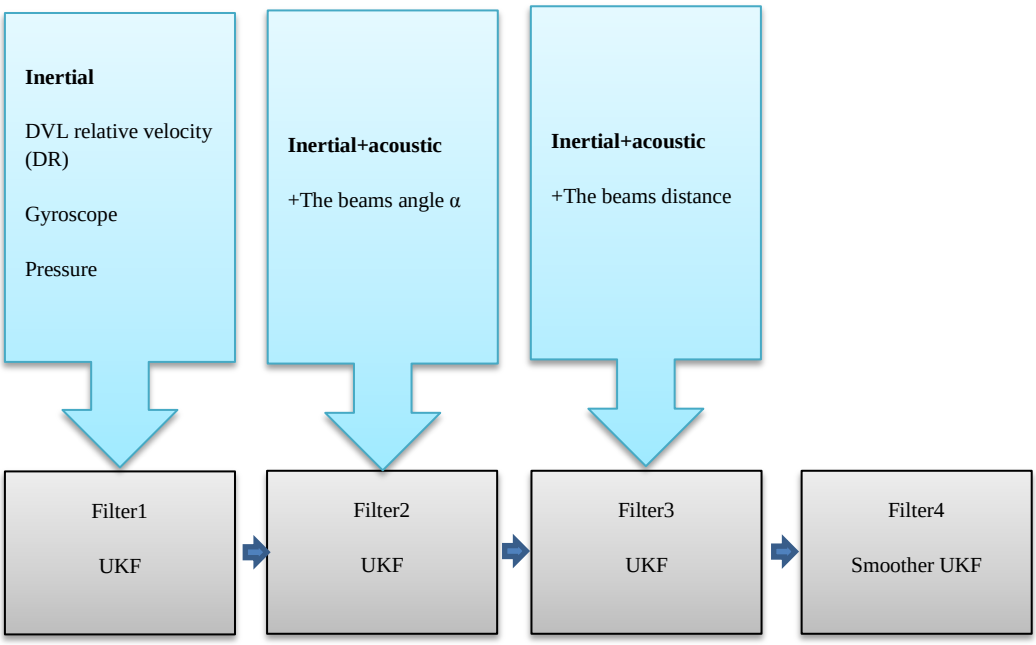


Figure 17: summary of the design process.

5 Results and discussion

5.1 Introduction

The results from the simulation of all filters are presented as follows:

- The estimation error in each axis (with standard deviation boundaries)
- The true path and the estimated path 3D plot
- The performance of Filter1 is compared to filter2
- The performance of filter2 is compared to Filter3
- The performance of filter3 is compared to filter4

To highlight the findings after each implementation, the results from each filter and filters comparison are discussed in the same chapter. The performance and accuracy of the pressure sensor makes the estimation of the z component very accurate and within reasonable limits for all filters and will not be discussed here.

5.2 Filter 1

Error in x direction:

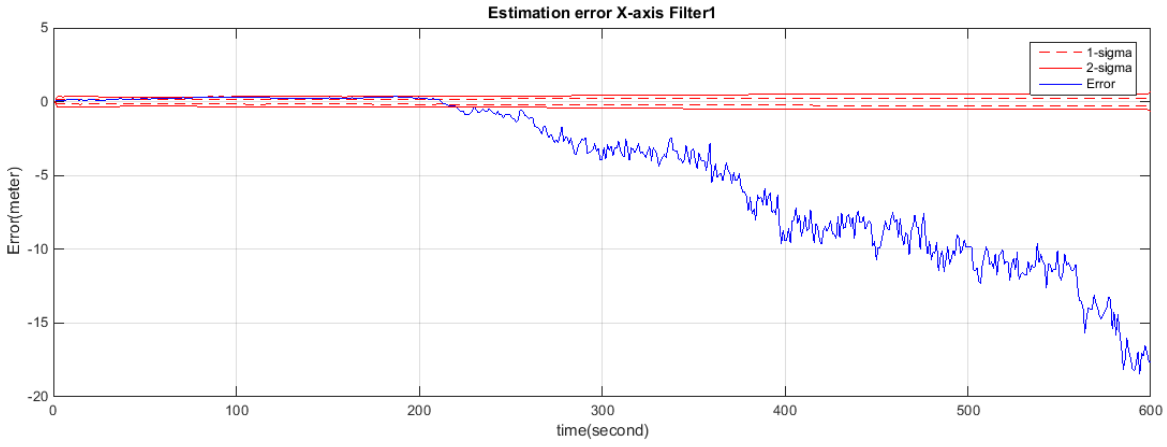


Figure 18: Estimation error in the x axis (surge). Both -sigma and 2-sigma boundaries are included.

Error in y direction:

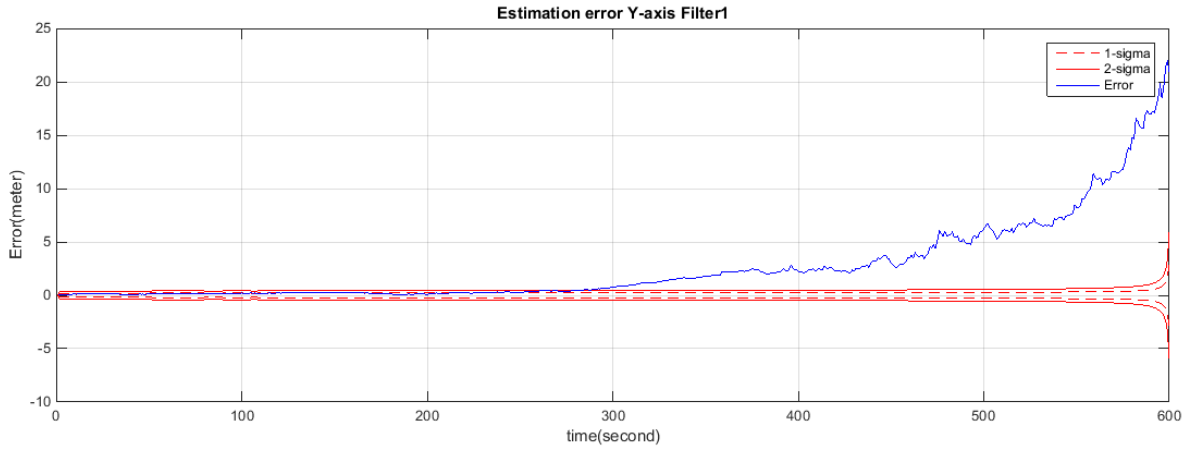


Figure 19: Estimation error in the y axis (sway). Both 1-sigma and 2-sigma boundaries are included.

Error in z direction:

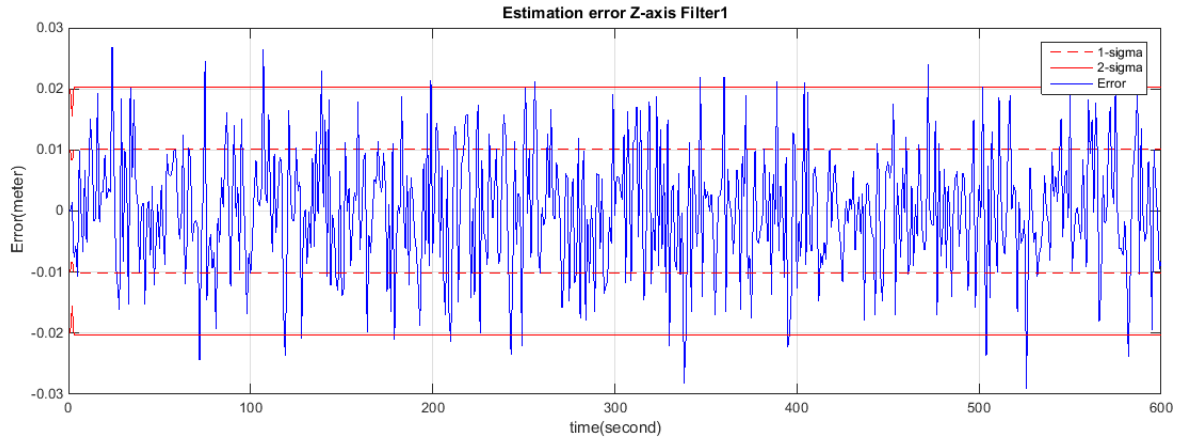


Figure 20: Estimation error in the z axis (heave). Both 1-sigma and 2-sigma boundaries are included.

3D plot of the true path and estimated path:

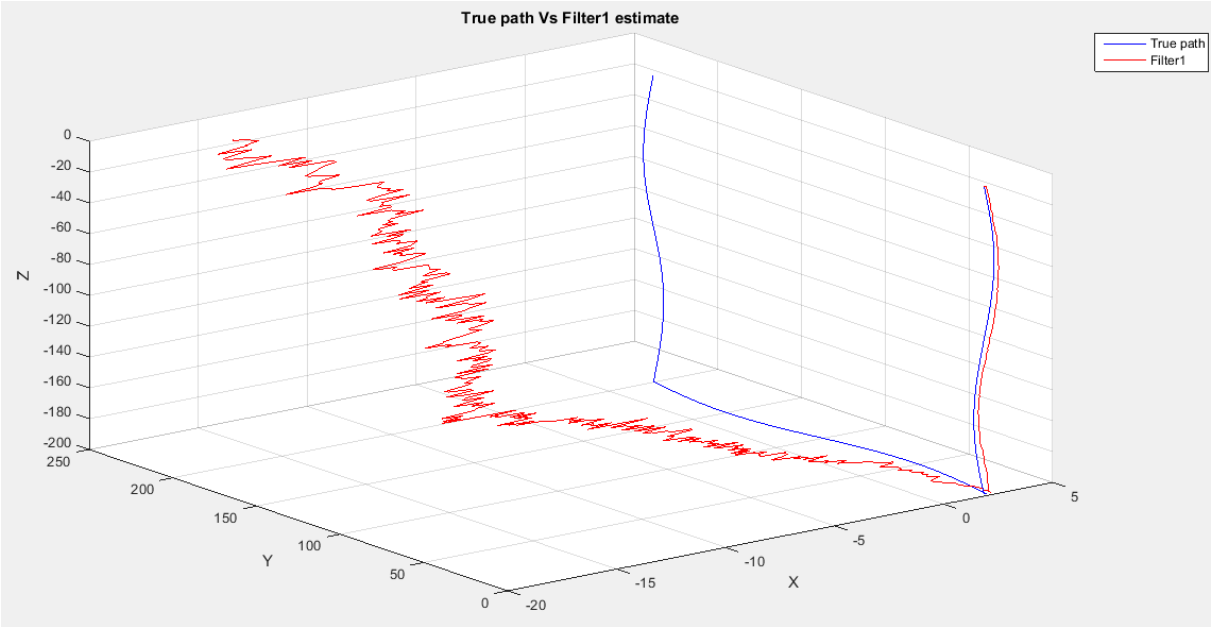


Figure 21: 3D illustration of the true path and the filter1 estimate.

Figures (18-21) show that Filter1 manages to track the ROV path on the way down until the ROV changes direction and starts moving parallel to the water surface. The estimation error in all axis stays within 2-sigma limits then drifts away in both x and y directions. The final estimated position is around 20 meters (in both directions x and y) far away from the ROV true final position. This is expected due to the numerical integration of DVL relative velocity components and the gyroscope angular rate.

5.3 Filter 2

Error in x direction:

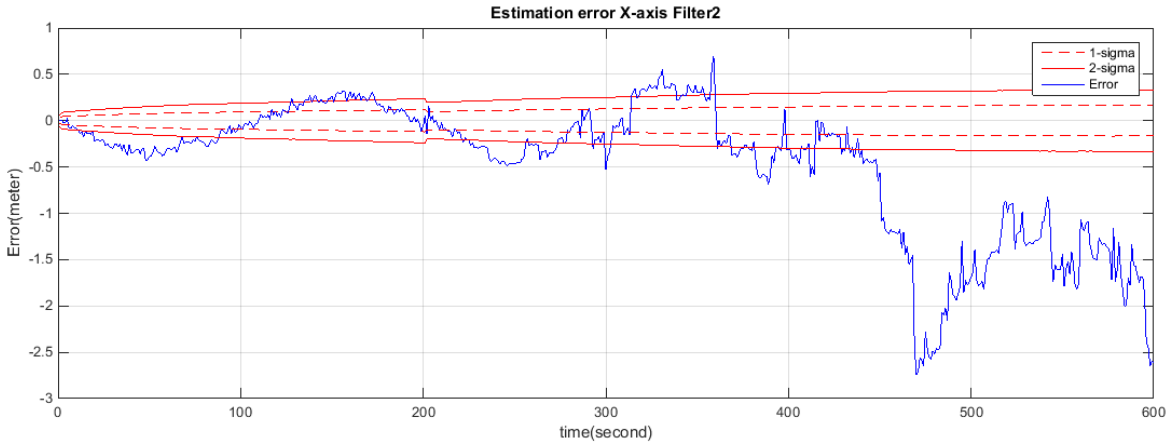


Figure 22: Estimation error in the x axis (surge). Both 1-sigma and 2-sigma boundaries are included.

Error in y direction:

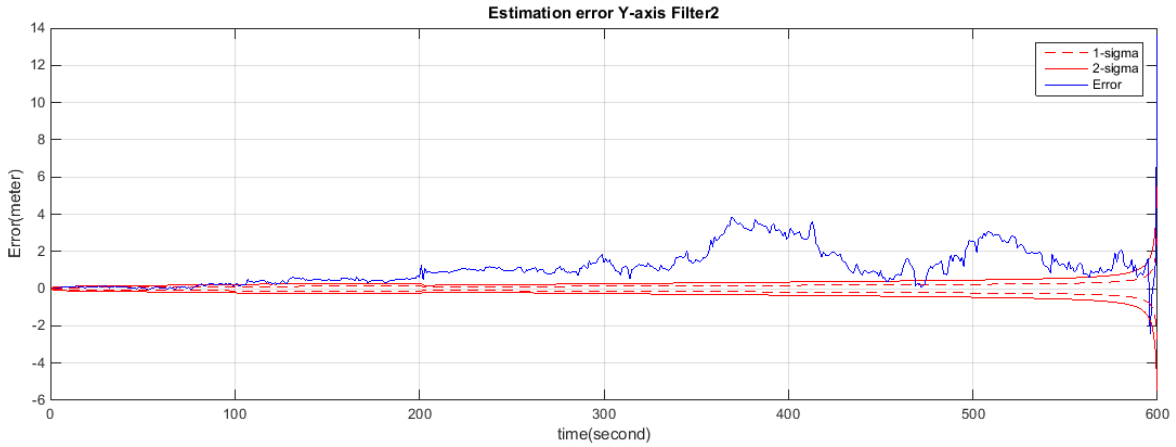


Figure 23: Estimation error in the y axis (sway). Both 1-sigma and 2-sigma boundaries are included.

Error in z direction:

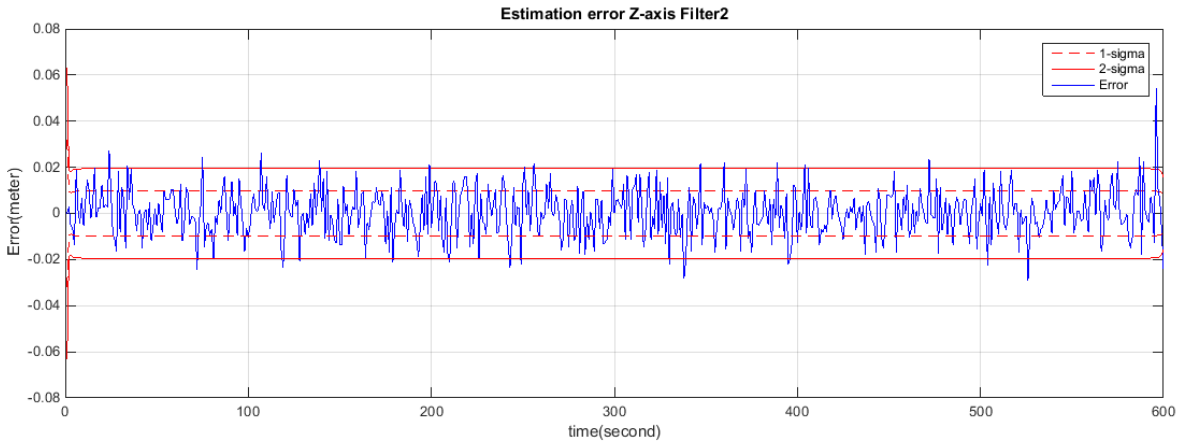


Figure 24: Estimation error in the z axis heave). Both 1-sigma and 2-sigma boundaries are included.

3D plot true path vs. estimated:

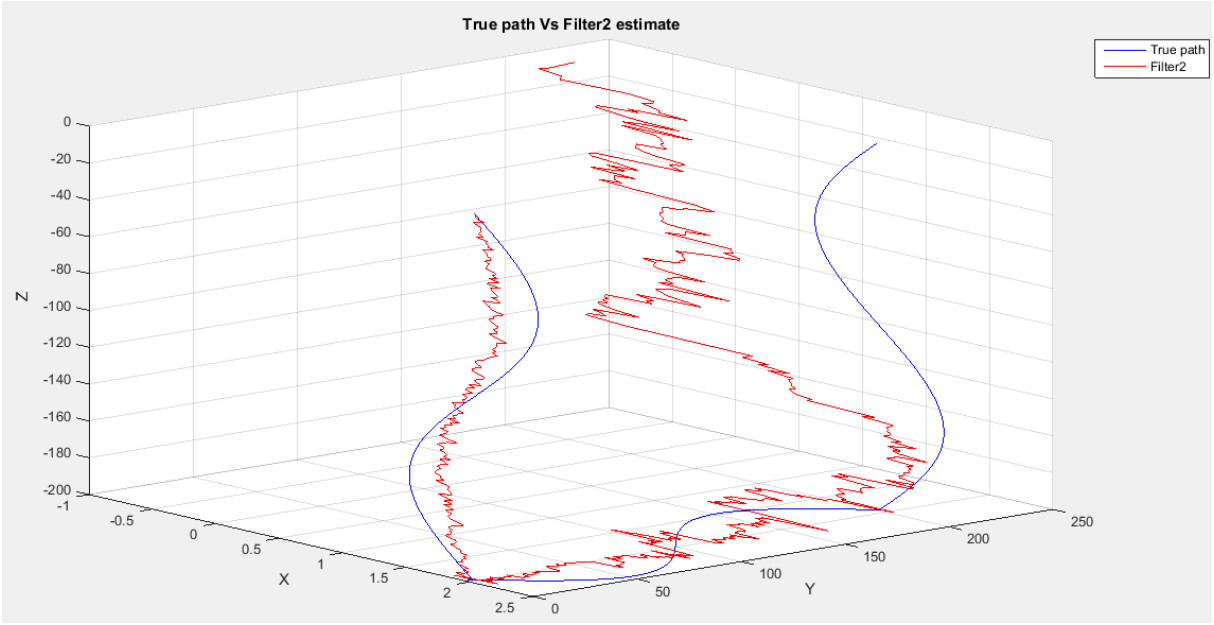


Figure 25: 3D illustration of the true path and the filter2 estimate.

Figures (22-25) show that Filter2 manages to track the ROV until the ROV changes direction and starts moving upwards to reach the water surface. The estimation error in all axis stays within 2-sigma limits then drifts away in both x and y directions. The final estimated position

is around 2-3 meters (in both directions x and y) far away from the ROV true final position. This is expected due to the numerical integration of DVL relative velocity components.

5.4 Filter 1 vs. filter 2

Error in x:

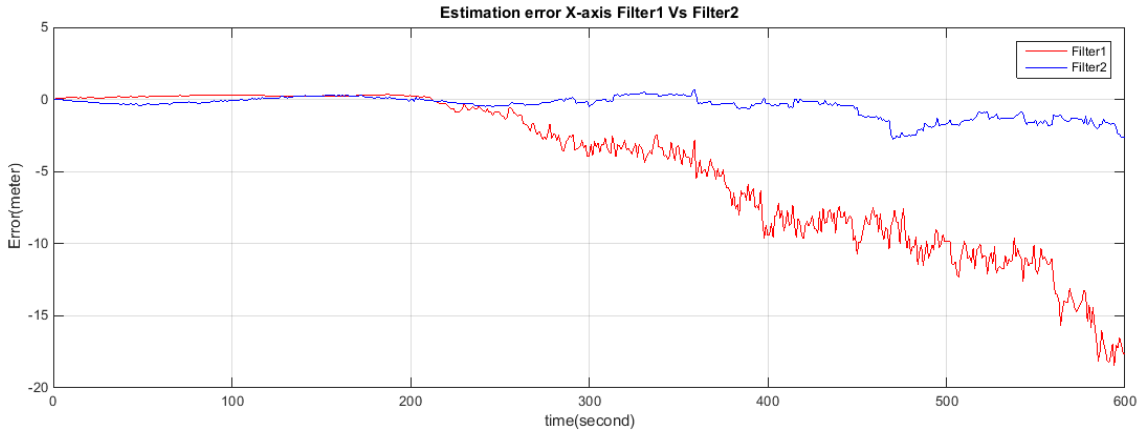


Figure 26: Estimation error comparison in the x axis (surge) between Filter1 and Filter2.

Error in y:

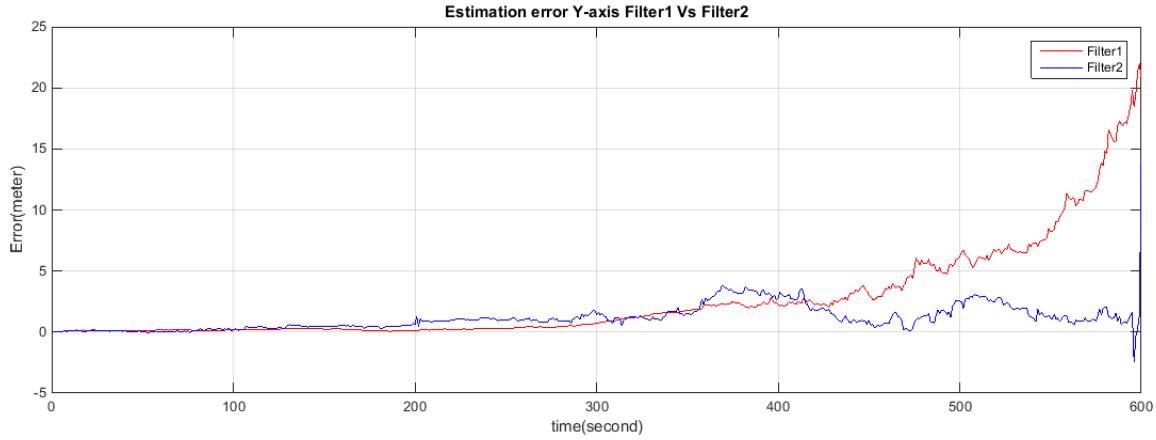


Figure 27: Estimation error comparison in the y axis (sway) between Filter1 and Filter2.

Error in z:

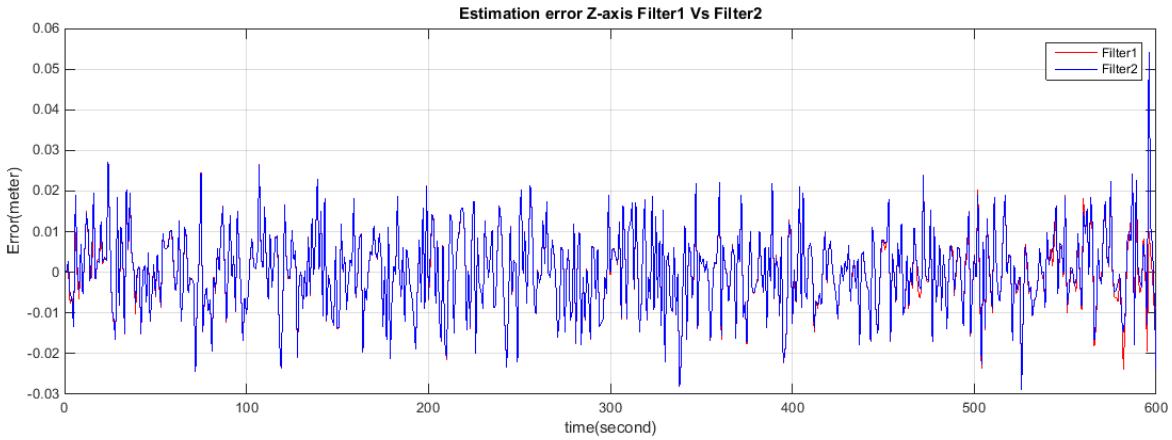


Figure 28: Estimation error comparison in the z axis (heave) between Filter1 and Filter2.

3D plot true path vs. estimated:

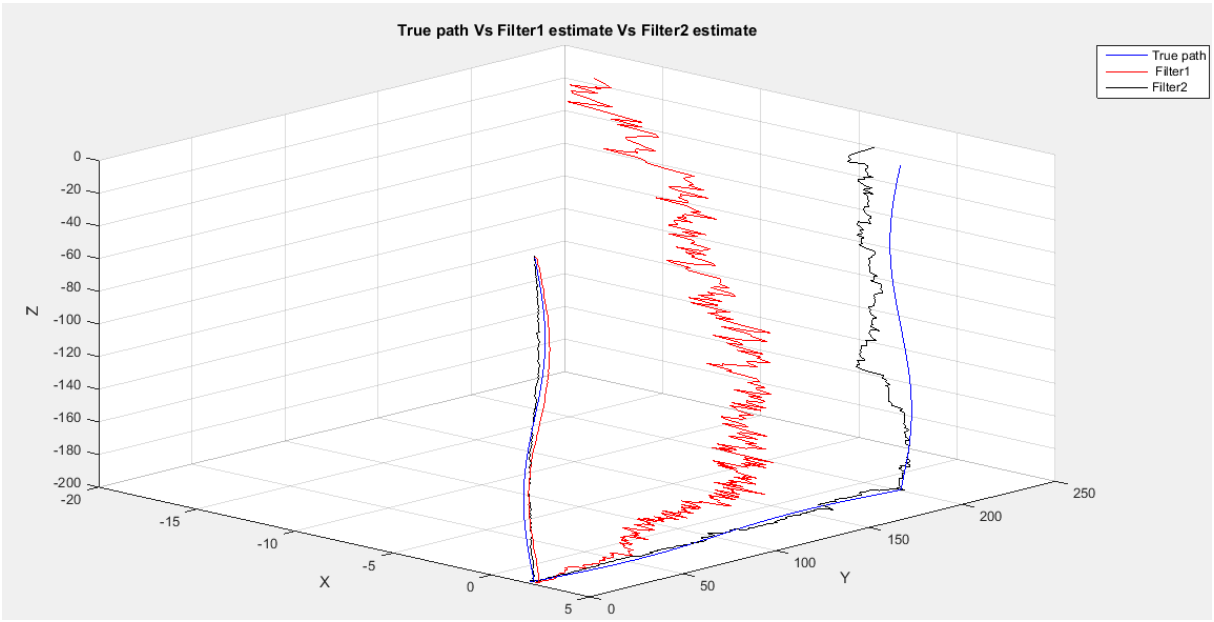


Figure 29: 3D illustration of both trajectory estimations for Filter1 and Filter2.

Adding the measurement of the angle α improves the performance of Filter2 in comparison to Filter 1 as seen in figures (26-29).The angle α reduces the drift caused by the numerical integration of the gyroscope angular rate. The x component of the position is dependent on the angle α measurements. Thus, its estimation benefits from this addition. The final estimated position using Filter2 is less than 2 meters (in both directions x and y) away from the true position, while the Filter1 final estimated position is around 20 meters away (in both direction x and y).

5.5 Filter3 sigma-set3

Error in x:

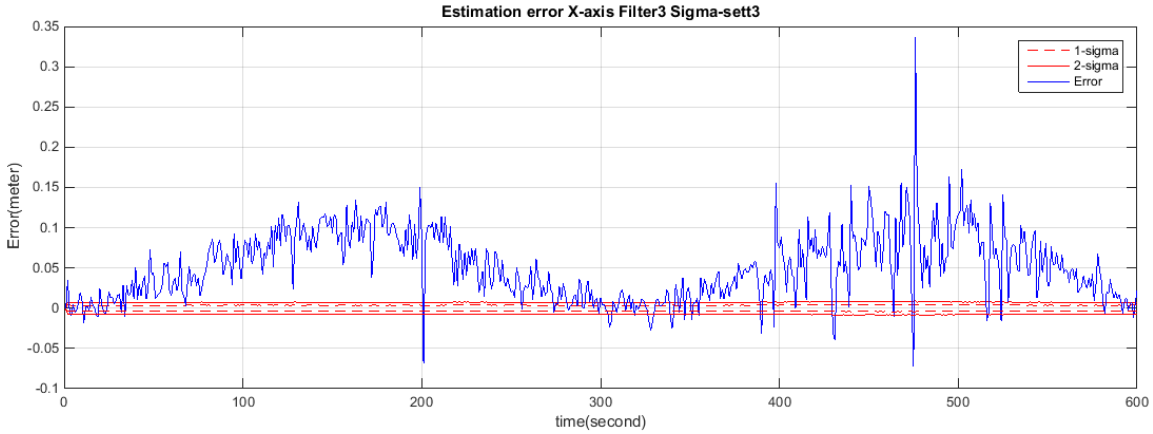


Figure 30: Estimation error in the x axis (surge). Both 1-sigma and 2-sigma boundaries are included.

Error in y:

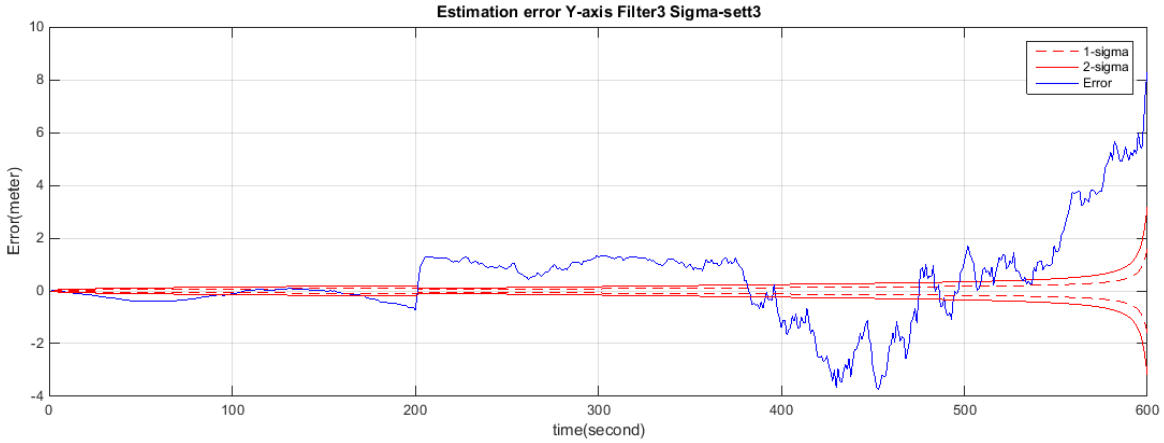


Figure 31: Estimation error in the y axis (sway). Both 1-sigma and 2-sigma boundaries are included**.**

Error in z:

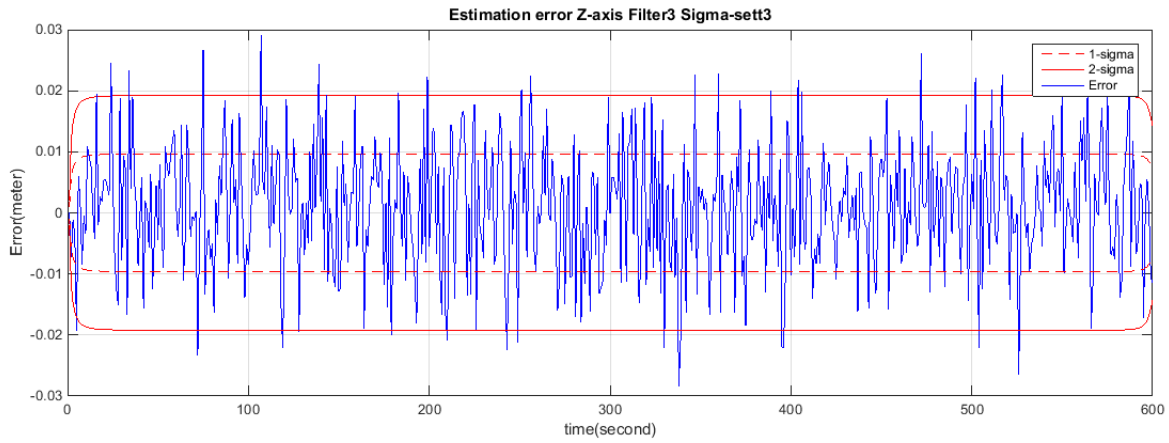


Figure 32: Estimation error in the z axis (heave). Both 1-sigma and 2-sigma boundaries are included.

3D plot true path vs. estimated:

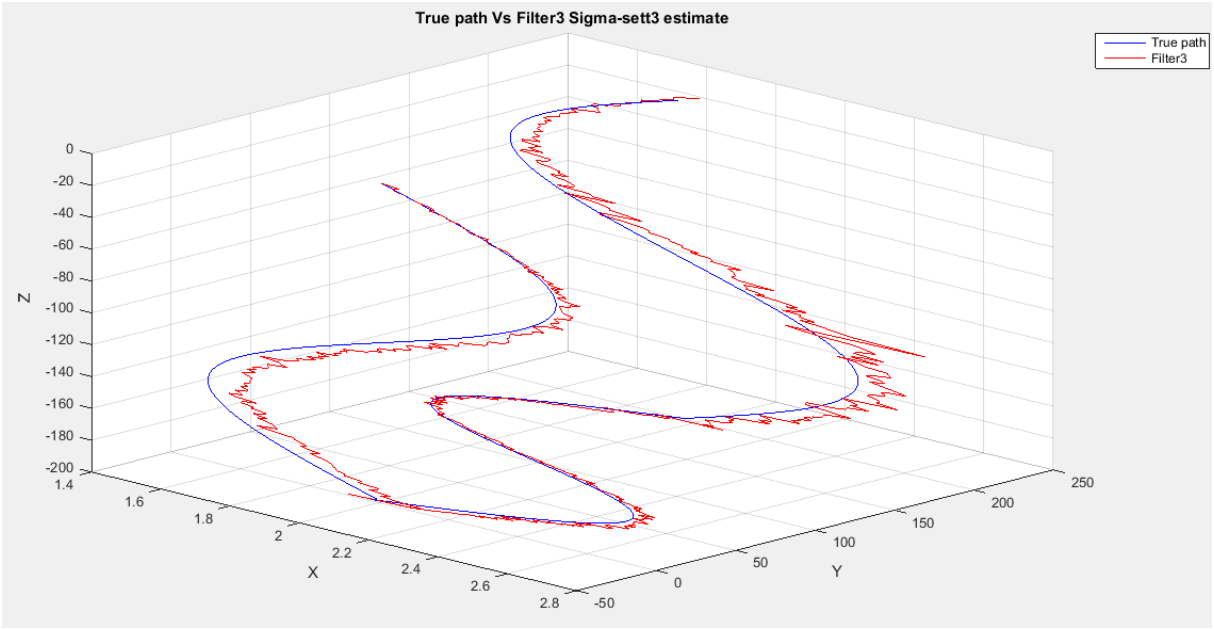


Figure 33: 3D illustration of the true path and the filter2 estimate.

Figures (30-33) show that Filter3 using Sigma-sett3 manages to track the ROV path from start to finish. The estimation of the x component benefits for the redundancy provided by the beam distances measurements. The estimation error in the x direction oscillates in and out of the 2-sigma limits, but doesn't exceed 0.3 meters during the full sequence. The estimation error in y direction is less than 1meter for the first two thirds of the sequence, and then drifts

away. This is expected due to the numerical integration of DVL relative velocity used to estimate the y component.

The final estimated position is pretty accurate for the x component and around 8 meters far away from the ROV true final position.

5.6 Filter3 sigma set1

Error in x:

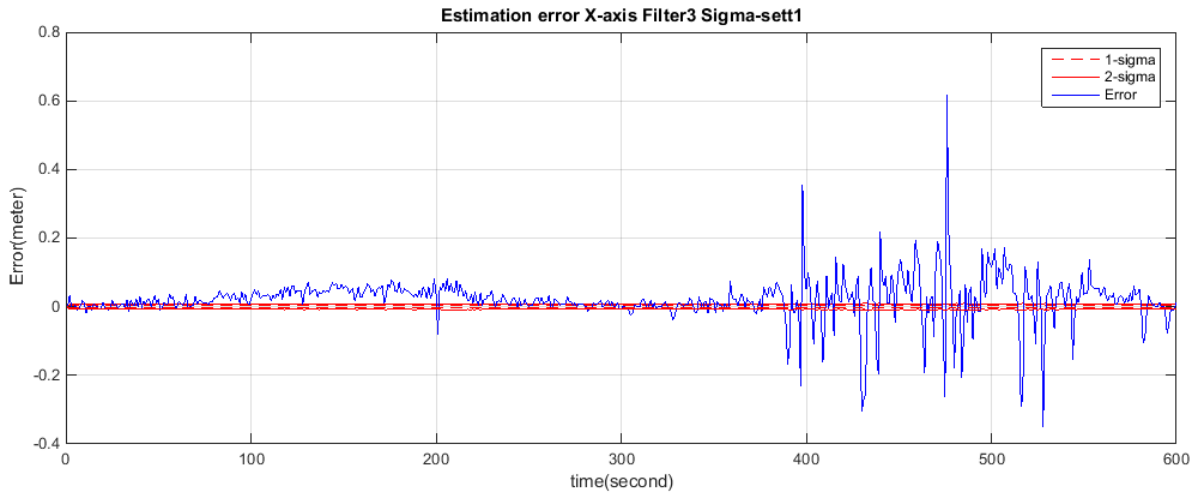


Figure 34: Estimation error in the x axis (surge). Both 1-sigma and 2-sigma boundaries are included.

Error in y:

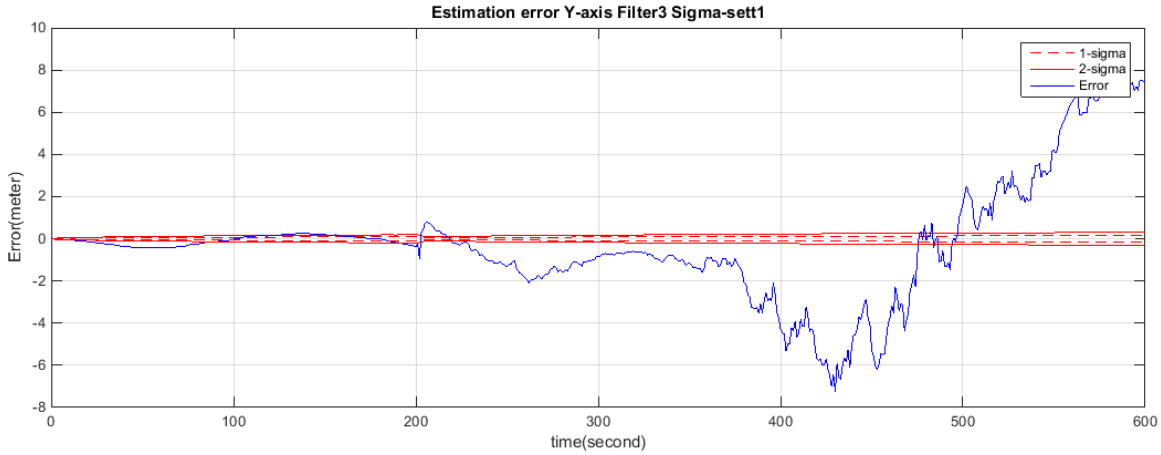


Figure 35: Estimation error in the y axis (sway). Both 1-sigma and 2-sigma boundaries are included.

Error in z:

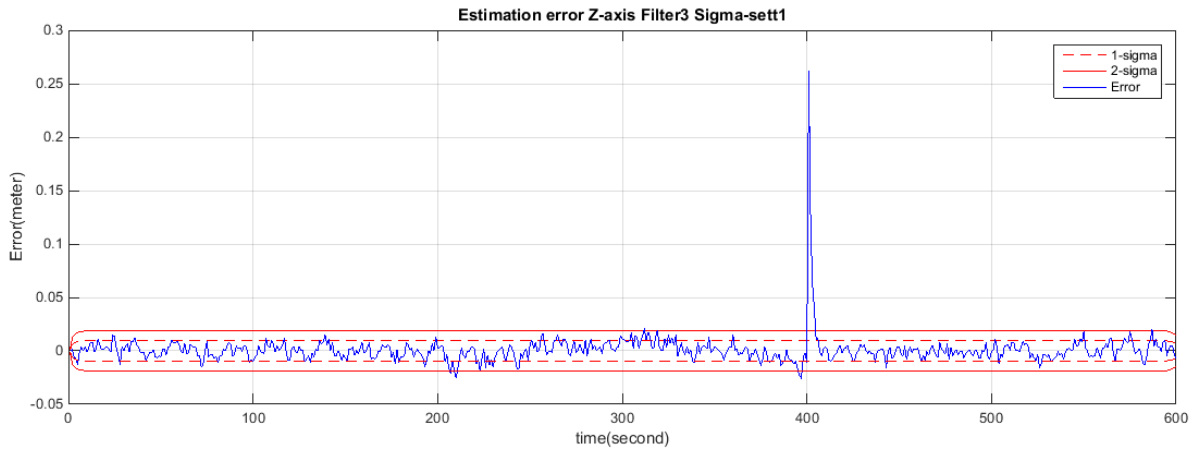


Figure 36: Estimation error in the z axis (heave). Both 1-sigma and 2-sigma boundaries are included

3D plot true path vs. estimated:

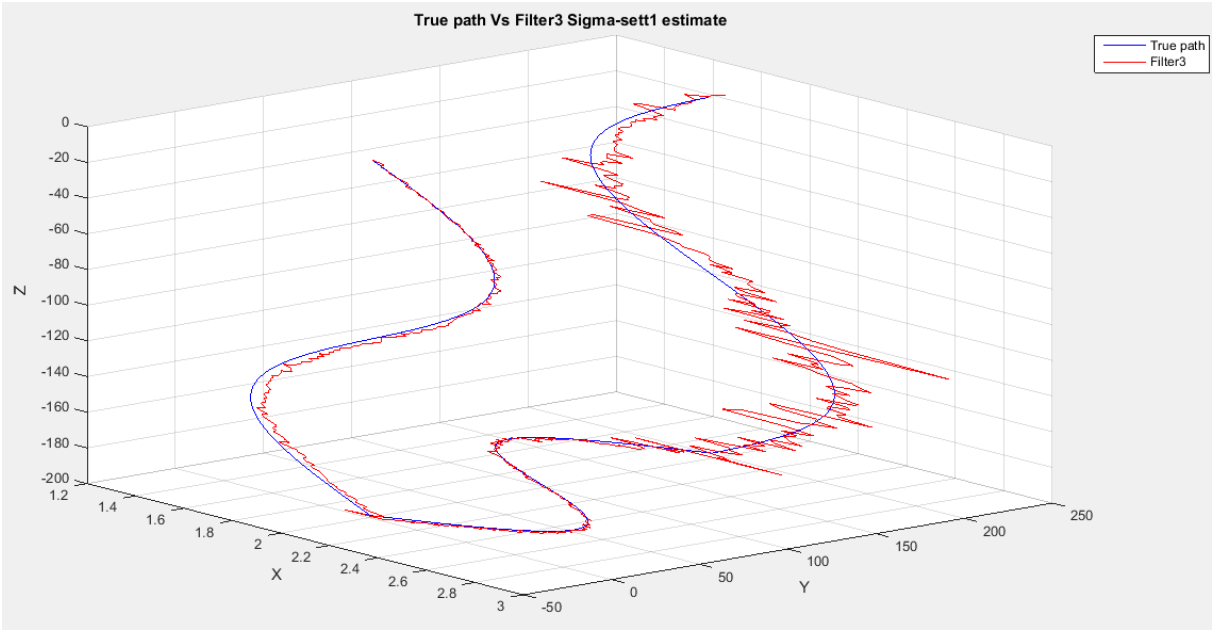


Figure 37: 3D illustration of the true path and the filter2 estimate.

Figures (34-37) show that Filter3 using sigma-set1 manages to track the ROV path from start to finish. The estimation error in the x direction oscillates in and out of the 2-sigma limits, but doesn't exceed 0.6 meters during the full sequence. The estimation error in y direction is less than 2 meters for the first two thirds of the sequence, and then drifts away. This is expected due to the numerical integration of DVL relative velocity used to estimate the y component.

The final estimated position is pretty accurate for the x component and around 8 meters far away from the ROV true final position.

5.7 Filter 3 sigma set3 vs. sigma set1

Error in x:

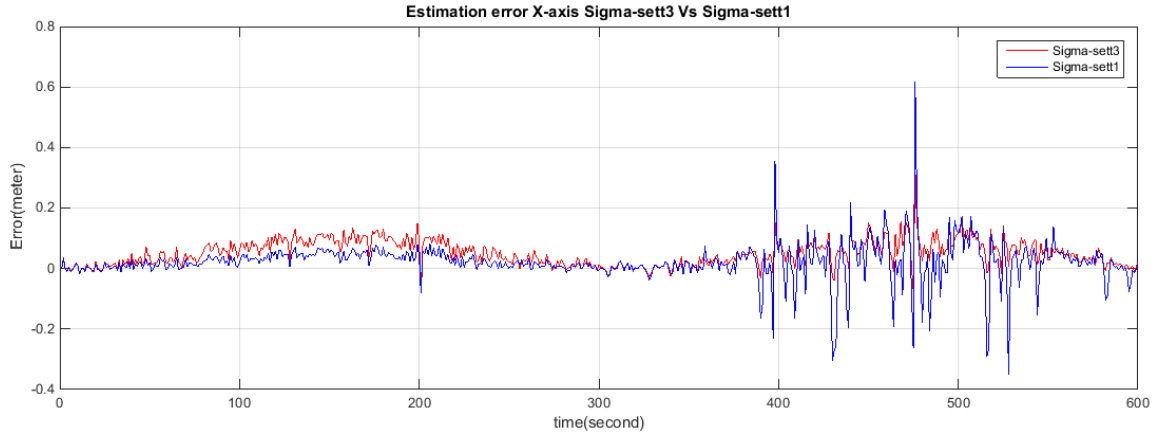


Figure 38: Estimation error comparison in the x axis (surge) between Filter 3 sigma set1 vs. sigma set3

Error in y:

Figure 39: Estimation error comparison in the y axis (sway) between Filter 3 sigma set1 vs. sigma set3

Error in z:

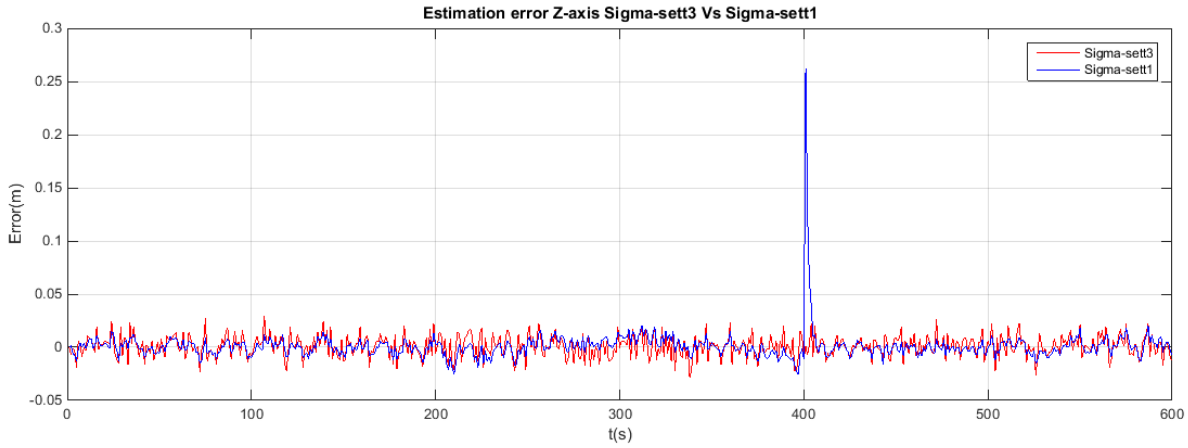


Figure 40: Estimation error comparison in the z axis (heave) between Filter 3 sigma set1 vs. sigma set3

3D plot true path vs. estimated:

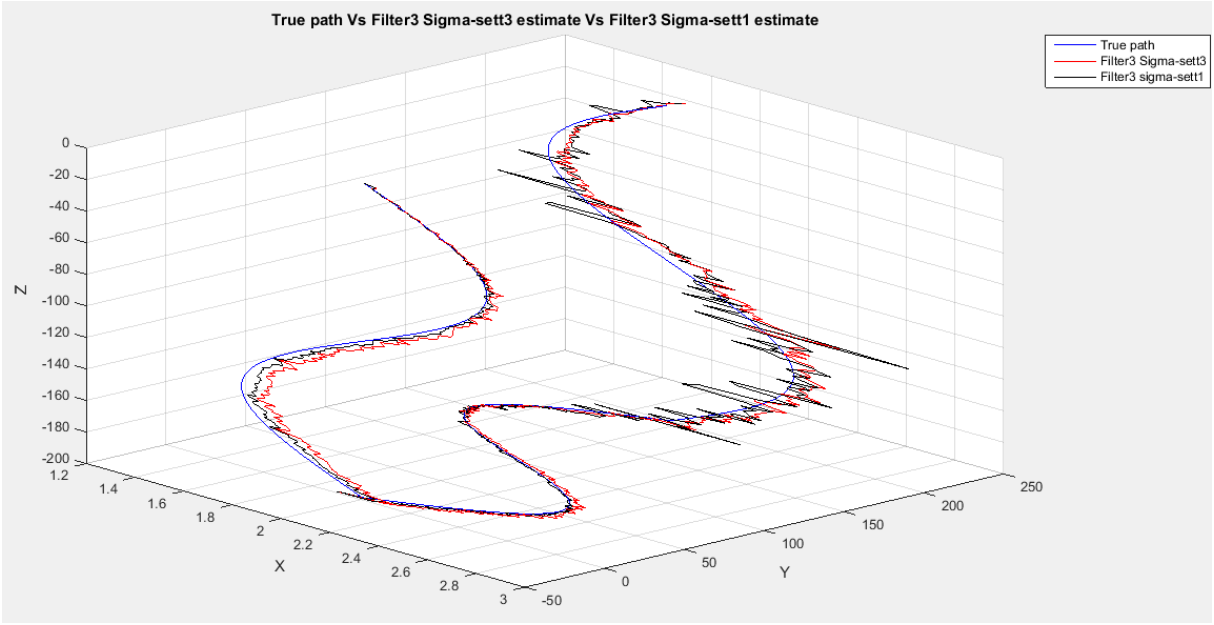


Figure 41: 3D illustration of both trajectory estimations for Filter 3 sigma set1 vs. sigma set3

Figures (38-41) show a comparison between the two sigma sets results, both perform pretty close. The sigma-sett3 does a slightly better job in the second half of the sequence while the

sigma-sett1 performs slightly better in the first half. The sigma-sett 3 is picked to represent Filter3 for other comparisons.

5.8 Filter 3 sigma set3 vs. filter 2

Error in x:

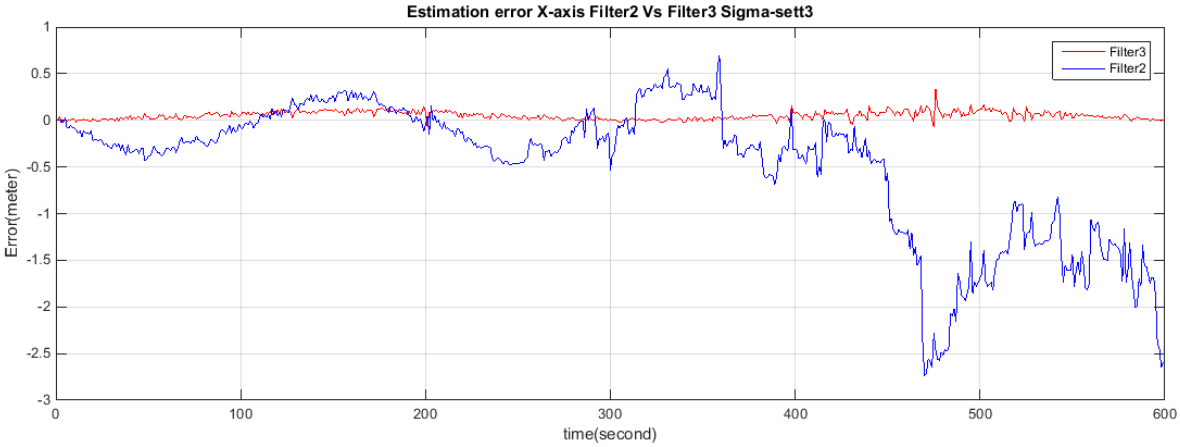


Figure 42: Estimation error comparison in the x axis (surge) between Filter3 and Filter2.

Error in y:

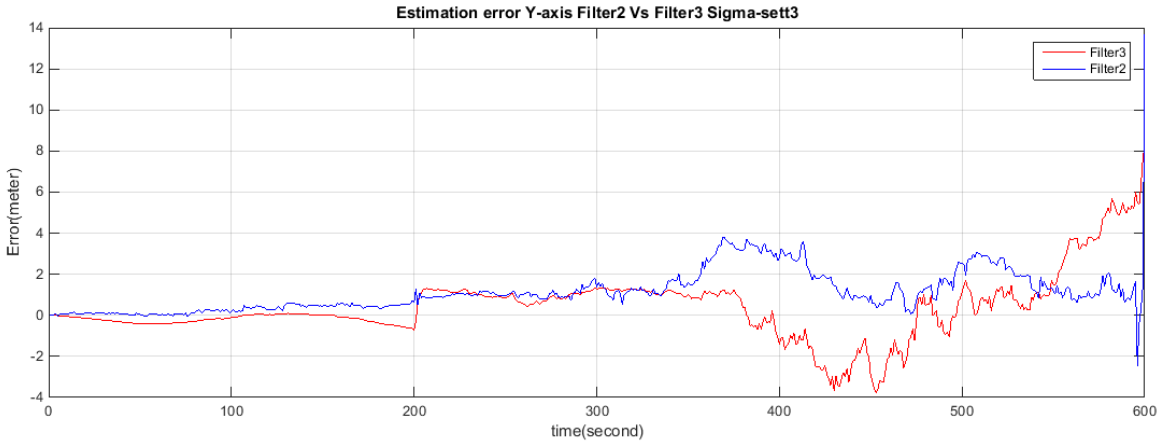


Figure 43: Estimation error comparison in the y axis (sway) between Filter3 and Filter2.

Error in z:

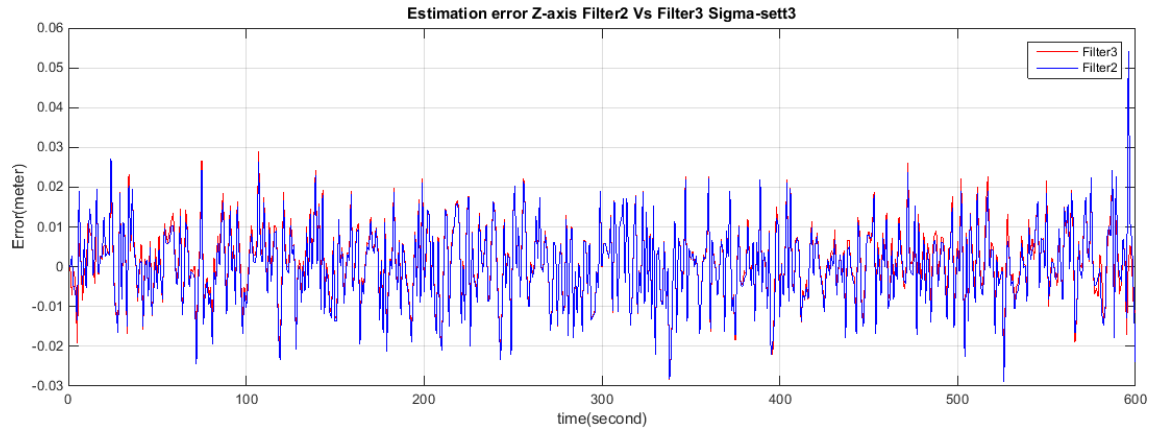


Figure 44: Estimation error comparison in the z axis (heave) between Filter3 and Filter2.

3D plot true path vs. estimated:

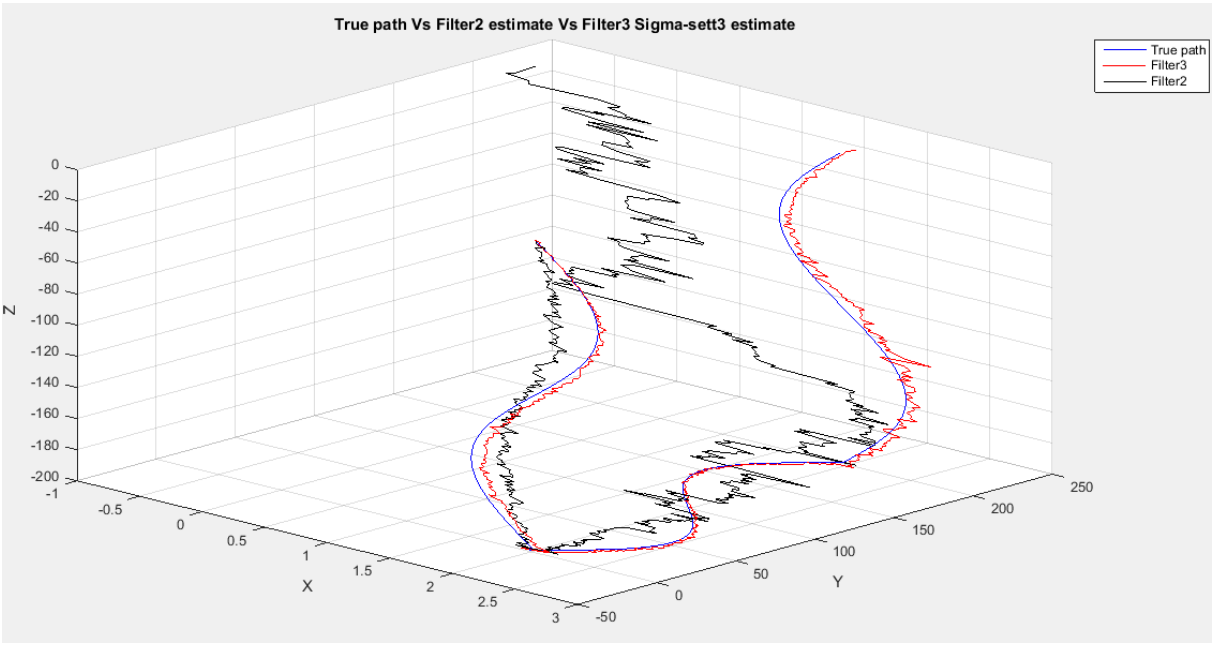


Figure 45: 3D illustration of both trajectory estimations for Filter3 and Filter2

Adding the measurements [b_1, b_2, b_3, b_4] from the beams improves the overall performance of Filter3 in comparison to filter 2 as seen in figures (42-45). The x component of the estimated position is dependent on the beam distances and benefits from the added measurements. There

is a clear improvement in the x component estimate. As shown in figure (42) the drift caused by integrating the DVL relative velocity in Filter2 is gone.

The drift in the y component is still present in both filters mainly due to the numerical integration used for estimation.

5.9 Filter 4

Error in x:

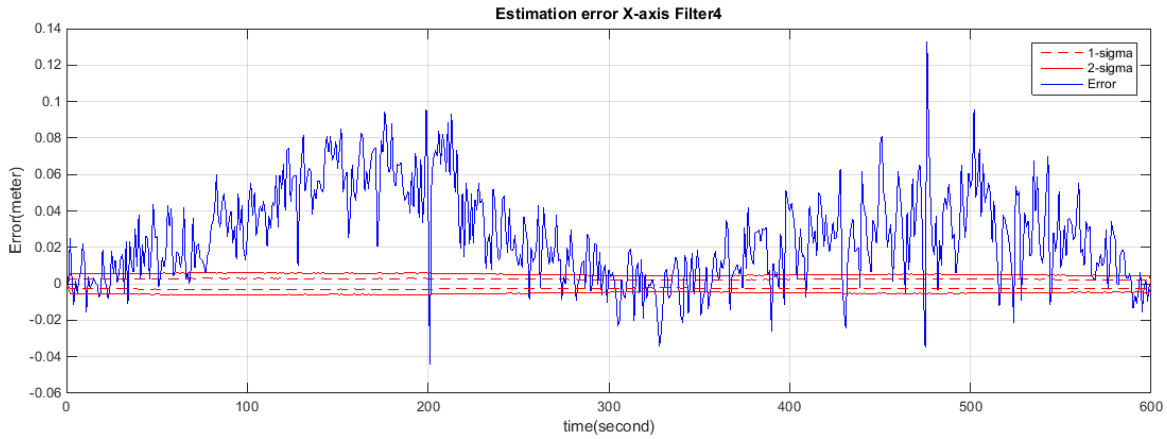


Figure 46: Estimation error in the x axis (surge). Both 1-sigma and 2-sigma boundaries are included.

Error in y:

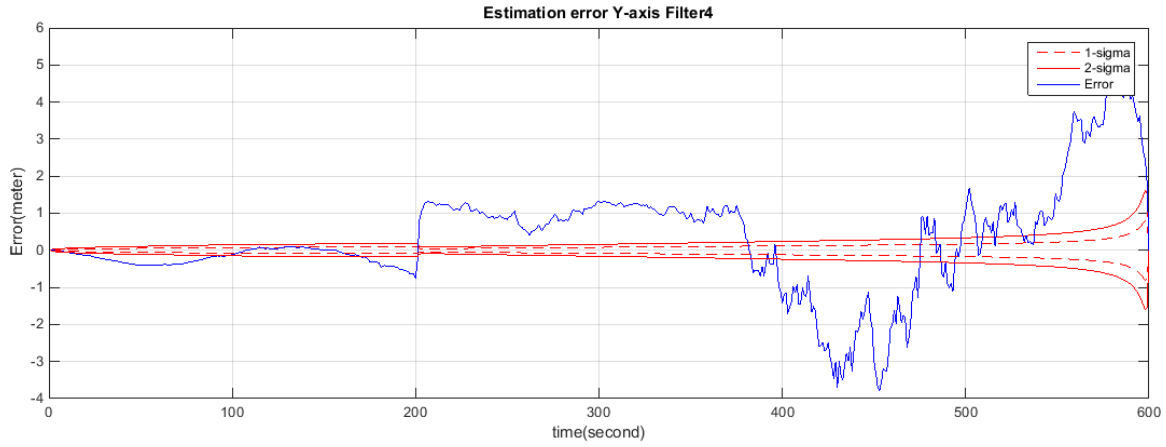


Figure 47: Estimation error in the y axis (sway). Both 1-sigma and 2-sigma boundaries are included.

Error in z:

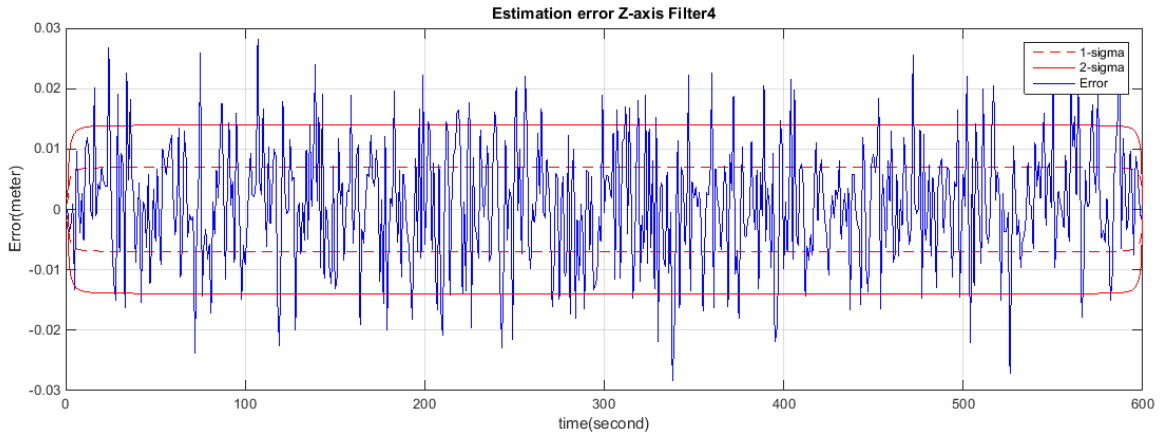


Figure 48: Estimation error in the z axis (heave). Both 1-sigma and 2-sigma boundaries are included

3D plot true path vs. estimated:

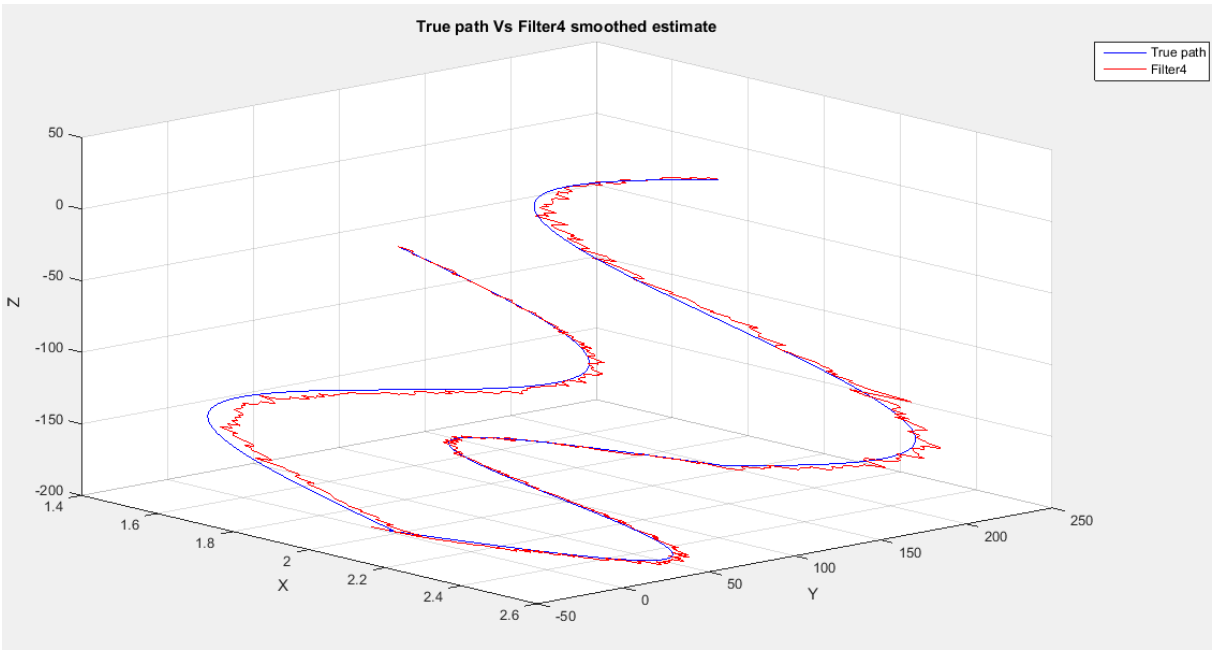


Figure 49: 3D illustration of the true path and Filter 4 estimate

Figures (46-49) show that Filter4 manages to track the ROV path from start to finish. The estimation error in the x direction oscillates in and out of the 2-sigma limits, but doesn't exceed 0.13 meters during the full sequence. The estimation error in y direction is less than 1 meter for the first two thirds of the sequence, and then drifts away. This is expected due to the numerical integration of DVL relative velocity used to estimate the y component. The backwards filter makes it possible to use the GPS data for the final position and improves the final position estimate.

5.10 Filter 3 sigma set3 vs. Filter 4

Error in x:

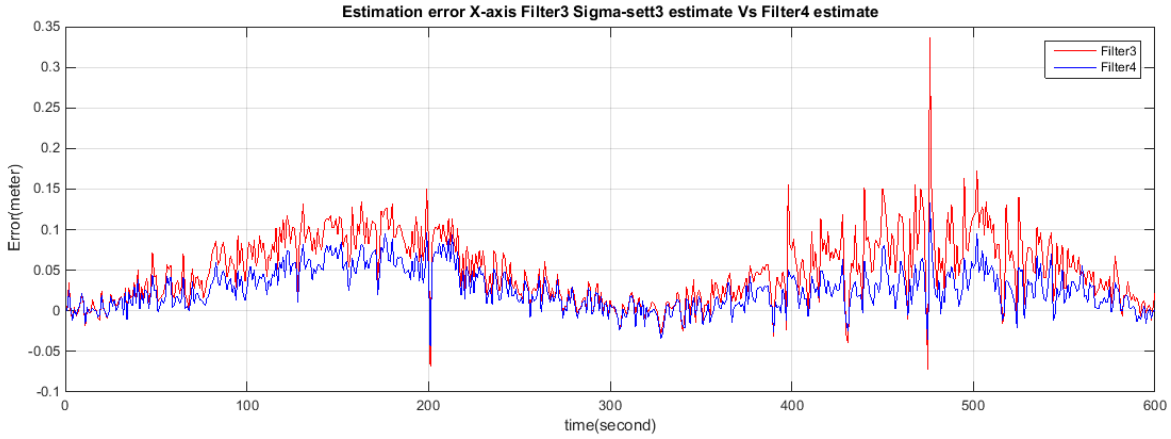


Figure 50: Estimation error comparison in the x axis (surge) between Filte3 and Filter4.

Error in y:

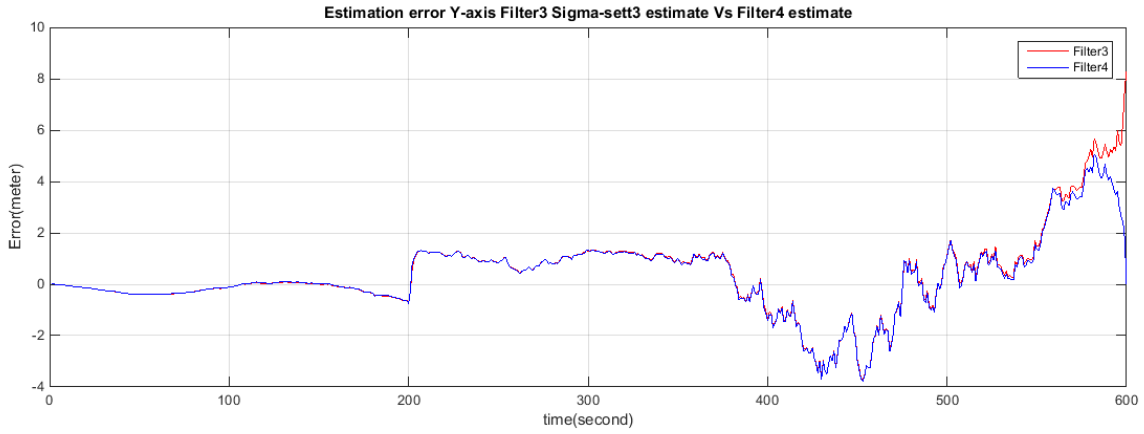


Figure 51: Estimation error comparison in the y axis (sway) between Filte3 and Filter4.

Error in z:

Figure 52: Estimation error comparison in the z axis (heave) between Filter4 and Filter3 using Sigma-set3.

3D plot true path vs. estimated:

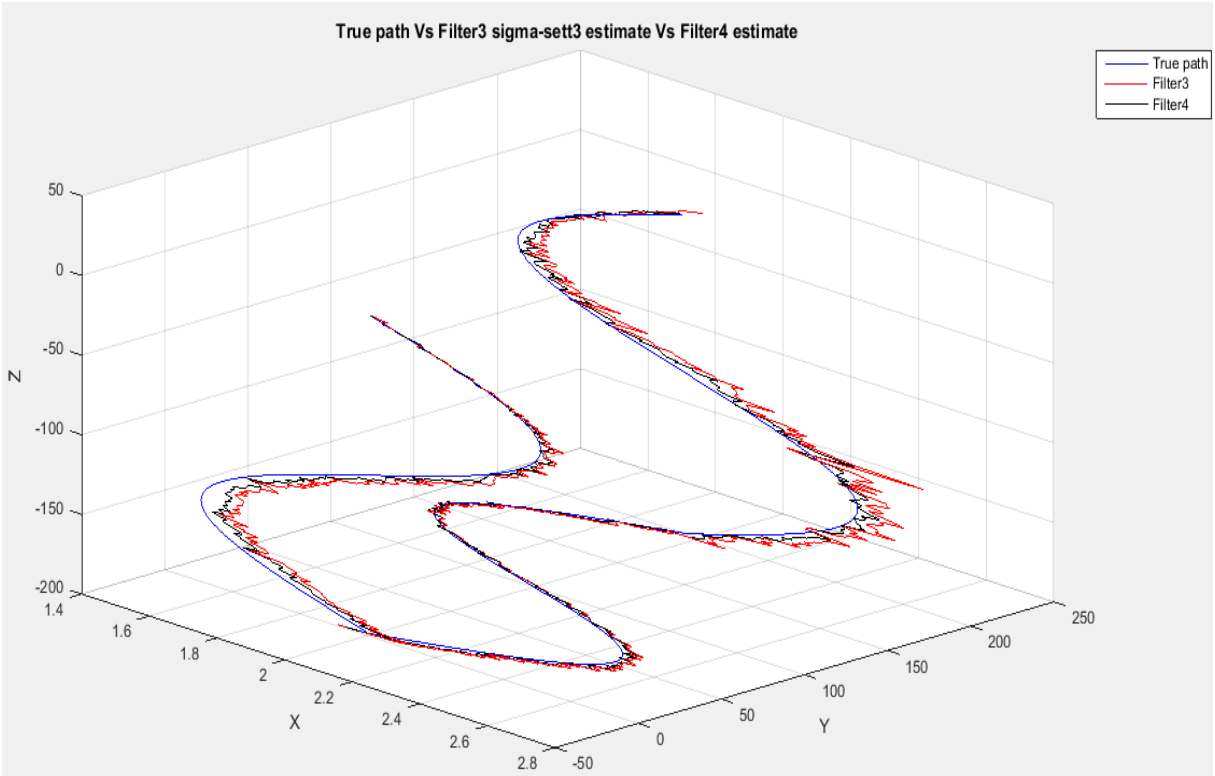


Figure 53: 3D illustration of both trajectory estimations for Filte4 and Filter3 using sigma-set3.

Figures (50-53) shows that Filter4 has a better overall performance than Filter 3. The x component of the estimated position benefits slightly from the backward run. The y component of the estimated position has similar performance for the first 90% of the sequence, and then Filter 4 converges to the true final position while filter 3 keeps drifting.

5.11 Summary

The smoother UKF filter Filter 4 gives the best estimation of the ROV position during the sequence of motion.

The design procedure and results can be summed up in figure (54).

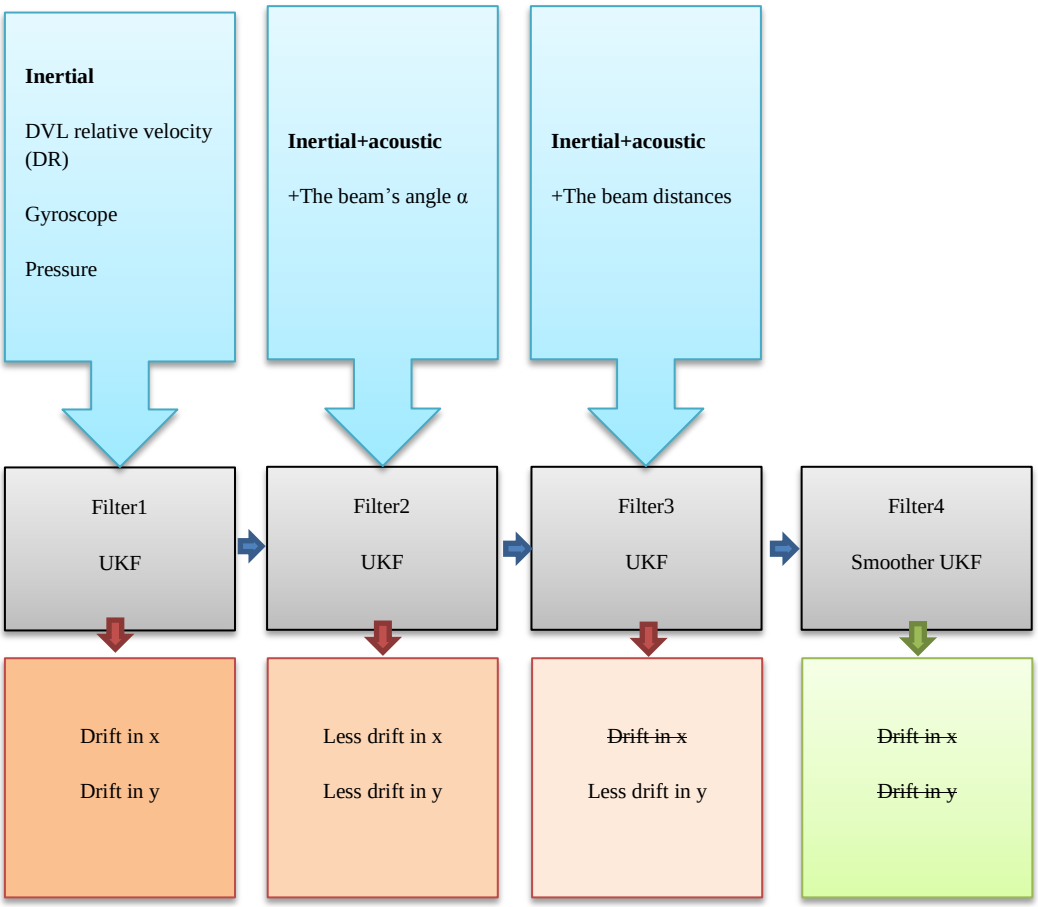


Figure 54: Summary of the design process of the Filter 4, and sensor fusion levels with results

6 Conclusion and further work

6.1 Conclusion

The aim of this project was to investigate the abilities to track the motion of a ROV relative to a net pen, using a DVL system, a single axis gyroscope, a GPS and a pressure sensor. The literature review of current methods used for this purpose indicates that they suffer from several challenges, but can offer good accuracy when implemented in an adequate environment. Whereas equipping the ROV with the sensor configuration available for this assignment will not only save money and time, but more importantly extend the reach of underwater localization. The proposed smoother unscented Kalman filter algorithm (Filter 4) manages to track the ROVs travelled path within acceptable margins.

The design of the filter was a result of a series of different filter implementations and different sensor combinations, tested in a simulated environment. The conception of the filter was done by gradual sensor fusion and identification of weaknesses and improvements at each stage.

The first implemented filter was based on the unscented Kalman filter using only the gyroscope, the pressure sensor and the DVL relative velocity. The second filter adds the measurement of the angle α. The third filter adds the beam distances. The last filter utilizes both past and future information and delivers the best estimate of the traveled path. The expected performance improvement at each level of sensor fusion was met.

While both inertial and acoustic methods have each their own downsides, they work perfectly together to overcome each other weaknesses.

6.2 Further suggested work

In the following some ideas for further work are proposed.

A more precise and accurate model of the ROV dynamics and sensors

The linear process filter model used in this assignment is a poor approximation of the actual dynamics model. In order to get a better idea about the performance of the algorithm in a real situation, a better modeling can be done including all the environmental factors that affect the motion underwater. The same goes for the sensor models.

Outlier identification

Because of the biomass interaction during net inspection operations, measurements cannot always be trusted. There are many techniques and research invested in outlier detection. This can be beneficial for underwater operations and trajectory estimation in general.

Visual odometry

The ROV is equipped with a camera used by the pilot for navigation. Using visual odometry may improve the overall performance and the robustness. The correlation between performance improvement and sensor fusion levels witnessed in this thesis, predicts further enhancement when more measurements are available.

Implementing a correction system

The net can be equipped with a set of landmarks that the ROV can use as references. The basic idea here is to use the camera to identify these markers and correct or update the position estimate and relevant measurements with the known markers position.

Aquaculture industry is interested in taking advantage of the progress made in the underwater ROV localization industry, but most of the challenges that motivated the research, from sensor choice to estimation algorithms, are usually oriented towards improving navigation in a relatively unknown environment. In aquaculture, net pens and fish cages are designed with a lot of flexibility and control over the shape, materials etc. Manufacturing a suitable net pen that can be equipped with landmarks or even a grid might give better results and can reduce

the cost of such ROV setups. Visibility in underwater environment poses a problem, but it is still a field to investigate.

7 References

Corke, Peter. *Robotics vision and control.* Fundamental algorithms in Matlab. Appendix H. (2013)

Hallingstad, Oddvar. *Analyse av suboptimale KF, UNIK4500 Stokastiske systemer.* (2009)

Hallingstad, Oddvar. *Eksempler på TNS-modeller, UNIK4540 Matematisk Modellering Av Dynamiske Systemer.* (2004)

Hallingstad, Oddvar. *Monte Carlo- og Kovariansanalyse av Kalmanfilteret, UNIK4500 Stokastiske Systemer*.(2009)

Hallingstad, Oddvar. *Notat, UNIK 4680 Anvendt Parameter og Tilstandsestimering*. (2005)

Hallingstad, Oddvar. *Notat, UNIK4540 Matematisk Modellering Av Dynamiske Systemer*. (2014)

Hallingstad, Oddvar. *Sammendrag av modeller og ligninger for Kalmanfilteret anvendt på et ulineært system, UNIK4500 Stokastiske Systemer*. (2009)

Jensen, Ø., Dempster, T., Thorstad, E., Uglem, I., Fredheim, A. *Escapes of fishes from Norwegian sea-cage aquaculture, consequences and prevention*. Aquacult Environ Interact. (2010).

Julier, S.J., Uhlman, J.K. *Unscented Filtering and Nonlinear Estimation*. Proceedings of the IEEE, Vol. 92, No. 3. (2004).

Paull, L., Saeedi, S., Seto, M., Li, H. *AUV navigation and localization: a review.* IEEE J. Oceanic Eng., 39. (2014)

Rundtop, Per, Frank, Kevin. *Experimental evaluation of hydroacoustic instruments for ROV navigation along aquaculture net pens.* Aquacultural Engineering. Volume 74. (2016). Link: http://www.sciencedirect.com/science/article/pii/S0144860916300012#bbib0015

Spong, Mark W., Hutchinson, Seth, Vidyasagar, M. *Robot Modeling and Control*. (2006)

Vaganay, J., Leonard, J., Curcio, J., Willcox, J. *Experimental validation of the moving long base-line navigation concept*. in Proc. IEEE/OES Autonom. Underwater Veh. Conf. (2004) Link: https://marinerobotics.mit.edu/sites/default/files/Vaganay04auv.pdf

Wan, A., Eric, Van der Merwe, Rudolph. *The unscented Kalman filter*. Kalman Filtering and Neural Networks, edited by Simon Haykin, Chapter 7. (2010)

Fiskeridirektoratet. *Kategorisering av rømmingshendelser i 2016*. (2017) Link: https://www.fiskeridir.no/Akvakultur/Statistikk-akvakultur/Roemmingsstatistikk

Teledyne RD Instruments. *Acoustic Doppler Current Profiler Principles of Operation: A Practical Primer*. (2011). Link: http://www.comm-tec.com/Docs/Manuali/RDI/BBPRIME.pdf

8 Appendices

Appendix A – Main

Appendix B – Gyroscope sensor model

Appendix C – Pressure sensor model

Appendix D – DVL sensor model

Appendix E – Filter1

Appendix F – Filter1: Linear Kalman filter

Appendix G – Filter2

Appendix H – Filter3

Appendix I – Filter4: Backward run

Appendix J – Plot of all results

8.1 MATLAB script instructions

In the following the applied MATLAB scripts are presented. Running Main will generate all the needed data and apply filter functions. Appendix J contains the master plot. Running it will produce all the relevant results from all filters and filter comparisons. All figures are titled to make it easy to find the right figure among the total of 36 figures.

8.2 Appendix A – Main

```
clear all;
%INITIAL CONDITIONS
% n is the number of measurement steps and travelled distance
n=600;
%frequency, time increment
dt=1;
%the input from the ROV pilot
U=[0 ; 0 ; -1];
%simulated yaw angle.
%initial yaw angle is 0
gyroyaw=zeros(1,n);
gyroyaw(1)=0;
w=0.01;
for i=1:dt:n
gyroyaw(i)=0.3*sin(w*i);
end

%True path generator using a sinx.
sysposition=zeros(3,n);
% k1 is the first turn in point and k2 is the going up point.
k1=200;
k2=400;
T=k1;
T2=k2;
w1=1/T;
%initial position of the ROV is M
M=[2 2 0]';
F(:,1)=M;
sysposition(:,1)=M;
for i=1:k1-1
    F(1,i+1)=0.5*sin(2*w1*pi*(i))+M(1,1);
    F(3,i+1)=-i;
    F(2,i+1)=0.5*sin(2*w1*pi*(i))+M(2,1);
    sysposition(:,i+1)=F(:,i+1);
end
F1(:,1)=F(:,k1);
for i=k1:k2-1
    F(1,i+1)=0.5*sin(2*w1*pi*(i))+F1(1,1);
    F(3,i+1)=0.5*sin(2*w1*pi*(i))+F1(3,1);
    F(2,i+1)=-(k1-i);
    sysposition(:,i+1)=F(:,i+1);
end
F3(:,1)=F(:,k2);
for i=k2:n-1
    F(1,i+1)=0.5*sin(2*w1*pi*(i))+F3(1,1);
    F(3,i+1)=F3(3,1)-(-i+k2);
    F(2,i+1)=0.5*sin(2*w1*pi*(i))+F3(2,1);
    sysposition(:,i+1)=F(:,i+1);
end

%Getting measurements using the sysposition data (true path).
%pressure
zmeas=trykk(sysposition(3,:),n);
%gyroscope angular rate
gyrovelo=gyro1(gyroyaw,n);
%generate measurement from dVL (Vreal is the relative velocity/alphas is
the angle alpha/b1,..,b4 the beam distances)
```

```
[Vreal,alphas,b1,b2,b3,b4]=DVLread(gyroyaw,sysposition,n,gyrovelo,U);


%write generated measurements to the measurement vector
[z,Vreal,gyrovelo,b1,b2,b3,b4]
measposition1=zeros(9,n);
measposition1(1,1)=M(3,1);
measposition1(6:9,1)=[b1(1) b2(1) b3(1) b4(1)]';
measposition1(2:4,1)=Vreal(:,1);
measposition1(5,1)=0;
 for i=2:n

     measposition1(1,i)=zmeas(i);
     measposition1(6:9,i)=[b1(i) b2(i) b3(i) b4(i)]';
     measposition1(2:4,i)=Vreal(:,i);
     measposition1(5,i)=gyrovelo(1,i);

 end

%writing to Filter2 measurement set (No beam distances; adds angle alpha)
measposition12=[measposition1(1:5,:); alphas];
%writing to Filter3
measposition13=[measposition1; alphas];



%Filer1
[estimpositionDVL1,Pdvlgyro]=Filter1(M,U,measposition1,n);
%Filter2
[estimpositionDVL,Pdvlalpha]=Filter2(M,U,measposition12,n);
%Flter3 both sigma sets. (estimposition is the estimated position using
sigma-set3)
[estimposition,sigmasett1,Psmooth,Psmooths1]=Filter3(M,U,measposition13,n);


%Filter 4 is a combination of the backward run and Filter3 using sigma-set3
%Backwards
[estimpositionbb,Pback]=BackKF(measposition1,n,sysposition);
%The smoother UKf
%state variable is 5x1 vector
n_xx=5;
%estimpostionsmoothed is the estimate from filter4, PF4 is the covariance
%of the state
estimpositionsmoothed=zeros(n_xx,n);
PF4=zeros(n_xx,n_xx,n);
for i=1:n

PF4(:,:,i)=inv(inv(Psmooth(:,:,i))+inv(Pback(:,:,n+1-i)));

estimpositionsmoothed(:,i)=(PF4(:,:,i)/Pback(:,:,n+1-
i))*estimpositionbb(1:n_xx,n+1-
i)+(PF4(:,:,i)/Psmooth(:,:,i))*estimposition(1:n_xx,i);


end
```

8.3 Appendix B – Gyroscope sensor model

```
function w=gyro1(a,n)
%frequency 1sec
dt=1;
%gyro bias
bias=0.0005*pi/180;
%initialization and white noise generation
w=zeros(1,n);
gyronoise=0.01*wgn(n,1,0);
%calculation of angular rate from angle inputs
for i=2:dt:n-1

    w(1,i)=((a(1,i+1)-a(1,i))/dt)+bias+gyronoise(i);

end
```

8.4 Appendix C – Pressure sensor model

```
function a=trykk(b,n)
a=zeros(1,n);
%t is Gaussian white noise
t=0.01*wgn(n,1,0);
dt=1;
for k=1:dt:n
    a(k)=b(k)+t(k);
end
```

8.5 Appendix D – DVL sensor model

```
function
[Vreal,alphas,b1,b2,b3,b4]=DVLread(gyroyaw,sysposition,n,gyrovelo,U)

% Beams initial value when at position [2,2,0]'
b1(1)=2*4/3;
b2(1)=2*4/3;
b3(1)=2*4/3;
b4(1)=2*4/3;
%the angle alpha and relative velocity initialization
alphas=zeros(1,n);
Vreal(:,1)=-U;
%measurement noise beams
nx=0.03*wgn(n,1,0);
nx2=0.03*wgn(n,1,0);
nx3=0.03*wgn(n,1,0);
nx4=0.03*wgn(n,1,0);
%measurement noise alpha
ny=0.05*wgn(n,1,0);
%noise on relative velocity
B1=0.01*wgn(n,1,0);
B2=0.01*wgn(n,1,0);
B3=0.01*wgn(n,1,0);
BB=[B1 B2 B3];
%dd is the vector o1o2 from net frame origin to body/ROV frame origin
dd=zeros(3,n);

for i=2:n

    %alpha
    b1(i)=(sysposition(1,i)/cos(pi/6+gyroyaw(i))/cos(pi/6))+nx(i);
    b2(i)=(sysposition(1,i)/cos(pi/6-gyroyaw(i)))/cos(pi/6)+nx2(i);
    b3(i)=(sysposition(1,i)/cos(pi/6-gyroyaw(i)))/cos(pi/6)+nx3(i);
    b4(i)=(sysposition(1,i)/cos(pi/6+gyroyaw(i)))/cos(pi/6)+nx4(i);
    a1(i)=(b2(i)+b3(i))/2;
    a2(i)=(b1(i)+b4(i))/2;
    AB(i)=(a1(i)/tan(pi/3))+a2(i)/tan(pi/3);
    BC(i)=a2(i)-a1(i);
    alphas(i)=atan(BC(i)/AB(i))+ny(i);
    %v is The ROV velocity vector represented in net frame
    v(1,i)=sysposition(1,i)-sysposition(1,i-1);
    v(2,i)=sysposition(2,i)-sysposition(2,i-1);
    v(3,i)=sysposition(3,i)-sysposition(3,i-1);
    %dd is the vector o1o2 from net frame origin to body/ROV frame origin
    dd(1,i)=sysposition(1,i);
    dd(2,i)=sysposition(2,i);
    dd(3,i)=sysposition(3,i);
    %calculating R and omega
    Rnewnetttorov=[-cos(gyroyaw(i)) sin(gyroyaw(i)) 0; -sin(gyroyaw(i)) -
cos(gyroyaw(i)) 0; 0 0 1];
    Wrovnett=[0 -gyrovelo(1,i) 0; gyrovelo(1,i) 0 0; 0 0 0];

    %transforming the velocity from net frame to ROV frame and adding
    %measurement noise BB
    Vreal(:,i)=Rnewnetttorov'*Wrovnett*dd(:,i)-
Rnewnetttorov'*v(:,i)+BB(i,:)';

end
```

8.6 Appendix E – Filter1

```
function [estimpositionDVL,Pdvlgyro]=Filter1(M,U,measposition1,n)
%get the yaw and yaw angular rate estimates from the first linear kf
implementation
[KFestimposi]=Kalman(measposition1,n);
%n_x size of the augmented state
n_x = 10;
%n_xx size of the original state
n_xx=3;
%n_z size of the measurement vector
n_z=4;
%weights calculation
n_p = 2*n_x;
w_i=zeros(n_p+1);
w_z=1/(2*n_x);
w_i(1) = 1-n_x/3;
for i=2:n_p
w_i(i)=(1-w_i(1))/(2*n_x);
end
%the sequence of inputs from ROV pilot
Unew=zeros(3,3);
Unew(:,1)=U;
Unew(:,2)=[0;-1;0];
Unew(:,3)=-U;
%Xstate= x,y,z,vx,vy,vz,psi,psi'     Z=z, vx, vy, vz, psi'
%calculating Q and R
%process noise
wwx=0.09*wgn(n,1,0);
wwy=0.06*wgn(n,1,0);
wwz=0.1*wgn(n,1,0);
wwyaw=0.01*wgn(n,1,0);
wwvyaw=0.01*wgn(n,1,0);
WW=[wwx,wwy,wwz];
bigR=cov(WW);
%measurement noise
vvz=0.01*wgn(n,1,0);
vvvx=0.01*wgn(n,1,0);
vvvy=0.01*wgn(n,1,0);
vvvz=0.01*wgn(n,1,0);
vvvyaw=0.01*wgn(n,1,0);
VV=[vvz,vvvx,vvvy,vvvz];%,vvvyaw];
bigQ=cov(VV);


%initializing the state vector augmented with process and meas noise w and
%v
estimpositionDVL=zeros(n_x,n);
estimpositionDVL(1:3,1)=M;
estimpositionDVL(n_xx:n_x,1)=0;
%diagonalization
diagR=diag(bigR);
DR =diag(diagR,0);
diagQ=diag(bigQ);
DQ=diag(diagQ,0);
%intialization of the covariance matrix
Pprior=zeros(n_x,n_x);
Pp=0.0001*eye(n_xx);
Pprior(1:n_xx,1:n_xx)=Pp;
```

```
Pprior(n_xx+1:2*n_xx,n_xx+1:2*n_xx)=DR;
Pprior(2*n_xx+1:n_x,2*n_xx+1:n_x)=DQ;
%Variable used to store the covariance for each iteration
Pdvlgyro=zeros(n_xx,n_xx);
Pdvlgyro(:,:,1)=Pp;


for k = 1:n-1

    %covariance initialization/update
    Pprior(1:n_xx,1:n_xx)=Pp;
    Pprior(n_xx+1:2*n_xx,n_xx+1:2*n_xx)=DR;
    Pprior(2*n_xx+1:n_x,2*n_xx+1:n_x)=DQ;
    %deciding the input from ROV pilot
     if k<=200
      U=Unew(:,1);
     end
      if (200<=k)&&(k<400)
          U=-Unew(:,2);
      end
      if k>=400
      U=Unew(:,3);
      end
      %Calculating sqrt(P)
        u = chol(Pprior)';
       %sigma points: x_i is for sigma set3
        x_i = zeros(n_x, n_p+1);
        y_i=zeros(n_xx, n_p+1);
        z_i=zeros(n_z, n_p+1);
        %variables to store the means
        ysum=zeros(n_xx,1);
        zsum=zeros(n_z,1);
       %variable used for calculation help
        P=0;
        P1=0;
        P2=0;
        Py=0;
        Pz=0;
        Pyz=0;
        Ph=0;
        Pj=0;

        % sigma points
       estimpositionDVL(n_xx+1:n_x,:)=0;
        for i= 1:n_p


           x_i(:,1)=estimpositionDVL(:,k);
            if(i <= (n_p/2) )
                x_i(:,i+1) = estimpositionDVL(:,k) + sqrt(n_x/(1-
w_i(1)))*u(:,i);

                x_i(:,i+(n_p/2)+1) = estimpositionDVL(:,k) - sqrt(n_x/(1-
w_i(1)))*u(:,i);

            end
        end

        %propagation through process model
```

```
        for i= 1:n_p+1
             %xyz position
            y_i(1:3,i)=x_i(4:6,i)+x_i(1:3,i)+U;
        end


            %Weighted average
            ysum=w_i(1)*y_i(:,1)+w_i(2)*sum(y_i,2);

            %finding Py
            for i = 1:n_p+1
            P=w_i(i)*(y_i(:,i)-ysum)*(y_i(:,i)-ysum)';
            Ph=Ph+P;
            end
            Py=Ph;

           %propagation through measurement model
            for i = 1:n_p+1
                 %z
                 z_i(1,i)=y_i(3,i)+x_i(7,i);
                 %calculating R from rov to net
                 RRR=[-cos(KFestimposi(1,k+1)) sin(KFestimposi(1,k+1)) 0; -
sin(KFestimposi(1,k+1)) -cos(KFestimposi(1,k+1)) 0; 0 0 1];
                 %calculating skew w
                 omega=[0 -KFestimposi(2,k+1) 0;KFestimposi(2,k+1) 0 0; 0 0
0];
                 %ROV velocity in net
                 Gx=y_i(1:3,i)-x_i(1:3,i);
                 %relative velocity
                  z_i(2:4,i)=RRR'*(-Gx+(omega*y_i(1:3,i)))+x_i(8:10,i);

            end
         %Weighted average
         zsum=w_i(1)*z_i(:,1)+w_i(2)*sum(z_i,2);

        %Calculating Pz
            for i = 1:n_p+1
            P1=w_i(i)*(z_i(:,i)-zsum)*(z_i(:,i)-zsum)';
            Pj=Pj+P1;
            P2=w_i(i)*(y_i(:,i)-ysum)*(z_i(:,i)-zsum)';
            Pyz=Pyz+P2;
            end
            Pz=Pj;

            %Kalman gain
            K=Pyz/Pz;

            %measurement update
            estimpositionDVL(1:n_xx,k+1)=ysum+K*(measposition1(1:n_z,k+1)-
zsum);
            Pp=Py-K*Pz*K';
            Pdvlgyro(:,:,k+1)=Pp;

end
```

8.7 Appendix F – Filter1: Linear Kalman filter

```
function [KFestimposi]=Kalman(measposition1,n)
%variable initialization
KFestimposi=zeros(2,n);
KFestimposi(:,1)=0;
%KF process and measurement matrices
A=[1 1;0 1];
B=0;
H=[0 1];
%calculating Q and R
x1=0.1*wgn(n,1,0);
y1=0.1*wgn(n,1,0);
z1=0.1*wgn(n,1,0);
Wk=[x1,y1]';
Vk=[z1]';
Q=cov(Wk');
R=cov(Vk');
U=0;
%covariance initialization
priorposition=zeros(2,n);
Pprior=zeros(2,2,n);
Pestim=zeros(2,2,n);
I=eye(2);
Pestim(:,:,1)=0.0001*eye(2);
Pprior(:,:,1)=0.0001*eye(2);
%Kalman filter
for i=2:n
    %Prediction step
    priorposition(:,i)=(A*KFestimposi(:,i-1)+B*U);
    Pprior(:,:,i)=A*Pestim(:,:,i-1)*A'+Q;
    %measurement update
    K=Pprior(:,:,i)*H'/(H*Pprior(:,:,i)*H'+R);
    KFestimposi(:,i)=priorposition(:,i)+K*(measposition1(5,i)-
H*priorposition(:,i));
    Pestim(:,:,i)=(I-K*H)*Pprior(:,:,i);
end
```

8.8 Appendix G – Filter2

```
function [estimpositionDVL,Pdvlalpha]=Filter2(M,U,measposition1,n)
%n_x size of the augmented state
n_x = 16;
%n_xx size of the original state
n_xx=5;
%n_z size of the meas
n_z=6;
%Weights calculation
n_p = 2*n_x;
w_i=zeros(n_p+1);
w_i(1) = 1-n_x/3;
for i=2:n_p
w_i(i)=(1-w_i(1))/(2*n_x);
end
%the sequence of inputs from ROV pilot
Unew=zeros(3,3);
Unew(:,1)=U;
Unew(:,2)=[0;-1;0];
Unew(:,3)=-U;
%Xstate= x,y,z,psi,psi'     Z=z, vx, vy, vz, psi'
%process noise and covariance R calculation
wwx=0.01*wgn(n,1,0);
wwy=0.02*wgn(n,1,0);
wwz=0.01*wgn(n,1,0);
wwyaw=0.01*wgn(n,1,0);
wwvyaw=0.01*wgn(n,1,0);
WW=[wwx,wwy,wwz,wwyaw,wwvyaw];
bigR=cov(WW);
%measurement noise and measurement covariance Q
vvz=0.01*wgn(n,1,0);
vvvx=0.01*wgn(n,1,0);
vvvy=0.01*wgn(n,1,0);
vvvz=0.01*wgn(n,1,0);
vvvyaw=0.01*wgn(n,1,0);
vvalpha=0.01*wgn(n,1,0);
VV=[vvz,vvvx,vvvy,vvvz,vvvyaw,vvalpha];
bigQ=cov(VV);


%initializing the state vector (augmented with process and meas noise w and
%v
estimpositionDVL=zeros(n_x,n);
estimpositionDVL(4:5,1)=0;
estimpositionDVL(1:3,1)=M;
estimpositionDVL(n_xx:n_x,1)=0;
sigmasett1(:,1)=estimpositionDVL(:,1);

%Diagonalization
diagR=diag(bigR);
DR =diag(diagR,0);
diagQ=diag(bigQ);
DQ=diag(diagQ,0);
%initializing P with Px Q R and Q
Pprior=zeros(n_x,n_x);
Pp=0.001*eye(n_xx);
Pprior(1:n_xx,1:n_xx)=Pp;
Pprior(n_xx+1:2*n_xx,n_xx+1:2*n_xx)=DR;
```

```
Pprior(2*n_xx+1:n_x,2*n_xx+1:n_x)=DQ;

%used to store the covariance for each iteration
Pdvlalpha=zeros(n_xx,n_xx);
Pdvlalpha(:,:,1)=Pp;

for k = 1:n-1

   %covariance initialization/update
    Pprior(1:n_xx,1:n_xx)=Pp;
    Pprior(n_xx+1:2*n_xx,n_xx+1:2*n_xx)=DR;
    Pprior(2*n_xx+1:n_x,2*n_xx+1:n_x)=DQ;
%deciding the input from ROV pilot
     if k<200
      U=Unew(:,1);

     end
      if (200<=k)&&(k<400)
          U=-Unew(:,2);

      end
      if k>=400
      U=Unew(:,3);
      end
        %Calculating sqrt(P)
        u = chol(Pprior)';
          %sigma points: x_i is for sigma set3 and xx_i is for sigma set1
         x_i = zeros(n_x, n_p+1);
         y_i=zeros(n_xx, n_p+1);
         z_i=zeros(n_z, n_p+1);
         %variables to store the means
        ysum=zeros(n_xx,1);
        zsum=zeros(n_z,1);
        %Variable used for calculation help
        P=0;
        P1=0;
        P2=0;
        Py=0;
        Pz=0;
        Pyz=0;
        Ph=0;
        Pj=0;

        %sigma points calculation from augmented stated vector
        estimpositionDVL(n_xx+1:n_x,:)=0;
        for i= 1:n_p

           x_i(:,1)=estimpositionDVL(:,k);
            if(i <= (n_p/2) )
                x_i(:,i+1) = estimpositionDVL(:,k) + sqrt(n_x/(1-
w_i(1)))*u(:,i);

                x_i(:,i+(n_p/2)+1) = estimpositionDVL(:,k) - sqrt(n_x/(1-
w_i(1)))*u(:,i);

            end
        end
        %propagation through process
        for i= 1:n_p+1
```

```
            %xyz position
            y_i(1:3,i)=x_i(6:8,i)+x_i(1:3,i)+U;
           %yaw
            y_i(4,i)=x_i(4,i)+x_i(9,i)+x_i(5,i);
            %angular rate
            y_i(5,i)=x_i(5,i)+x_i(10,i);

        end


            %Calculating the weighted average
            ysum=w_i(1)*y_i(:,1)+w_i(2)*sum(y_i,2);

            %finding Py
            for i = 1:n_p+1
            P=w_i(i)*(y_i(:,i)-ysum)*(y_i(:,i)-ysum)';
            Ph=Ph+P;
            end
            Py=Ph;

           %propagation through the measurement model
            for i = 1:n_p+1
                 %z
                 z_i(1,i)=y_i(3,i)+x_i(11,i);
                 %angular velocity
                 z_i(5,i)=x_i(15,i)+y_i(5,i);
                 %alpha
                 z_i(6,i)=y_i(4,i)+x_i(16,i);
                 %calcultaing R from rov to net
                 RRR=[-cos(y_i(4,i)) sin(y_i(4,i)) 0; -sin(y_i(4,i)) -
cos(y_i(4,i)) 0; 0 0 1];
                 %calculating skew w
                 omega=[0 -y_i(5,i) 0;y_i(5,i) 0 0; 0 0 0];
                 %ROV velocity in net frame
                 Gx=y_i(1:3,i)-x_i(1:3,i);
                 %relative velocity
                 z_i(2:4,i)=RRR'*(-Gx+(omega*y_i(1:3,i)))+x_i(12:14,i);

            end
            %calculating weighted average
         zsum=w_i(1)*z_i(:,1)+w_i(2)*sum(z_i,2);


           %calcualting Pz
            for i = 1:n_p+1
            P1=w_i(i)*(z_i(:,i)-zsum)*(z_i(:,i)-zsum)';
            Pj=Pj+P1;
            P2=w_i(i)*(y_i(:,i)-ysum)*(z_i(:,i)-zsum)';
            Pyz=Pyz+P2;
            end
            Pz=Pj;

            %Kalman gain
            K=Pyz/Pz;

            %measurement update
```

```
            estimpositionDVL(1:n_xx,k+1)=ysum+K*(measposition1(1:n_z,k+1)-
zsum);
            Pp=Py-K*Pz*K';
            Pdvlalpha(:,:,k+1)=Pp;

end
```

8.9 Appendix H – Filter3

```
function
[estimposition,sigmasett1,Psmooth,Psmooths1]=Filter3(M,U,measposition1,n)
dt=1;
%n_x size of the augmented state
n_x = 20;
%n_xx size of the original state
n_xx=5;
%n_z size of the measurement vector
n_z=10;
n_p = 2*n_x;
%weights calculation
w_i=zeros(n_p+1);
w_z=1/(2*n_x);
w_i(1) = 1-n_x/3;
for i=2:n_p
w_i(i)=(1-w_i(1))/(2*n_x);
end
%the sequence of motion from ROV pilot
Unew=zeros(3,3);
Unew(:,1)=U;
Unew(:,2)=[0;-1;0];
Unew(:,3)=-U;
%Xstate= x,y,z,psi,psi'     Z=z, vx, vy, vz, psi', b1...b4,Alpha
%process noise and covariance R calculation
wwx=0.01*wgn(n,1,0);
wwy=0.01*wgn(n,1,0);
wwz=0.01*wgn(n,1,0);
wwyaw=0.01*wgn(n,1,0);
wwvyaw=0.01*wgn(n,1,0);
WW=[wwx,wwy,wwz,wwyaw,wwvyaw];
bigR=cov(WW);

%measurement noise and measurement covariance Q
vvz=0.01*wgn(n,1,0);
vvvx=0.01*wgn(n,1,0);
vvvy=0.01*wgn(n,1,0);
vvvz=0.01*wgn(n,1,0);
vvvyaw=0.01*wgn(n,1,0);
vvb1=0.01*wgn(n,1,0);
vvb2=0.01*wgn(n,1,0);
vvb3=0.01*wgn(n,1,0);
vvb4=0.01*wgn(n,1,0);
vvalpha=0.01*wgn(n,1,0);
VV=[vvz,vvvx,vvvy,vvvz,vvvyaw,vvb1,vvb2,vvb3,vvb4,vvalpha];
bigQ=cov(VV);

%initializing the state vector, (augmented with process and meas noise w)
%estimposion is the estimate using sigma set3 and sigmasett1 is the
%estimate using sigma set1
estimposition=zeros(n_x,n);
sigmasett1=zeros(n_x,n);

estimposition(4,1)=0;
estimposition(5,1)=0;
estimposition(1:3,1)=M;
estimposition(6:n_x,1)=0;
sigmasett1(:,1)=estimposition(:,1);
```

```
%diagonalization
diagR=diag(bigR);
DR =diag(diagR,0);
diagQ=diag(bigQ);
DQ=diag(diagQ,0);



%initialization of the augmented covariance
%sigma set3
Pprior=zeros(n_x,n_x);
Pp=0.001*eye(n_xx);
%sigmasett1
Ppriors1=zeros(n_x,n_x);
Pps1=0.001*eye(n_xx);



%Psmooth and Psmooths1 are used to store the covariance for each iteration
Psmooth=zeros(n_xx,n_xx,n);
Psmooth(:,:,1)=Pp;
Psmooths1=zeros(n_xx,n_xx,n);
Psmooths1(:,:,1)=Pp;

for k = 1:n-1

    %covariance initialization/update
    Pprior(1:n_xx,1:n_xx)=Pp;
    Ppriors1(1:n_xx,1:n_xx)=Pps1;

    Pprior(n_xx+1:2*n_xx,n_xx+1:2*n_xx)=DR;
    Pprior(2*n_xx+1:n_x,2*n_xx+1:n_x)=DQ;

    Ppriors1(n_xx+1:2*n_xx,n_xx+1:2*n_xx)=DR;
    Ppriors1(2*n_xx+1:n_x,2*n_xx+1:n_x)=DQ;

%deciding the input from ROV pilot
  if k<200
      U=Unew(:,1);
  end
      if (200<=k)&&(k<400)
          U=-Unew(:,2);
      end
  if k>=400
      U=Unew(:,3);
  end
     %Calculating sqrt(P)

        u = chol(Pprior)';
        us1 = chol(Ppriors1)';
     %sigma points: x_i is for sigma set3 and xx_i is for sigma set1
         x_i = zeros(n_x, n_p+1);
         xx_i = zeros(n_x, n_p);
         y_i=zeros(n_xx, n_p+1);
         yy_i=zeros(n_xx, n_p);
```

```
        z_i=zeros(n_z, n_p+1);
        zz_i=zeros(n_z, n_p+1);
        %variables to store the means
        ysum=zeros(n_xx,1);
        yysum=zeros(n_xx,1);
        zsum=zeros(n_z,1);
        zzsum=zeros(n_z,1);
        %Variable used for calculation help
        P=0;
        P1=0;
        P2=0;
        Py=0;
        Pz=0;
        Pyz=0;
        Ph=0;
        Pj=0;
        Ps1=0;
        P1s1=0;
        P2s1=0;
        Pys1=0;
        Pzs1=0;
        Pyzs1=0;
        Phh=0;
        Pjs1=0;

        % sigma points calculation from augmented stated vector
       estimposition(n_xx+1:n_x,:)=0;
       sigmasett1(n_xx+1:n_x,:)=0;

       %sigma set3
        for i= 1:n_p

           x_i(:,1)=estimposition(:,k);
            if(i <= (n_p/2) )
                x_i(:,i+1) = estimposition(:,k) + sqrt(n_x/(1-
w_i(1)))*u(:,i);

                x_i(:,i+(n_p/2)+1) = estimposition(:,k) - sqrt(n_x/(1-
w_i(1)))*u(:,i);

            end
        end
        %sigma sett1
        for i= 1:n_p

            if(i <= (n_p/2) )
                xx_i(:,i) = sigmasett1(:,k) + sqrt(n_x)*us1(:,i);

                xx_i(:,i+(n_p/2)) = sigmasett1(:,k) - sqrt(n_x)*us1(:,i);

            end
        end
         %propagation through process
        for i= 1:n_p+1
            y_i(4,i)=x_i(9,i)+x_i(4,i)+x_i(5,i);
            %xyz position
            y_i(1:3,i)=x_i(6:8,i)+x_i(1:3,i)+U*dt;
            %angular rate
            y_i(5,i)=x_i(10,i)+x_i(5,i);%+x_i(5,i);
```

```
        end
        %sigma-sett 1
        for i= 1:n_p
             yy_i(4,i)=xx_i(9,i)+xx_i(4,i)+xx_i(5,i);
             yy_i(1:3,i)=xx_i(6:8,i)+xx_i(1:3,i)+U;
             yy_i(5,i)=xx_i(10,i)+xx_i(5,i);
        end

            %Calculating the weighted average
            ysum=w_i(1)*y_i(:,1)+w_i(2)*sum(y_i,2);
            yysum=w_z*sum(yy_i,2);

            %finding Pyy
            %sigma set3
            for i = 1:n_p+1
            P=w_i(i)*(y_i(:,i)-ysum)*(y_i(:,i)-ysum)';
            Ph=Ph+P;
            end
            Py=Ph;

            %sigma set1
            for i = 1:n_p
                Ps1=w_z*(yy_i(:,i)-yysum)*(yy_i(:,i)-yysum)';
                Phh=Phh+Ps1;
            end
                Pys1=Phh;

            %propagation through measurement model

            for i = 1:n_p+1
                 %z
                 z_i(1,i)=y_i(3,i)+x_i(11,i);
                 %angular rate
                 z_i(5,i)=x_i(15,i)+y_i(5,i);
                 %alpha
                 z_i(10,i)=y_i(4,i)+x_i(20,i);
                 %calculating R from rov to net
                 RRR=[-cos(y_i(4,i)) sin(y_i(4,i)) 0; -sin(y_i(4,i)) -
cos(y_i(4,i)) 0; 0 0 1];
                 %calculating skew w
                 omega=[0 -y_i(5,i) 0;y_i(5,i) 0 0; 0 0 0];
                 %Rov velocity in net frame
                 Gx=y_i(1:3,i)-x_i(1:3,i);
                 %relative velocity
                 z_i(2:4,i)=RRR'*(-Gx+(omega*y_i(1:3,i)))+x_i(12:14,i);
                %beams
             z_i(6,i)=x_i(16,i)+(y_i(1,i)/cos(pi/6-y_i(4,i))/cos(pi/6));
             z_i(7,i)=x_i(17,i)+(y_i(1,i)/cos(pi/6+y_i(4,i))/cos(pi/6));
             z_i(8,i)=x_i(18,i)+(y_i(1,i)/cos(pi/6+y_i(4,i))/cos(pi/6));
             z_i(9,i)=x_i(19,i)+(y_i(1,i)/cos(pi/6-y_i(4,i))/cos(pi/6));
            end

            %sigmasett1
             for i = 1:n_p
                 %z
                 zz_i(1,i)=yy_i(3,i)+xx_i(11,i);
                 %Alpha
                 zz_i(10,i)=yy_i(4,i)+xx_i(20,i);
```

```
                %angular rate
                zz_i(5,i)=xx_i(15,i)+yy_i(5,i);
                %calcultaing R from rov to net
                RRRs1=[-cos(yy_i(4,i)) sin(yy_i(4,i)) 0; -sin(yy_i(4,i)) -
cos(yy_i(4,i)) 0; 0 0 1];
                %calculating skew w
                omegas1=[0 -yy_i(5,i) 0;yy_i(5,i) 0 0; 0 0 0];
                %Rov velocity in net frame
                Gxs1=yy_i(1:3,i)-xx_i(1:3,i);
                %relative velocity
                zz_i(2:4,i)=RRRs1'*(-
Gxs1+(omegas1*RRRs1'*yy_i(1:3,i)))+xx_i(12:14,i);
                %Beams
             zz_i(6,i)=xx_i(16,i)+(yy_i(1,i)/cos(pi/6-
yy_i(4,i))/cos(pi/6));

zz_i(7,i)=xx_i(17,i)+(yy_i(1,i)/cos(pi/6+yy_i(4,i))/cos(pi/6));

zz_i(8,i)=xx_i(18,i)+(yy_i(1,i)/cos(pi/6+yy_i(4,i))/cos(pi/6));
             zz_i(9,i)=xx_i(19,i)+(yy_i(1,i)/cos(pi/6-
yy_i(4,i))/cos(pi/6));
             end

             %calculating the weighted average
             zzsum=w_z*sum(zz_i,2);
             zsum=w_i(1)*z_i(:,1)+w_i(2)*sum(z_i,2);


           %calculating Pz
            %sigma set3
            for i = 1:n_p+1
            P1=w_i(i)*(z_i(:,i)-zsum)*(z_i(:,i)-zsum)';
            Pj=Pj+P1;
            P2=w_i(i)*(y_i(:,i)-ysum)*(z_i(:,i)-zsum)';
            Pyz=Pyz+P2;
            end
            Pz=Pj;
            %sigmasett1
            for i = 1:n_p
            P1s1=w_z*(zz_i(:,i)-zzsum)*(zz_i(:,i)-zzsum)';
            Pjs1=Pjs1+P1s1;
            P2s1=w_z*(yy_i(:,i)-yysum)*(zz_i(:,i)-zzsum)';
            Pyzs1=Pyzs1+P2s1;
            end
            Pzs1=Pjs1;
           %Kalman gain
            K=Pyz/Pz;
            Ks1=Pyzs1/Pzs1;

           %measurement update
            %sigma set3
            estimposition(1:n_xx,k+1)=ysum+K*(measposition1(:,k+1)-zsum);
            Pp=Py-K*Pz*K';
            Psmooth(:,:,k+1)=Pp;
            %sigma set1
            sigmasett1(1:n_xx,k+1)=yysum+Ks1*(measposition1(:,k+1)-zzsum);
            %sigmasett1(4,k+1)=-sygmasett1(4,k+1);
            Pps1=Pys1-Ks1*Pzs1*Ks1';
            Psmooths1(:,:,k+1)=Pps1;
end
```

8.10 Appendix I – Filter4: Backward run

```
function [estimpositionbb,Pback]=BackKF(measposition1,n,sysposition)
dt=1;
%n_x size of the augmented state
n_x = 19;
%n_xx size of the original state
n_xx=5;
%n_z size of the measurement
n_z=9;
%weights calculation
n_p = 2*n_x;
w_i=zeros(n_p+1);

w_i(1) = 1-n_x/3;
for i=2:n_p
w_i(i)=(1-w_i(1))/(2*n_x);
end
%the sequence of motion from ROV pilot
Unew=zeros(3,3);
Unew(:,1)=[0;0;-1];
Unew(:,2)=[0;1;0];
Unew(:,3)=[0;0;1];
%reversing measurement velocity
measposition1(2:5,:)=-measposition1(2:5,:);
%Xstate= x,y,z,psi,psi'     Z=z, vx, vy, vz, psi', b1...b4
%process noise and covariance R calculation
wwx=0.01*wgn(n,1,0);
wwy=0.01*wgn(n,1,0);
wwz=0.01*wgn(n,1,0);
wwyaw=0.01*wgn(n,1,0);
wwvyaw=0.05*wgn(n,1,0);
WW=[wwx,wwy,wwz,wwyaw,wwvyaw];
bigR=cov(WW);
%measurement noise and covariance Q
vvz=0.01*wgn(n,1,0);
vvb1=0.01*wgn(n,1,0);
vvb2=0.01*wgn(n,1,0);
vvb3=0.01*wgn(n,1,0);
vvb4=0.01*wgn(n,1,0);
vvvx=0.3*wgn(n,1,0);
vvvy=0.3*wgn(n,1,0);
vvvz=0.01*wgn(n,1,0);
vvvyaw=0.2*wgn(n,1,0);
VV=[vvz,vvvx,vvvy,vvvz,vvvyaw,vvb1,vvb2,vvb3,vvb4];
bigQ=cov(VV);

%initializing the state vector augmented with process and measurement noise
w and
%v
estimpositionbb=zeros(n_x,n);
estimpositionbb(4:n_x,1)=0;
estimpositionbb(1:3,1)=sysposition(:,n);
%diagonalization
diagR=diag(bigR);
DR =diag(diagR,0);
diagQ=diag(bigQ);
DQ=diag(diagQ,0);
%initializing P with Px Q R and Q
```

```
Pprior=zeros(n_x,n_x);
Pp=0.0000001*eye(5);
Pprior(1:5,1:5)=Pp;
Pprior(6:10,6:10)=DR;
Pprior(11:n_x,11:n_x)=DQ;
%used to store the covariance for each iteration
Pback=zeros(n_xx,n_xx);
Pback(:,:,1)=Pp;

for k = 1:n-1

    %covariance initialization/update
    Pprior(1:n_xx,1:n_xx)=Pp;
    Pprior(6:10,6:10)=DR;
    Pprior(11:19,11:19)=DQ;

 %deciding the input from ROV pilot
  if k<=200
      U=Unew(:,1);
  end
      if (200<k)&&(k<400)
          U=-Unew(:,2);
      end
  if k>=400
      U=Unew(:,3);
  end
     %Calculating sqrt(P)
        u = chol(Pprior)';

        %sigma points calculation using sigma set3
        %initialization
        x_i = zeros(n_x, n_p+1);
        y_i=zeros(n_xx, n_p+1);
        z_i=zeros(n_z, n_p+1);
        ysum=zeros(n_xx,1);
        zsum=zeros(n_z,1);
        %variable for calculation help
        P=0;
        P1=0;
        P2=0;
        Py=0;
        Pz=0;
        Pyz=0;
        Ph=0;
        Pj=0;


        % sigma points calculation

       estimpositionbb(n_xx+1:n_x,:)=0;
        for i= 1:n_p

           x_i(:,1)=estimpositionbb(:,k);
            if(i <= (n_p/2) )
                x_i(:,i+1) = estimpositionbb(:,k) + sqrt(n_x/(1-
w_i(1)))*u(:,i);

                x_i(:,i+(n_p/2)+1) = estimpositionbb(:,k) - sqrt(n_x/(1-
w_i(1)))*u(:,i);
```

```
        end
    end


     %propagation through process
    for i= 1:n_p+1

        %yaw
        y_i(4,i)=x_i(9,i)+x_i(4,i)+x_i(5,i);
        %xyz position
        y_i(1:3,i)=x_i(6:8,i)+x_i(1:3,i)+U*dt;
        %angular rate
         y_i(5,i)=x_i(10,i)+x_i(5,i);
    end


        %Calculating the weighted average
        ysum=w_i(1)*y_i(:,1)+w_i(2)*sum(y_i,2);

        %calculating Py
        for i = 1:n_p+1
        P=w_i(i)*(y_i(:,i)-ysum)*(y_i(:,i)-ysum)';
        Ph=Ph+P;
        end
        Py=Ph;


        %propagation through measurement model

        for i = 1:n_p+1
             %z
             z_i(1,i)=y_i(3,i)+x_i(11,i);
             %angular velocity
             z_i(5,i)=x_i(15,i)+y_i(5,i);%+y_i(4,i)-x_i(4,i);
             %calcultaing R from rov to nett
             RRR=[-cos(y_i(4,i)) sin(y_i(4,i)) 0; -sin(y_i(4,i)) -
cos(y_i(4,i)) 0; 0 0 1];
             %calculating skew w
             omega=[0 -y_i(5,i) 0;y_i(5,i) 0 0; 0 0 0];
             %ROV velocity in net
             Gx=y_i(1:3,i)-x_i(1:3,i);
             %relative velocity
             z_i(2:4,i)=RRR'*(-Gx+(omega*y_i(1:3,i)))+x_i(12:14,i);

            %beams
          z_i(6,i)=x_i(16,i)+(y_i(1,i)/cos(pi/6-y_i(4,i))/cos(pi/6));
          z_i(7,i)=x_i(17,i)+(y_i(1,i)/cos(pi/6+y_i(4,i))/cos(pi/6));
          z_i(8,i)=x_i(18,i)+(y_i(1,i)/cos(pi/6+y_i(4,i))/cos(pi/6));
          z_i(9,i)=x_i(19,i)+(y_i(1,i)/cos(pi/6-y_i(4,i))/cos(pi/6));

        end

         %Calculating the weighted average
        zsum=w_i(1)*z_i(:,1)+w_i(2)*sum(z_i,2);

       %Calculating Pz
        for i = 1:n_p+1
```

```
            P1=w_i(i)*(z_i(:,i)-zsum)*(z_i(:,i)-zsum)';
            Pj=Pj+P1;
            P2=w_i(i)*(y_i(:,i)-ysum)*(z_i(:,i)-zsum)';
            Pyz=Pyz+P2;
            end
            Pz=Pj;

            %kalman gain
            K=Pyz/Pz;

           %measurement update
            estimpositionbb(1:n_xx,k+1)=ysum+K*(measposition1(:,n-k)-zsum);
            Pp=Py-K*Pz*K';
            Pback(:,:,k+1)=Pp;

end
```

8.11 Appendix J – Plot of all results

```
%Filter1
%plot the true path
plot3(sysposition(1,:),sysposition(2,:),sysposition(3,:),'b');
hold on
%plot the estimated path filter1
plot3(estimpositionDVL1(1,:),estimpositionDVL1(2,:),estimpositionDVL1(3,:),
'r');
legend('True path','Filter1');
title('True path Vs Filter1 estimate')
grid on
xlabel('X') % x-axis label
ylabel('Y') % y-axis label
zlabel('Z') % y-axis label
% 1-sigma limits calculation
varians=zeros(3,n);
Presult=Pdvlgyro;
for i=1:n
varians(1,i)=sqrt(Presult(1,1,i));
varians(2,i)=sqrt(Presult(2,2,i));
varians(3,i)=sqrt(Presult(3,3,i));
end
%plot error Vs 1-sigma and 2-sigma
Error1=estimpositionDVL1(1:3,:)-sysposition;
%error x-direction
figure()
set(gcf, 'Color', [1,1,1]);
plot(1*varians(1,:),'r--');
hold on
plot(2*varians(1,:),'r');
hold on
plot(Error1(1,:),'b');
hold on
plot(-1*varians(1,:),'r--');
hold on
plot(-2*varians(1,:),'r');
hold on
legend('1-sigma','2-sigma','Error');
title('Estimation error X-axis Filter1')
xlabel('time(second)')
ylabel('Error(meter)')
grid on
%ydirection
figure()
set(gcf, 'Color', [1,1,1]);
plot(1*varians(2,:),'r--');
hold on
plot(2*varians(2,:),'r');
hold on
plot(Error1(2,:),'b');
hold on
plot(-1*varians(2,:),'r--');
hold on
plot(-2*varians(2,:),'r');
hold on
legend('1-sigma','2-sigma','Error ');
title('Estimation error Y-axis Filter1')
xlabel('time(second)')
ylabel('Error(meter)')
```

```
grid on
%z_direction
figure()
set(gcf, 'Color', [1,1,1]);
plot(1*varians(3,:),'r--');
hold on
plot(2*varians(3,:),'r');
hold on
plot(Error1(3,:),'b');
hold on
plot(-1*varians(3,:),'r--');
hold on
plot(-2*varians(3,:),'r');
hold on
legend('1-sigma','2-sigma','Error');
title('Estimation error Z-axis Filter1')
xlabel('time(second)')
ylabel('Error(meter)')
grid on



%Filter2
%plot the true path
figure()
plot3(sysposition(1,:),sysposition(2,:),sysposition(3,:),'b');
hold on
%plot the estimated path
plot3(estimpositionDVL(1,:),estimpositionDVL(2,:),estimpositionDVL(3,:),'r'
);
legend('True path','Filter2');
title('True path Vs Filter2 estimate')
grid on
xlabel('X') % x-axis label
ylabel('Y') % y-axis label
zlabel('Z') % y-axis label
%sigma limits calculation
varians=zeros(3,n);
Presult=Pdvlalpha;
for i=1:n
varians(1,i)=sqrt(Presult(1,1,i));
varians(2,i)=sqrt(Presult(2,2,i));
varians(3,i)=sqrt(Presult(3,3,i));
end
%defining the error and plot Vs 1-sigma and 2-sigma
Error2=estimpositionDVL(1:3,:)-sysposition;
%error x-direction
figure()
set(gcf, 'Color', [1,1,1]);
plot(1*varians(1,:),'r--');
hold on
plot(2*varians(1,:),'r');
hold on
plot(Error2(1,:),'b');
hold on
plot(-1*varians(1,:),'r--');
hold on
plot(-2*varians(1,:),'r');
hold on
legend('1-sigma','2-sigma','Error');
```

```
title('Estimation error X-axis Filter2')
xlabel('time(second)')
ylabel('Error(meter)')
grid on
%ydirection
figure()
set(gcf, 'Color', [1,1,1]);
plot(1*varians(2,:),'r--');
hold on
plot(2*varians(2,:),'r');
hold on
plot(Error2(2,:),'b');
hold on
plot(-1*varians(2,:),'r--');
hold on
plot(-2*varians(2,:),'r');
hold on
legend('1-sigma','2-sigma','Error');
title('Estimation error Y-axis Filter2')
xlabel('time(second)')
ylabel('Error(meter)')
grid on
%z_direction
figure()
set(gcf, 'Color', [1,1,1]);
plot(1*varians(3,:),'r--');
hold on
plot(2*varians(3,:),'r');
hold on
plot(Error2(3,:),'b');
hold on
plot(-1*varians(3,:),'r--');
hold on
plot(-2*varians(3,:),'r');
hold on
legend('1-sigma','2-sigma','Error');
title('Estimation error Z-axis Filter2')
xlabel('time(second)')
ylabel('Error(meter)')
grid on



%plot filter1 VS filter 2
figure()
plot3(sysposition(1,:),sysposition(2,:),sysposition(3,:),'b');
hold on
plot3(estimpositionDVL1(1,:),estimpositionDVL1(2,:),estimpositionDVL1(3,:),
'r');
hold on
plot3(estimpositionDVL(1,:),estimpositionDVL(2,:),estimpositionDVL(3,:),'k'
);
legend('True path',' Filter1','Filter2');
title('True path Vs Filter1 estimate Vs Filter2 estimate');
grid on
xlabel('X') % x-axis label
ylabel('Y') % y-axis label
zlabel('Z') % y-axis label
```

```
%comparing error from filter 1 and 2

figure()
set(gcf, 'Color', [1,1,1]);
plot(Error1(1,:),'r');
hold on
plot(Error2(1,:),'b')
hold on
legend('Filter1 ','Filter2 ');
title('Estimation error X-axis Filter1 Vs Filter2');
xlabel('time(second)')
ylabel('Error(meter)')
grid on

figure()
set(gcf, 'Color', [1,1,1]);
plot(Error1(2,:),'r');
hold on
plot(Error2(2,:),'b');
hold on
legend('Filter1','Filter2');
title('Estimation error Y-axis Filter1 Vs Filter2');
xlabel('time(second)')
ylabel('Error(meter)')
grid on

figure()
set(gcf, 'Color', [1,1,1]);
plot(Error1(3,:),'r');
hold on
plot(Error2(3,:),'b')
legend('Filter1','Filter2');
title('Estimation error Z-axis Filter1 Vs Filter2')
xlabel('time(second)')
ylabel('Error(meter)')
grid on


%Filter 3
%sigmasett3
%plot the true path
figure()
plot3(sysposition(1,:),sysposition(2,:),sysposition(3,:),'b');
hold on
%plot the estimated path
plot3(estimposition(1,:),estimposition(2,:),estimposition(3,:),'r');
legend('True path','Filter3');
title('True path Vs Filter3 Sigma-sett3 estimate')
grid on
xlabel('X') % x-axis label
ylabel('Y') % y-axis label
zlabel('Z') % y-axis label
%sigma limits calculation
varians=zeros(3,n);
Presult=Psmooth;
for i=1:n
varians(1,i)=sqrt(Presult(1,1,i));
varians(2,i)=sqrt(Presult(2,2,i));
varians(3,i)=sqrt(Presult(3,3,i));
end
```

```
%defining the error and plot Vs 1-sigma and 2-sigma
Error33=estimposition(1:3,:)-sysposition;
%error x-direction

figure()
set(gcf, 'Color', [1,1,1]);
plot(1*varians(1,:),'r--');
hold on
plot(2*varians(1,:),'r');
hold on
plot(Error33(1,:),'b');
hold on
plot(-1*varians(1,:),'r--');
hold on
plot(-2*varians(1,:),'r');
hold on
legend('1-sigma','2-sigma','Error');
title('Estimation error X-axis Filter3 Sigma-sett3')
xlabel('time(second)')
ylabel('Error(meter)')
grid on
%ydirection

figure()
set(gcf, 'Color', [1,1,1]);
plot(1*varians(2,:),'r--');
hold on
plot(2*varians(2,:),'r');
hold on
plot(Error33(2,:),'b');
hold on
plot(-1*varians(2,:),'r--');
hold on
plot(-2*varians(2,:),'r');
hold on
legend('1-sigma','2-sigma','Error');
title('Estimation error Y-axis Filter3 Sigma-sett3')
xlabel('time(second)')
ylabel('Error(meter)')
grid on
%z_direction

figure()
set(gcf, 'Color', [1,1,1]);
plot(1*varians(3,:),'r--');
hold on
plot(2*varians(3,:),'r');
hold on
plot(Error33(3,:),'b');
hold on
plot(-1*varians(3,:),'r--');
hold on
plot(-2*varians(3,:),'r');
hold on
legend('1-sigma','2-sigma','Error');
title('Estimation error Z-axis Filter3 Sigma-sett3')
xlabel('time(second)')
ylabel('Error(meter)')
grid on
```

```
%sigmasett1
%plot the true path
figure()
plot3(sysposition(1,:),sysposition(2,:),sysposition(3,:),'b');
hold on
%plot the estimated path
plot3(sigmasett1(1,:),sigmasett1(2,:),sigmasett1(3,:),'r');
legend('True path','Filter3');
title('True path Vs Filter3 Sigma-sett1 estimate')
grid on
xlabel('X') % x-axis label
ylabel('Y') % y-axis label
zlabel('Z') % y-axis label
%sigma limits calculation
varians=zeros(3,n);
Presult=Psmooths1;
for i=1:n
varians(1,i)=sqrt(Presult(1,1,i));
varians(2,i)=sqrt(Presult(2,2,i));
varians(3,i)=sqrt(Presult(3,3,i));
end
%defining the error and plot Vs 1-sigma and 2-sigma
Error31=sigmasett1(1:3,:)-sysposition;
%error x-direction

figure()
set(gcf, 'Color', [1,1,1]);
plot(1*varians(1,:),'r--');
hold on
plot(2*varians(1,:),'r');
hold on
plot(Error31(1,:),'b');
hold on
plot(-1*varians(1,:),'r--');
hold on
plot(-2*varians(1,:),'r');
hold on
legend('1-sigma','2-sigma','Error');
title('Estimation error X-axis Filter3 Sigma-sett1')
xlabel('time(second)')
ylabel('Error(meter)')
grid on
%ydirection

figure()
set(gcf, 'Color', [1,1,1]);
plot(1*varians(2,:),'r--');
hold on
plot(2*varians(2,:),'r');
hold on
plot(Error31(2,:),'b');
hold on
plot(-1*varians(2,:),'r--');
hold on
plot(-2*varians(2,:),'r');
hold on
legend('1-sigma','2-sigma','Error');
title('Estimation error Y-axis Filter3 Sigma-sett1')
xlabel('time(second)')
ylabel('Error(meter)')
```

```
grid on
%z_direction

figure()
set(gcf, 'Color', [1,1,1]);
plot(1*varians(3,:),'r--');
hold on
plot(2*varians(3,:),'r');
hold on
plot(Error31(3,:),'b');
hold on
plot(-1*varians(3,:),'r--');
hold on
plot(-2*varians(3,:),'r');
hold on
legend('1-sigma','2-sigma','Error');
title('Estimation error Z-axis Filter3 Sigma-sett1')
xlabel('time(second)')
ylabel('Error(meter)')
grid on
%sigmasett3 VS sigmasett1
%

figure()
plot3(sysposition(1,:),sysposition(2,:),sysposition(3,:),'b');
hold on
plot3(estimposition(1,:),estimposition(2,:),estimposition(3,:),'r');
hold on
plot3(sigmasett1(1,:),sigmasett1(2,:),sigmasett1(3,:),'k');
hold on
legend('True path','Filter3 Sigma-sett3','Filter3 sigma-sett1');
title('True path Vs Filter3 Sigma-sett3 estimate Vs Filter3 Sigma-sett1
estimate')
grid on
xlabel('X') % x-axis label
ylabel('Y') % y-axis label
zlabel('Z') % y-axis label

figure()
set(gcf, 'Color', [1,1,1]);
plot(Error33(1,:),'r');
hold on
plot(Error31(1,:),'b')
hold on
legend('Sigma-sett3','Sigma-sett1');
title('Estimation error X-axis Sigma-sett3 Vs Sigma-sett1');
xlabel('time(second)')
ylabel('Error(meter)')
grid on

figure()
set(gcf, 'Color', [1,1,1]);
plot(Error33(2,:),'r');
hold on
plot(Error31(2,:),'b');
hold on
legend('Sigma-sett3','Sigma-sett1');
title('Estimation error Y-axis Sigma-sett3 Vs Sigma-sett1');
xlabel('time(second)')
ylabel('Error(meter)')
```

```
grid on

figure()
set(gcf, 'Color', [1,1,1]);
plot(Error33(3,:),'r');
hold on
plot(Error31(3,:),'b')
legend('Sigma-sett3','Sigma-sett1');
title('Estimation error Z-axis Sigma-sett3 Vs Sigma-sett1')
xlabel('t(s)')
ylabel('Error(m)')
grid on

%comparing filter 2 and filter3
figure()
plot3(sysposition(1,:),sysposition(2,:),sysposition(3,:),'b');
hold on
plot3(estimposition(1,:),estimposition(2,:),estimposition(3,:),'r');
hold on
plot3(estimpositionDVL(1,:),estimpositionDVL(2,:),estimpositionDVL(3,:),'k'
);
hold on
legend('True path','Filter3','Filter2');
title('True path Vs Filter2 estimate Vs Filter3 Sigma-sett3 estimate')
grid on
xlabel('X') % x-axis label
ylabel('Y') % y-axis label
zlabel('Z') % y-axis label

figure()
set(gcf, 'Color', [1,1,1]);
plot(Error33(1,:),'r');
hold on
plot(Error2(1,:),'b')
hold on
legend('Filter3','Filter2');
title('Estimation error X-axis Filter2 Vs Filter3 Sigma-sett3');
xlabel('time(second)')
ylabel('Error(meter)')
grid on

figure()
set(gcf, 'Color', [1,1,1]);
plot(Error33(2,:),'r');
hold on
plot(Error2(2,:),'b');
hold on
legend('Filter3','Filter2');
title('Estimation error Y-axis Filter2 Vs Filter3 Sigma-sett3');
xlabel('time(second)')
ylabel('Error(meter)')
grid on

figure()
set(gcf, 'Color', [1,1,1]);
plot(Error33(3,:),'r');
hold on
plot(Error2(3,:),'b')
legend('Filter3','Filter2');
title('Estimation error Z-axis Filter2 Vs Filter3 Sigma-sett3')
```

```
xlabel('time(second)')
ylabel('Error(meter)')
grid on



%Filter4

%plot the true path
figure()
plot3(sysposition(1,:),sysposition(2,:),sysposition(3,:),'b');
hold on
%plot the estimated path
plot3(estimpositionsmoothed(1,:),estimpositionsmoothed(2,:),estimpositionsm
oothed(3,:),'r');
legend('True path','Filter4');
title('True path Vs Filter4 smoothed estimate')
grid on
xlabel('X') % x-axis label
ylabel('Y') % y-axis label
zlabel('Z') % y-axis label
%sigma limits calculation
varians=zeros(3,n);
Presult=PF4;
for i=1:n
varians(1,i)=sqrt(Presult(1,1,i));
varians(2,i)=sqrt(Presult(2,2,i));
varians(3,i)=sqrt(Presult(3,3,i));
end
%defining the error and plot Vs 1-sigma and 2-sigma
Errorfinal=estimpositionsmoothed(1:3,:)-sysposition;
%error x-direction

figure()
set(gcf, 'Color', [1,1,1]);
plot(1*varians(1,:),'r--');
hold on
plot(2*varians(1,:),'r');
hold on
plot(Errorfinal(1,:),'b');
hold on
plot(-1*varians(1,:),'r--');
hold on
plot(-2*varians(1,:),'r');
hold on
legend('1-sigma','2-sigma','Error');
title('Estimation error X-axis Filter4')
xlabel('time(second)')
ylabel('Error(meter)')
grid on
%ydirection

figure()
set(gcf, 'Color', [1,1,1]);
plot(1*varians(2,:),'r--');
hold on
plot(2*varians(2,:),'r');
hold on
plot(Errorfinal(2,:),'b');
hold on
```

```
plot(-1*varians(2,:),'r--');
hold on
plot(-2*varians(2,:),'r');
hold on
legend('1-sigma','2-sigma','Error');
title('Estimation error Y-axis Filter4')
xlabel('time(second)')
ylabel('Error(meter)')
grid on
%z_direction

figure()
set(gcf, 'Color', [1,1,1]);
plot(1*varians(3,:),'r--');
hold on
plot(2*varians(3,:),'r');
hold on
plot(Errorfinal(3,:),'b');
hold on
plot(-1*varians(3,:),'r--');
hold on
plot(-2*varians(3,:),'r');
hold on
legend('1-sigma','2-sigma','Error');
title('Estimation error Z-axis Filter4')
xlabel('time(second)')
ylabel('Error(meter)')
grid on
%Plotting Filter3 sigma sett 3 Vs Filter4

figure()
plot3(sysposition(1,:),sysposition(2,:),sysposition(3,:),'b');
hold on
%plot the estimated path
plot3(estimposition(1,:),estimposition(2,:),estimposition(3,:),'r');
hold on
plot3(estimpositionsmoothed(1,:),estimpositionsmoothed(2,:),estimpositionsm
oothed(3,:),'k');
legend('True path','Filter3','Filter4');
title('True path Vs Filter3 sigma-sett3 estimate Vs Filter4 estimate')
grid on
xlabel('X') % x-axis label
ylabel('Y') % y-axis label
zlabel('Z') % y-axis label

%plotting sigmasett3 Vs smoothed error
figure()
set(gcf, 'Color', [1,1,1]);
plot(Error33(1,:),'r');
hold on
plot(Errorfinal(1,:),'b')
hold on
legend('Filter3','Filter4');
title('Estimation error X-axis Filter3 Sigma-sett3 estimate Vs Filter4
estimate');
xlabel('time(second)')
ylabel('Error(meter)')
grid on

figure()
```

```
set(gcf, 'Color', [1,1,1]);
plot(Error33(2,:),'r');
hold on
plot(Errorfinal(2,:),'b');
hold on
legend('Filter3','Filter4');
title('Estimation error Y-axis Filter3 Sigma-sett3 estimate Vs Filter4
estimate');
xlabel('time(second)')
ylabel('Error(meter)')
grid on

figure()
set(gcf, 'Color', [1,1,1]);
plot(Error33(3,:),'r');
hold on
plot(Errorfinal(3,:),'b')
legend('Filter3','Filter4');
title('Estimation error Z-axis Filter3 Sigma-sett3 estimate Vs Filter4
estimate')
xlabel('time(second)')
ylabel('Error(meter)')
grid on
```