Master Thesis

Sensor Fusion and Observer Design for Dynamic Positioning

Zhongyuan Liu

Supervisors : Dr. Ir. M. Godjevac
Teun Fekkes

Professor : Prof.ir. JJ Hopman

TU Delft Delft University of Technology

PRAXIS AUTOMATION TECHNOLOGY

Challenge the future

SENSOR FUSION AND OBSERVER DESIGN FOR DYNAMIC POSITIONING

MASTER OF SCIENCE THESIS

For the degree of Master of Science in Marine Technology at Delft University of Technology

Zhongyuan Liu

January 30, 2015

Faculty of Mechanical, Maritime and Materials Engineering · Delft University of Technology

Report number: SDPO.15.008.m

The thesis of Zhongyuan Liu is approved by the Thesis Examination Committee.

Hans Hopman (Chairman)

Milinko Godjevac (Supervisor)

Teun Fekkes (Supervisor)

Rudy Negenborn (Advisor)

Delft University of Technology

2015

Abstract

The fusion of measurements from distributed sensors for dynamic positioning (DP) system based on state estimation algorithms is studied in this thesis in order to increase the accuracy and redundancy of the reference system in DP and a state observer is also designed to estimate of the low frequency vessel motion for the input of DP controller. Different filters such as lowpass filter, notch filter, Kalman filter (KF), extended Kalman filter (EKF), unscented Kalman filter (UKF) and square-root unscented Kalman filter (SRUKF) are introduced and discussed in this thesis. The square-root unscented Kalman filter is proposed for the fusion of measurements, wave filtering and state estimation based on kinematics while the Kalman filter is used as an observer for the estimation of the state vector of a vessel mathematical model for dynamic positioning operation. Thereafter a Matlab/Simulink based marine system simulator containing external environment, vessel model, reference sensors, sensor fusion system, observer and controller is built to test the algorithms of sensor fusion and observer designed in this thesis. The results of the simulation show that the errors of low frequency position and heading prediction are below 0.2m and 0.3 degree respectively which is accurate enough for DP operation. The dead reckoning starts automatically as soon as the reference system fails and the error of the dead reckoning increases gradually overtime. The position error in dead reckoning is about 5m after 1800s in simulation, which means the DP system is still able to estimate the position of the vessel without any position measurements in certain time, avoiding collision caused by the sudden position loss.

Table of Contents

1. Introduction

This chapter first explains what the dynamic positioning (DP) system is and the background knowledge of DP. Then, the motivation and goal of this thesis is introduced. The previous work related to the topic is briefly described afterwards. At last the structure of this thesis is presented to give readers the outline of the work.

1.1 Dynamic Positioning

There are some specific operations at sea which require a vessel to keep its position. It can be done by mooring system in shallow water. Whereas, it becomes more difficult by using mooring system in deep water where dynamic positioning (DP) is applied to maintain the position of a vessel due to the invalidation or difficulty of mooring system. Three degrees of freedom (surge, sway and yaw) of a vessel which is exposed to environmental forces (wind, current and wave) are controlled by DP system as shown in Figure 1-1.

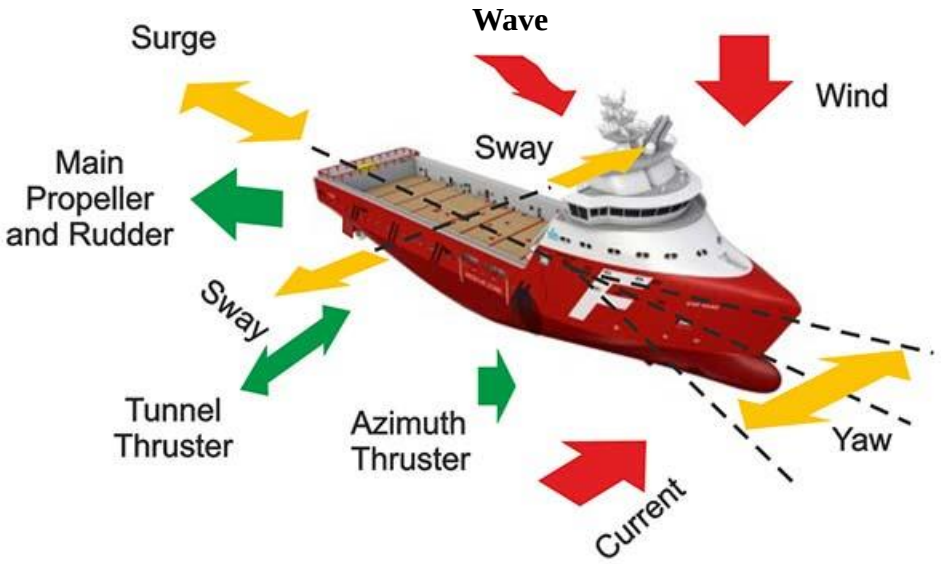


Figure 1-1: What is dynamic positioning. Courtesy of [1]

DP system is a computer control system which keeps the position and heading of a vessel by controlling the propellers and thrusters. The most significant modules of a DP System are (see Figure 1-3):

- Environmental sensors (wind)
- Reference systems (GPS, IMU, gyroscope, radar, etc.)
- Controller
- Thruster interface

Figure 1-2 describes control system and involved external factors in DP and how the relevent componets are interconnected.

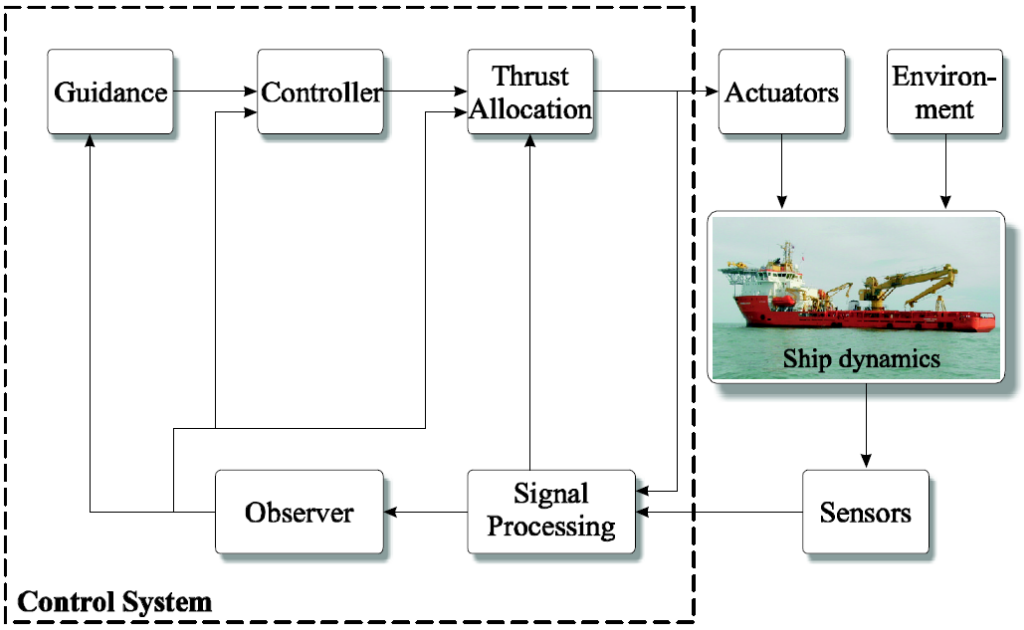


Figure 1-2: Overview of a ship control system. Courtesy of [2]

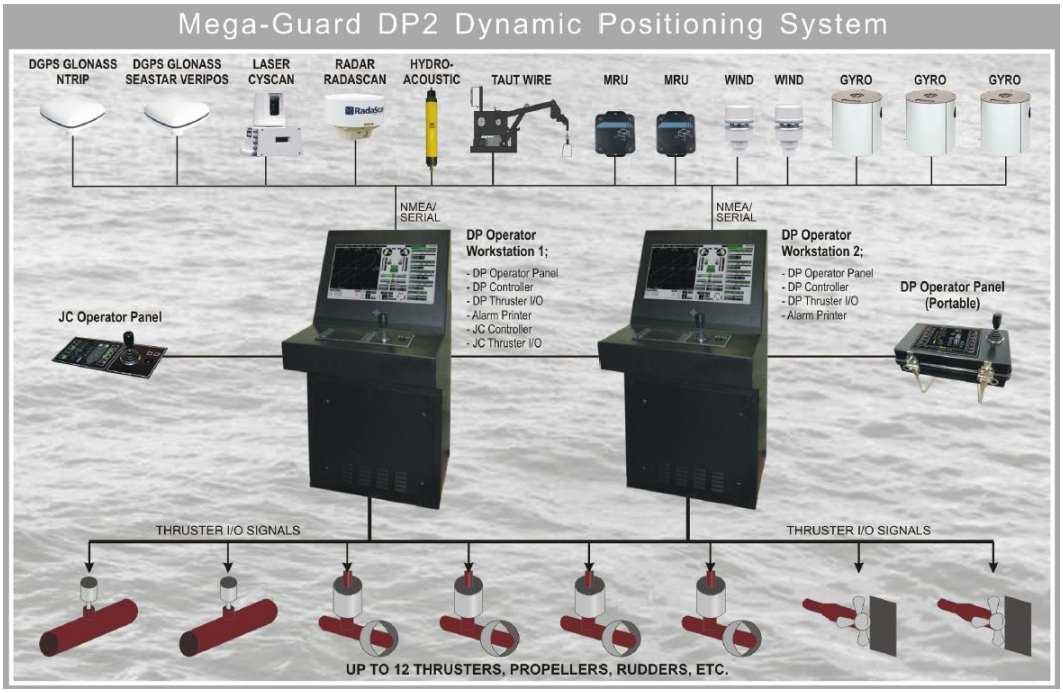


Figure 1-3: Components of DP system. Courtesy of [3]

1.2 Motivation and Goal

The knowledge of heading and position is required in order to carry out DP operation. Since the motions of different points on a vessel are various, it is more convenient to compute the motions and forces about the same point which is usually the center of gravity (CG). Thus, the measurements of sensors such as GPS, IMU, radar, etc. at different locations should be transformed into CG. On one hand, the number of sensors is normally larger than the least required number for position and heading measurement due to the redundancy of the sensor system, so how can we utilize the extra measurements to increase accuracy? [4] On the other hand, the measurements at different points on the vessel are related by rigid body kinematics, so the rotational motion can be calculated based on three known translational motions of three different points. Is it possible to achieve that the failure of the sensor of angles dose not result in the loss of heading by sensor fusion so that the redundancy of the reference system can

be increased? There are few works of sensor fusion for dynamic positioning published in the past, which motivates the author to achieve the first goal of this thesis: that is to design a sensor fusion system to transform all reference sensors into CG and also give estimation about CG motion based on all available sensors. Proper fusion algorithms will be selected among Kalman filter family later.

The external disturbances caused by waves, wind, current and other unmodeled dynamics categorized by second order low frequency (LF) disturbances and first order wave-induced frequency (WF) disturbances. [5] And the motion of the vessel corresponding to LF disturbance and WF disturbance is called the low frequency motion and wave frequency motion respectively. [6] Figure 1-4 shows an example of LF and WF motion. However, neither LF motion nor WF motion is known directly from the reference system which only measures the total motion (superposition of the LF and WF) of the vessel. The second goal of this thesis is aimed to design an observer to estimate the LF motion of the vessel with reference system or without reference system (dead reckoning) as controlling only LW motion in DP operation is a better choice than controlling total motion which results in unacceptable operation for the propulsion system due to wear of the actuators and power consumption. [5]

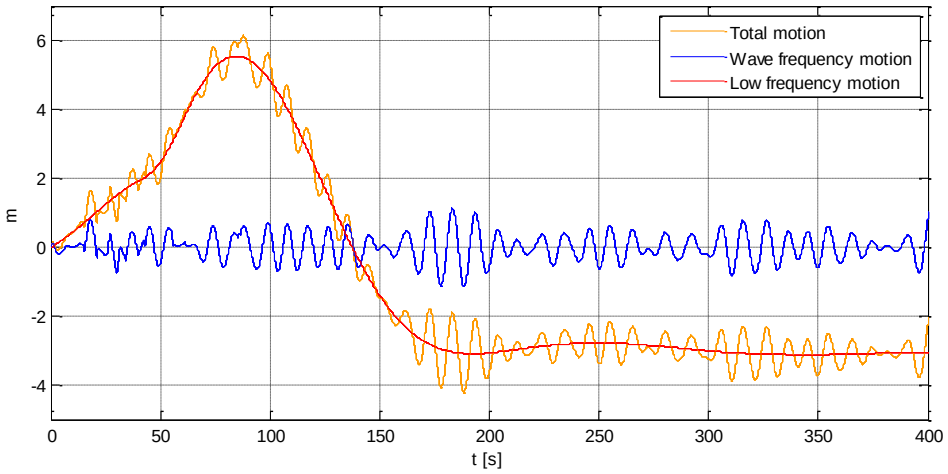


Figure 1-4: Low frequency motion and wave frequency motion

1.3 Previous Work

Simon [7], Paul and Howard [8] proposes different mathematical approaches (various filters are introduced) to the best possible way of estimating the state of a general system. Mobinder and Angus [9] have researched both the theoretical and practical aspects of Kalman filtering by including real-world problems in practice as illustrative examples. When it comes to application of sensor fusion in navigation system, Mayhew [10], Nassar [11] and Han [12] uses fusion of INS and GPS to enhance the position estimate and Mayhew also compares several algorithms for fusion parameter optimization. Van Der Merwe [13] extends traditional extended Kalman filter and deeply researches a family of sigma-point Kalman filters (SPKF). In order to improve the numerical performance in PC computation, Van Der Merwe [14, 15], Zhang [16] and Tang [17] propose square-root sigma-point Kalman filer. The properties of the properties of covariance matrixes of different filters (EKF, UKF, SRUKF and AEKF) are presented by Hovland. [18]

Fossen [5, 19] builds mathematical model of vessel dynamics for positioning control systems. Muhammad [6] uses a mathematical model describing the dynamics of a floating vessel based on Fossen's work and he also sperates low frequency motion from position measurement. Torsetnes [20] designes an observer model for gain-scheduled wave filtering using contraction theory and acceleration meaurement is utilized in his model besides position measurement.

1.4 Structure of the Thesis

- Chapter 1 is the introduction which describes background knowledge, precious work, motivation and goal of this thesis.
- Chapter 2 gives a description of reference frames corresponding to this thesis and transformation method between these reference frames.

- Chapter 3 is a short introduction for different sensors used in reference system for DP.
- Chapter 4 introduces various filters that are considered useful tools in this thesis.
- Chapter 5 is one of the core chapters in this thesis and contains the algorithms for multi-sensor fusion and estimation of total motion of CG.
- Chapter 6 is another core chapter and it focuses on observer design based on the vessel mathematical model.
- Chapter 7 implements the simulation and illustrates the results of the simulation.
- Chapter 8 contains discussion and relevant future work.

The core structure of the thesis is carried out based on Figure 1-5 which details the ship control system in Figure 1-2 according to the contents of the thesis. The green blocks in Figure 1-5 are the major work of this thesis and the 'sensor fusion' block is the work done in Chapter 5 whilst the 'observer' block is done in Chapter 6. The 'sensor fusion' block receives the measurements from all reference sensors and fuses them by proper fusion algorithm, outputting the total motion of CG as the input of 'observer' block which then estimates the low frequency motion of CG.

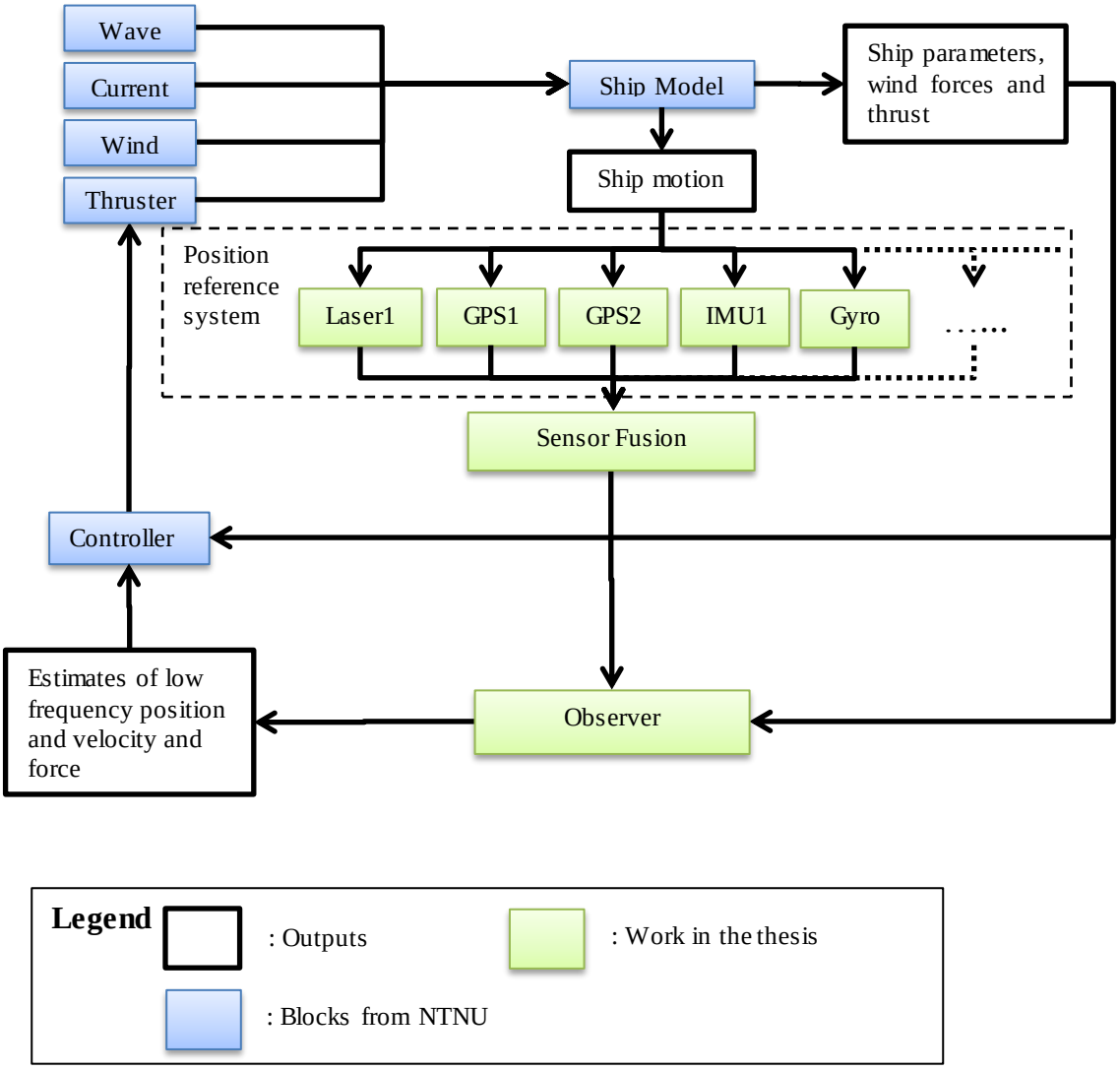


Figure 1-5: Overview diagram of DP in this thesis

2. Reference Frames

The measurements of sensors in reference system are based on various reference frames, which means measured and computed quantities between different reference frames should be unified in the same reference frame. Several reference frames are used in dynamic positioning system. This chapter will introduce the reference frames involved in the calculation in the thesis as well as coordinate transformation between different coordinate systems.

2.1 Earth-centered Earth-fixed Coordinate System

The Earth-centered Earth-fixed (ECEF) coordinate system (x_e, y_e, z_e), defined as a right handed coordinate system, has its origin at the Earth's center of mass with its x-axis fixed on the line of intersection of the prime meridian (0 degree longitude) and the equator (0 degree latitude) and z-axis points to the true north pole. [21] The y-axis complies with the right handed rule as shown in Figure 2-1. This frame rotates relative to the inertial frame with frequency

$$\omega_{ie} \approx 7.292115 \times 10^{-5} / s \tag{2.1}$$

Due to the earth rotation, the ECEF frame is not an inertial reference frame.

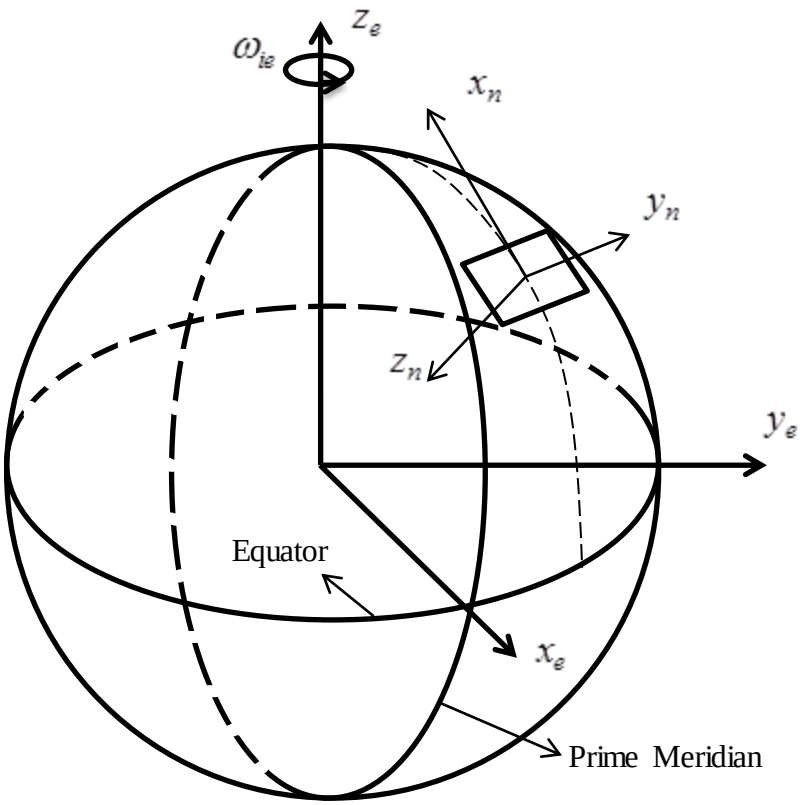


Figure 2-1: North-East-Down Coordinate System (NED) in relation to the ECEF frame. Courtesy of [22]

2.2 North-East-Down Coordinate System

The North-East-Down coordinate system (x_n, y_n, z_n) is the right handed rectangular coordinate system defined by the tangent plane to the geodetic reference ellipse at a point of interest. The geometrical shape of this geodetic reference ellipse is defined by WGS84 geodetic system. The constants of WGS84[21] are shown in Table 2-1. The tangent plane is attached to the point of measurement on a vessel and this point determines the origin of the local frame. In this frame, the x-axis extends to the true north while y-axis points to the east. The z-axis points downwards, perpendicular to the tangent plane mentioned above. This frame is shown in Figure 2-1.

The location of NED relative to ECEF is determined by longitude l and latitude μ. For a vessel operating in a local sea, the tangent plane fixed on the geodetic reference ellipse is used for navigation and this frame can be assumed inertial for simplicity.

Table 2-1 WGS84 defining parameters

Terms	Symbol	Value	Unit
Equatorial radius	a	6378137	m
Reciprocal flattening	$\frac{1}{f}$	298.257223563	
Angular rate	ω_{ie}	7.292115×10^{-5}	s^{-1}
Gravitational constant	GM	3.986004418×10^{14}	m^3/s^2

2.3 Body-fixed Reference Frame

The body-fixed reference frame (x_b, y_b, z_b) is rigidly fixed to the vessel as shown in Figure 2-2. The origin of the frame O_b is usually chosen to coincide with a point at midship in the water line. For marine craft, the axes of body frame are defined as (see Figure 2-2):

- x_b : longitudinal axis (pointing to fore)
- y_b : transversal axis (pointing to starboard)
- z_b : normal axis (pointing to bottom)

One objective of dynamic position system is to determine the position, velocity and heading based on the measurements of different sensors mounted on the vessel. The measurements of linear acceleration are expressed in the body-fixed coordinate system. And also it is more convenient to use body-fixed coordinate system in hydrodynamic computations.

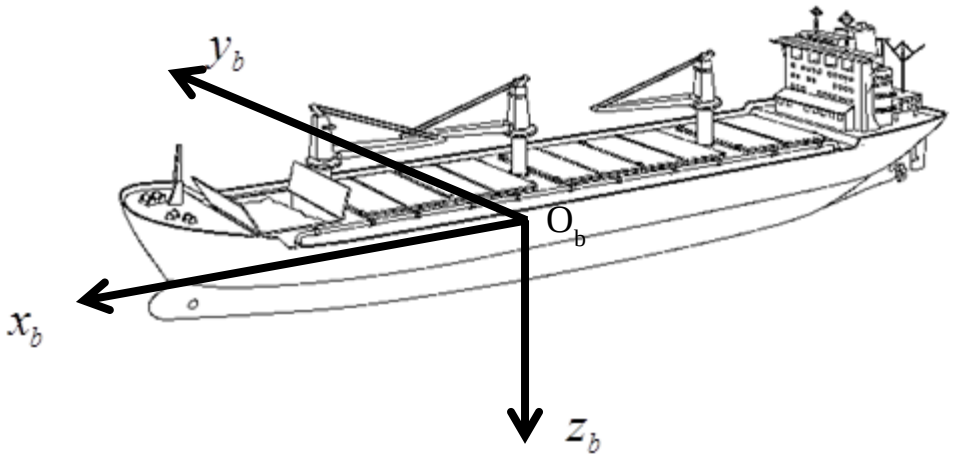


Figure 2-2: Body-fixed reference frame. Courtesy of [23]

2.4 Transformations between Body-fixed Frame and NED

An arbitrary column vector $\vec{\mathbf{V}}$ expressed in NED is denoted as $\vec{\mathbf{V}}_n$ and in body-fixed reference frame $\vec{\mathbf{V}}_b$. The body-fixed frame has three angular orientations referred to NED defined by Euler angles. [24] The three Euler angles roll, pitch and yaw are denoted as ϕ, θ and ψ respectively. See Figure 2-3. The argument of three angles is

$$\mathbf{\Theta}_{E|n} = [\phi, \theta, \psi]^T \tag{2.2}$$

The angular rate $\mathbf{\Omega}_{E|n}$ and angular acceleration $\mathbf{A}_{E|n}$ with respect to time are respectively:

$$\mathbf{\Omega}_{E|n} = \dot{\mathbf{\Theta}}_{E|n} = [\dot{\phi}, \dot{\theta}, \dot{\psi}]^T \tag{2.3}$$

$$\mathbf{A}_{E|n} = \ddot{\mathbf{\Theta}}_{E|n} = [\ddot{\phi}, \ddot{\theta}, \ddot{\psi}]^T \tag{2.4}$$

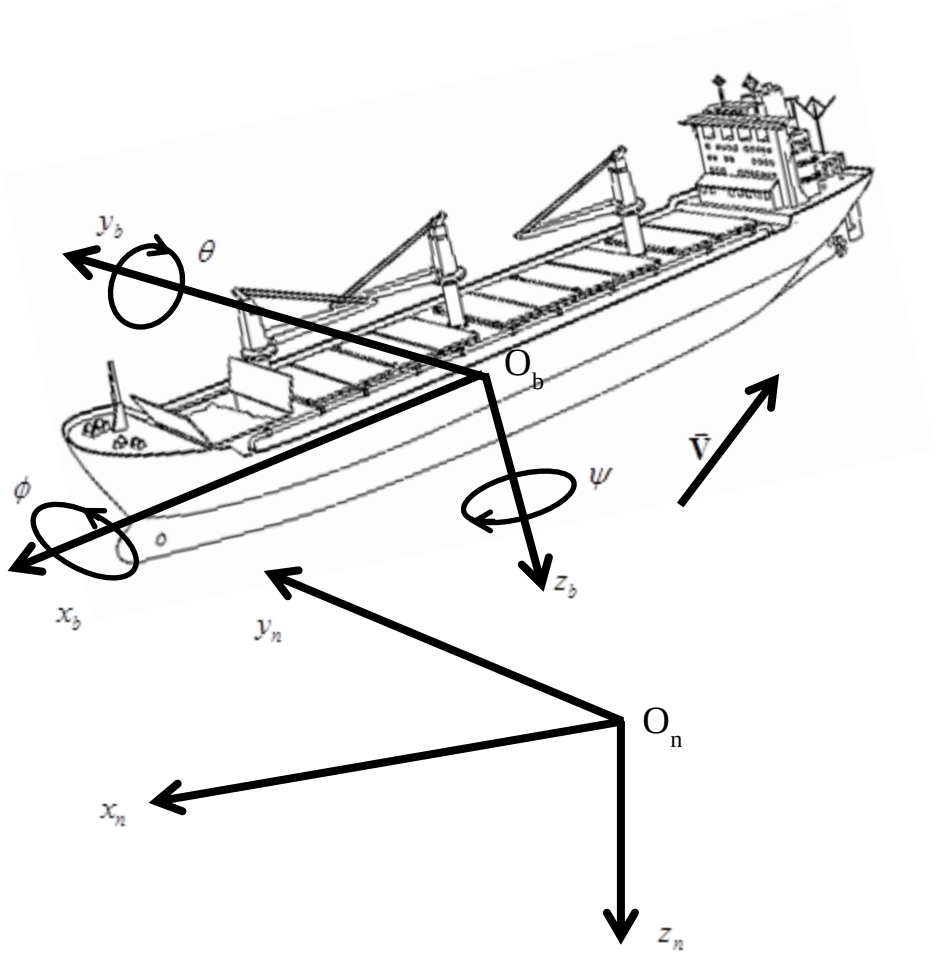


Figure 2-3 Body-fixed frame and NED

The principal rotation matrixes corresponding to the x, y and z axes can be respectively written as

$$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}, \mathbf{R}_y = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}, \mathbf{R}_z = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.5)$$

An arbitrary rotation transformation from body-fixed coordinates to NED coordinates about a single principle axis can be written as

$$\vec{\mathbf{V}}_n = \mathbf{R}_i \vec{\mathbf{V}}_b \tag{2.6}$$

As all the transformation matrixes are orthonormal, their inverse is equivalent to their transpose, which yields

$$\vec{\mathbf{V}}_b = \mathbf{R}_i^T \vec{\mathbf{V}}_n \tag{2.7}$$

where $i = x, y, z$ with respect to rotation about x, y or z axes.

The three independent rotations about three axes can be cascaded through matrix multiplication:

$$\vec{\mathbf{V}}_n = \mathbf{R}_z \mathbf{R}_y \mathbf{R}_x \vec{\mathbf{V}}_b \tag{2.8}$$

Let $\mathbf{R}$ denotes $\mathbf{R}_z \mathbf{R}_y \mathbf{R}_x$, and it can be easily proved that R is also orthonormal.

$$\vec{\mathbf{V}}_n = \mathbf{R} \vec{\mathbf{V}}_b \tag{2.9}$$

Where

$$\mathbf{R} = \begin{bmatrix} \cos\psi\cos\theta & -\sin\psi\cos\phi + \cos\psi\sin\theta\sin\phi & \sin\psi\sin\phi + \cos\psi\cos\phi\sin\theta \\ \sin\psi\cos\theta & \cos\psi\cos\phi + \sin\phi\sin\theta\sin\psi & -\cos\psi\sin\phi + \sin\theta\sin\psi\cos\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix} \tag{2.10}$$

Similarly, $\vec{\mathbf{V}}_b$ can be expressed by $\vec{\mathbf{V}}_n$

$$\vec{\mathbf{V}}_n = \mathbf{R}^T \vec{\mathbf{V}}_b \tag{2.11}$$

This multiplication order $\mathbf{R}_z \mathbf{R}_y \mathbf{R}_x$ is determined by rotation order corresponding to axes. The multiplication order in formula (2.6) means the body-fixed frame rotates about its z-axis first, then y-axis and x-axis at the last, which is the only order in this thesis due to the measurements of Euler angles from sensors based on this order.

If roll and pitch motion are small, it is approximated that $\cos\theta \approx 1$, $\cos\phi \approx 1$, $\sin\theta \approx \theta$ and $\sin\phi \approx \phi$. Then the transformation matrix $\mathbf{R}$ simplified as

$$\mathbf{R} \approx \begin{bmatrix} \cos\psi & -\sin\psi + \theta\phi\cos\psi & \phi\sin\psi + \theta\cos\psi \\ \sin\psi & \cos\psi + \theta\phi\sin\psi & -\phi\cos\psi + \theta\sin\psi \\ -\theta & \phi & 1 \end{bmatrix} \tag{2.12}$$

According to (2.9), the transformation of a position vector between NED and body-fixed frame is obtained by

$$\vec{\mathbf{p}}_n = \mathbf{R}\vec{\mathbf{p}}_b \tag{2.13}$$

The position vector $\vec{\mathbf{p}}_b$ of a fixed point in the body-fixed frame is constant, so the first order derivative of (2.13) with respect to time can be expressed as

$$\dot{\vec{\mathbf{p}}}_n = \dot{\mathbf{R}}\vec{\mathbf{p}}_b \tag{2.14}$$

The left term $\dot{\vec{\mathbf{p}}}_n$ in (2.14) is the velocity vector $\vec{v}_n$, then (2.14) yields

$$\vec{\mathbf{v}}_n = \dot{\mathbf{R}}\vec{\mathbf{p}}_b \tag{2.15}$$

Through the second order derivative of (2.13) the acceleration vector is derived

$$\vec{\mathbf{a}}_n = \ddot{\mathbf{R}}\vec{\mathbf{p}}_b \tag{2.16}$$

$\dot{\mathbf{R}}$ and $\ddot{\mathbf{R}}$ can be easily calculated using symbolic derivative in Matlab but their expressions are too large to be presented here.

2.5 Conclusion

The transformation between body-fixed frame and NED frame is derived in this chapter and it relates the motions in these two reference frames, which is the fundamental of

building kinematic equations for the filters in the later chapters. Note that the transformation is based on the assumption of low vessel speed where the Coriolis forces can be ignored.

3. Reference Systems

In order to keep position and heading, the prerequisite is that position and heading are known. There are various reference sensors such as GPS, inertial measurement unit (IMU), hydro-acoustic position reference (HPR), laser, radar, gyroscope, etc. (see Figure 3-1) to determine the translational and rotational motions of the vessel in operation at sea. These reference sensors involved in DP system will be introduced in this chapter.

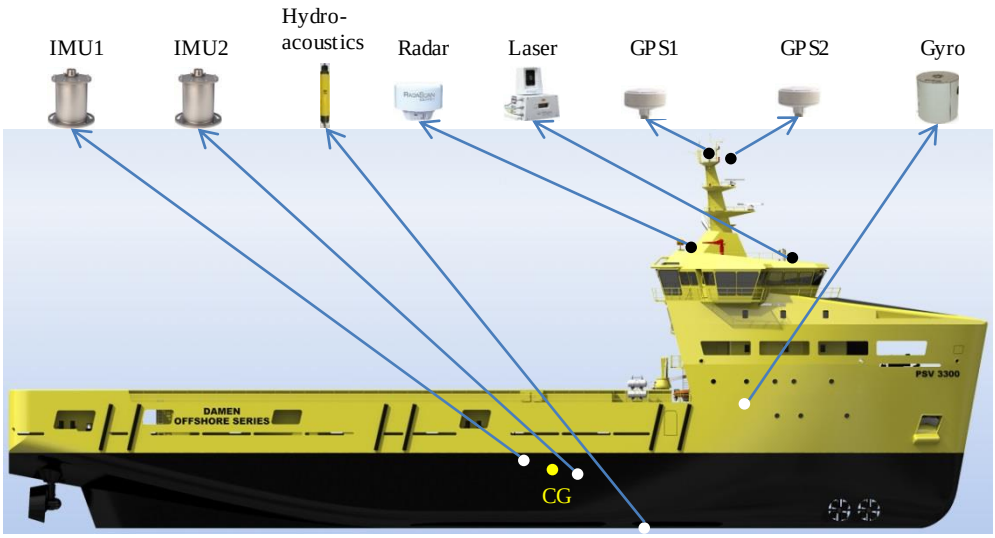


Figure 3-1: Position reference system (Figure from [25])

3.1 GPS

The Global Positioning System (GPS) is a satellite-based radio navigation system which provides continuous coverage in the globe. The satellites roam in the orbits which are

nearly circular with inclination angles of 55 degrees and have altitudes of about 20,200 km above the earth surface. [21] A GPS receiver decodes the signal sent by the satellites to determine the position. It must work with at least four observable satellites and its accuracy increases if more satellite signals are received.

There are several factors causing errors in the position measurement of GPS, such as ionospheric effect, tropospheric effect, ephemeris error, satellite clock error, multipath distortion and numerical error. See Table 3-1. The accuracy of standard GPS is limited to about 15 meters which is not precise enough in dynamic positioning.

Table 3-1 GPS errors

Different errors	Standard GPS /m	DGPS (500 km) /m
Ionospheric delay	5	0.5
Tropospheric delay	0.5	0.1
Ephemeris error	2	0
Multipath distortion	0.5	0.7
Satellite clock	2	0
Receiver white noise	0.3	0.3

Therefore, vessels are normally equipped with differential GPS (DGPS) which is an enhancement to GPS. The errors caused by ionosphere, troposphere, satellite ephemeris and clock are spatially and temporally correlated. [21] These errors could be estimated by one base receiver and broadcast to other GPS receivers. As a result, the positioning accuracy is improved (Table 3-1). This kind of GPS is called DGPS. See Figure: 3-2.

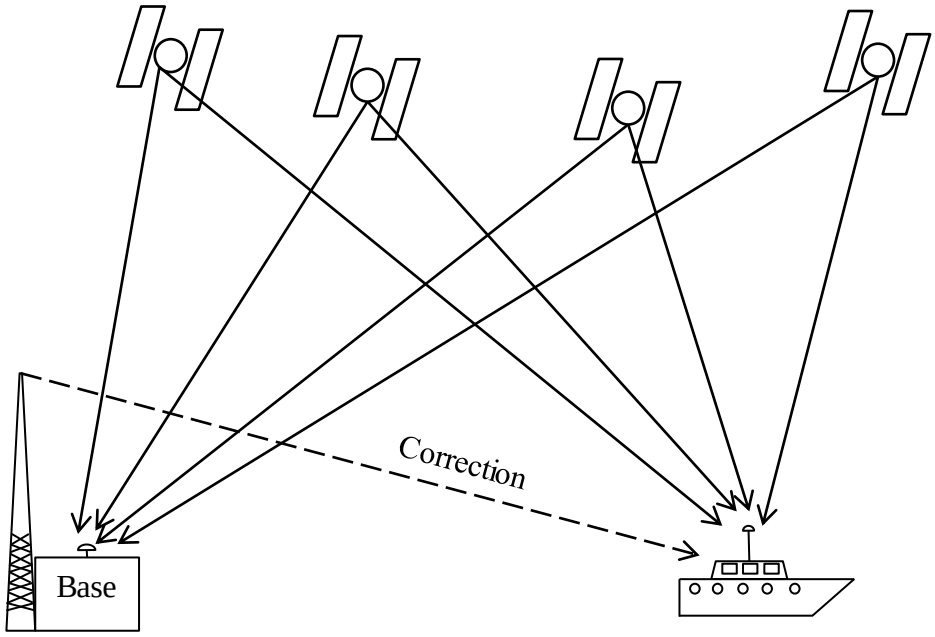


Figure: 3-2 Differential GPS

3.2 Inertial Measurement Unit

Inertial Measurement Units (IMUs) is a self-contained system that measures linear and angular motion usually with a triad of gyroscopes and triad of accelerometers. An IMU can either be gimballed or strapdown, outputting the integrating quantities of angular velocity and acceleration in the sensor/body frame. They are commonly referred to in literature as the rate-integrating gyroscopes and accelerometers.

Inertial Measurement Unit (IMU) sensors are designed to provide motion, position, and navigational sensing from a durable single device over six degrees of freedom. This is achieved by using MEMS (microelectromechanical system) technology to sense translational movement in three perpendicular axes (surge, heave and sway) and rotational movement about three perpendicular axes (roll, pitch and yaw). Because the movement and rotation along the three axes are independent of each other, such motion

is said to have “six degrees of freedom" as shown in Figure 3-3. The output of IMU consists of Euler anglers, Euler angler changing rates and linear accelerations.

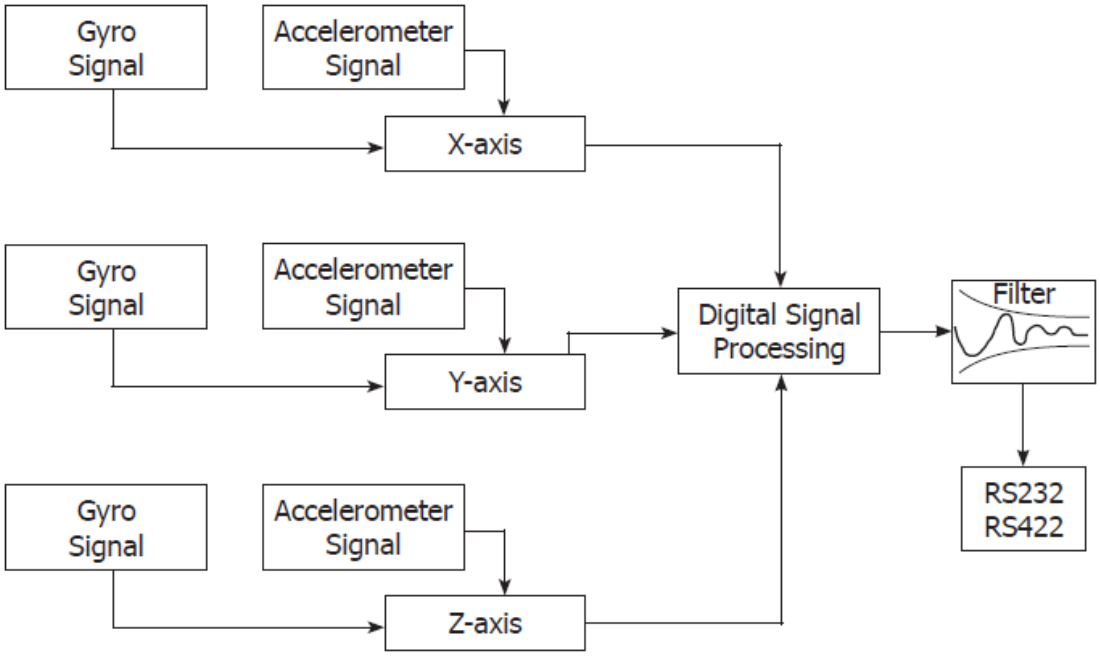


Figure 3-3: System structure of IMU [26]

3.3 Hydro-acoustic Position Reference

Hydro-acoustic positioning system is an underwater positioning system that uses a vessel mounted transceiver to detect the range and bearing to a target using acoustic signals. [27] It consists of both a transmitter (transducer) and a receiver (transponder). The transducer sends signal aimed towards the seabed transponder. Then the transponder is activated and responds immediately to the vessel transducer which then calculates a position of the transponder relative to the vessel. [28] The positioning accuracy of hydro-acoustic positioning system is proportional to the distance between the transceiver and transponder. According to the length of baseline, hydro-acoustics system is divided into different types: Long baseline system (<10com), short baseline system (20m-50m) and ultra-short baseline system (100m-6000m). [29]

3.4 Laser

Laser measures range by measuring the time it takes from sending the beam of light to receiving the reflection from the target. The introduction of scanning lasers allowed for vessel movement in the water by incorporating a vertical fan shaped laser beam which make it easier to keep the target in sight than single point laser beam does. [30] Lasers can the range and bearing of retro-reflective targets with high accuracy during DP operation and they can function 24 hours a day in most weather conditions.

3.5 Radar

RadaScan is an advanced position reference sensor for long range use in marine Dynamic Positioning applications. The RadaScan sensor is a rotating scanner mounted on the DP equipped vessel. It emits a microwave beam and accurately measures the range and bearing of one or more intelligent microwave targets called responders, allowing for the calculation of vessel position and heading. [31]

3.6 Conclusion

Reference sensors for dynamic positioning system are all mounted at different locations on the vessel as shown in Figure 3-1 for instance. GPS is normally located on top of a mast to avoid signal block. Hence, the rotational motion of the vessel results in large movement of GPS due to the long rotational arm between the GPS and rotating center. Consequently, there is an obvious error between the position measurement and the true position of CG. However, the measurements of GPS can be transformed to CG based on the method described in section 2.4. Similarly, the measurements of other position reference sensors like Laser CyScan, Acoustics and RadaScan can be processed in the

same way with GPS. Apparently, IMU is also influenced by the rotating arm and it is not possible to measure the acceleration of one point different from IMU location directly. The optimal positioning of the IMU is at the vessel center of gravity for the sake of best performance. [26] However, normally it is not possible to be mounted exactly at CG in practice. Therefore, the linear acceleration measurements of IMU should also be transformed to CG.

4. Different Filters

There are complex measurement systems working in DP system. Apparently, the measurements are inevitably noisy, which means proper filters have to be implemented to eliminate the unwanted noises. Besides, the mathematical model of the system may be inaccurate which may be equivalent to a noisy system. Some measures have to be taken if the inaccurate mathematical model is used for calculation. Different types of filters are also available to achieve the target and they will be discussed in this chapter.

4.1 Low Pass Filter and Notch Filter

A low-pass filter is a filter that passes signals with a frequency lower than a certain cutoff frequency and attenuates signals with frequencies higher than the cutoff frequency, [32] and the transfer function of first order lowpass filter is

$$H(s) = \frac{\omega_c}{s + \omega_c} \tag{3.1}$$

where ω_c is the cutoff frequency.

A notch filter is a band-stop filter with a narrow stopband (high Q factor), which passes most frequencies except those in a specific range [33] and the transfer function is expressed as

$$H(s) = \frac{s^2 + \omega_n}{s^2 + 2\omega_n s + \omega_n^2} \tag{3.2}$$

where ω_n is the notch frequency.

Low pass filter and notch filter is easy to implement but they have inherent flaws as is mentioned in [34], "Early dynamic positioning systems were implemented using PID controllers. In order to restrain thruster rambling caused by the wave-induced motion components, notch filters in cascade with low pass filters were used with the controllers. However, notch filters restrict the performance of closed loop systems because they introduce phase lag around the crossover frequency, which in turn tends to decrease phase margin."

4.2 Kalman Filter

Weighted average is an average in which some data count more strongly than others [35] and it is an important tool in mathematics and statistics to estimate the value of a variable with different measurements. The Kalman filter is weighted average in nature since it takes the weighted average of prediction and measurement. It is a set of mathematical equations that provides a recursive estimate for the state of a process, to minimize the mean of the squared error. "The filter is very powerful in several aspects: it supports estimations of past, present, and even future states, and it can do so even when the precise nature of the modeled system is unknown." [36]

4.2.1 Mathematical Model of Discrete Kalman filter

The discrete process model and measurement model of discrete Kalman filter can be described respectively by

$$\mathbf{x}_k = \mathbf{A}_k \mathbf{x}_{k-1} + \mathbf{B}_k \mathbf{u}_{k-1} + \boldsymbol{w}_{k-1} \tag{3.3}$$

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \boldsymbol{\upsilon}_k \tag{3.4}$$

where $\mathbf{x} \in \Re^n$ is state vector to be estimated. The matrix $\mathbf{A}_k \in \Re^{n \times n}$ in equation (3.3) is the process transition matrix which relates previous state at time step $k-1$ to that at

time step k. The matrix $\mathbf{B}_k \in \Re^{n\times l}$ is the control transition matrix that relates control vector $\mathbf{u}_{k-1} \in \Re^{l}$ to the state vector $\mathbf{x}_k$. The matrix $\mathbf{H}_k \in \Re^{m\times n}$ is the measurement transition matrix relating the state vector x to the measurement vector $\mathbf{z}$. The vector $\boldsymbol{w}_k$ and $\boldsymbol{\upsilon}_k$ represent the process and measurement noise. Assume that they are independent of each other and comply with normal probability distributions

$$\mathbf{p}(\boldsymbol{w}_k) \sim N(0, \mathbf{Q}_k)$$
$$\mathbf{p}(\boldsymbol{\upsilon}_k) \sim N(0, \mathbf{R}_k)$$

Process noise may not have physical meaning sometimes. However, it can be used as a parameter indicating that the model of the real world we build in the filter is not precise.[8]

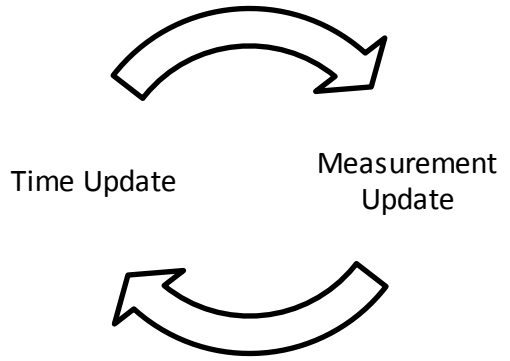


Figure 4-1: the ongoing discrete Kalman filter cycle

Figure 4-1 shows the discrete Kalman filter cycle. The time update projects the current state estimate ahead in time. The measurement update adjusts the projected estimate by an actual measurement at that time.

The specific equations for the time and measurement updates are presented as follows

Time update equations:

$$\hat{\mathbf{x}}_k^- = \mathbf{A}_k \hat{\mathbf{x}}_{k-1} + \mathbf{B}_k \mathbf{u}_{k-1} \tag{3.5}$$

$$\mathbf{P}_k^- = \mathbf{A}_k \mathbf{P}_{k-1} \mathbf{A}_k{}^T + \mathbf{Q}_k \tag{3.6}$$

Measurement update equations:

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k{}^T (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k{}^T + \mathbf{R}_k)^{-1} \tag{3.7}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^-) \tag{3.8}$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^- \tag{3.9}$$

The computation procedure for on Kalman filter cycle is given by (3.5)-(3.9). The a priori state estimated at time step k is defined by $\hat{\mathbf{x}}_k^- \in \Re^n$, which is computed by (3.5) using information $\hat{\mathbf{x}}_{k-1}$ and $\mathbf{u}_{k-1}$ from previous time step $k-1$. The a priori estimate error covariance is $\mathbf{P}_k^- \in \Re^{n \times n}$ and it is computed by (3.6) based on previous $\mathbf{P}_{k-1}$. The Kalman gain $\mathbf{K}_k \in \Re^{n \times m}$ is calculated via (3.7). The a posteriori state estimate $\hat{\mathbf{x}}_k$ in (3.8) is a linear combination of $\hat{\mathbf{x}}_k^-$ and a weighted difference between the measurement $\mathbf{z}_k$ and the predicted measurement $\mathbf{H}_k \hat{\mathbf{x}}_k^-$. The estimation error covariance $\mathbf{P}_k$ is given by (3.9). If all transition matrixes in (3.3) and (3.4) are constant as well as the noise covariance matrixes $\mathbf{Q}_k$ and $\mathbf{R}_k$, the system is time invariant. Under this circumstance, Kalman gain $\mathbf{K}_k$ will stabilize quickly and remain constant; [36] hence, it can be calculated offline to reduce the real-time computational loads.

4.2.2 General Equations for Kalman filter

In order to apply Kalman filter, the model of the system to be estimated must be described by a set of differential equations, the state space form of which is written as [8]

$$\dot{\mathbf{x}} = \mathbf{F}\mathbf{x} + \mathbf{G}\mathbf{u} + \boldsymbol{w} \tag{3.10}$$

and the measurement equation is

$$\mathbf{z} = \mathbf{H}\mathbf{x} + \boldsymbol{\upsilon} \tag{3.11}$$

where $\mathbf{x} \in \Re^n$ is a column vector of state to be estimated by Kalman filter. The matrix $\mathbf{F} \in \Re^{n \times n}$ in equation (3.10) is the process transition matrix which relates previous state at time step $k-1$ to that at time step k. The matrix $\mathbf{G} \in \Re^{n \times l}$ is the control transition matrix that relates control vector $\mathbf{u} \in \Re^l$ to the state vector $\mathbf{x}$. The matrix $\mathbf{H} \in \Re^{m \times n}$ is the measurement transition matrix relating the state vector x to the measurement vector $\mathbf{z}$. The vector $\boldsymbol{w}$ and $\boldsymbol{\upsilon}$ represent the process and measurement noise respectively. The process noise covariance matrix $\mathbf{Q}$ and measurement noise covariance matrix $\mathbf{R}$ is related to noise vector $\boldsymbol{w}$ and $\boldsymbol{\upsilon}$ according to

$$\begin{aligned} \mathbf{Q} &= E[\boldsymbol{w}\boldsymbol{w}^T] \\ \mathbf{R} &= E[\boldsymbol{\upsilon}\boldsymbol{\upsilon}^T] \end{aligned} \tag{3.12}$$

The equation (3.10) must be discretized to build a discrete Kalman filter in order to implement the filter on a computer. Assume that the time interval of measurements T_s is constant, then the process transition matrix $\mathbf{A}$ in section 4.2.1 is computed by the Taylor series expansion [8]

$$\mathbf{A}(T_s) = \mathrm{e}^{\mathbf{F}T_s} = \mathbf{I} + \mathbf{F}T_s + \frac{(\mathbf{F}T_s)^2}{2!} + \cdots + \frac{(\mathbf{F}T_s)^n}{n!} + \cdots \tag{3.13}$$

Then $\mathbf{B}_k$ in section 4.2.1 is obtained from

$$\mathbf{B}_k = \int_0^{T_s} \mathbf{A}(\tau)\mathbf{G}d\tau \tag{3.14}$$

and $\mathbf{Q}_k$ is given by

$$\mathbf{Q}_k = \int_0^{T_s} \mathbf{A}(\tau)\mathbf{Q}\mathbf{A}^T(\tau)d\tau \tag{3.15}$$

other matrix parameters in discrete Kalman filter are simply the same with that in (3.10) and (3.11):

$$\mathbf{H}_k = \mathbf{H} \tag{3.16}$$

$$\mathbf{R}_k = \mathbf{R} \tag{3.17}$$

In conclusion, if system can be described by linear different equations (3.10) and the measurement equation is known as (3.11), this system can be discretized in the form of discrete Kalman filter in section 4.2.1 and the state is estimated according to (3.5)-(3.9).

4.3 Extended Kalman Filter

If the process to be estimated or the measurement relationship is non-linear, the extended Kalman filter is applied. The extended Kalman filter (EKF) is similar with the discrete Kalman filter. The only difference is that the system model is described by non-linear equations instead of linear ones. The process model and measurement model of the EKF are written as

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \boldsymbol{w}_{k-1}) \tag{3.18}$$

$$\mathbf{z}_k = h(\mathbf{x}_k, \boldsymbol{\upsilon}_k) \tag{3.19}$$

The non-linear equations can be expanded by Taylor series. By ignoring the second order and higher order terms, the nonlinear equations are linearized and it can be proceeded like linear Kalman filter. The EKF is not used in the thesis as the performance has been proved not as good as unscented Kalman filter, which is the

conclusion of [13, 18, 37, 38]. Consequently, the details of EKF is not presented here, see [7, 8] for more details about EKF.

4.4 Unscented Kalman Filter

The unscented Kalman filter (UKF) is a recursive minimum mean-square-error (MMSE) estimator for Gaussian random variables, based on the unscented transformation. The unscented transformation was proposed by Julier and Uhlmann [39] to compute statistical properties of random variables determined by nonlinear functions. The UKF does not linearize the nonlinear functions by using the first order term of the Taylor series expansion like EKF. Instead, the UKF uses the true nonlinear models and the propagation of a minimal set of deterministically selected sample points which capture the true mean and covariance of Gaussian random variables to capture the a posterior mean and covariance accurately to the 2^{nd} order of nonlinear functions. [13]

4.4.1 Unscented Transformation

The unscented transformation (UT) is a method to calculate the mean and covariance of a nonlinear function by transforming the deterministic sigma points through the nonlinear function. [13] Consider an arbitrary nonlinear function

$$y = f(x) \tag{3.20}$$

where x is a L dimensional random variable and its mean and covariance are $\bar{x}$ and P_x.

In order to calculate the mean and covariance of y, the $2L+1$ sigma points $S_i = \{w_i, \mathbf{X}_i\}$ are generated deterministically:

$$\begin{aligned}
\mathbf{X}_0 &= \bar{x} \qquad & w_0 &= \frac{\kappa}{L+\kappa} \qquad & i &= 0 \\
\mathbf{X}_i &= \bar{x} + \left(\sqrt{(L+\kappa)P_x}\right)_i \qquad & w_i &= \frac{1}{2(L+\kappa)} \qquad & i &= 1,\ldots,L \\
\mathbf{X}_i &= \bar{x} - \left(\sqrt{(L+\kappa)P_x}\right)_i \qquad & w_i &= \frac{1}{2(L+\kappa)} \qquad & i &= L+1,\ldots,2L
\end{aligned} \tag{3.21}$$

where w_i is the weight of i^{th} sigma point. κ is a scaling parameter and $\left(\sqrt{(L+\kappa)P_x}\right)_i$ is the i^{th} column/row of the square root of the weighted covariance matrix which is normally calculated by Cholesky factorization method [40] due to its good numerical efficiency.

All sigma points are propagated through the nonlinear function

$$\mathbf{Y}_i = f\left(\mathbf{X}_i\right) \quad i = 0,\ldots,2L \tag{3.22}$$

and the mean, covariance and cross covariance of y are calculated approximately:

$$\bar{y} \approx \sum_0^{2L} w_i \mathbf{Y}_i \tag{3.23}$$

$$P_y \approx \sum_0^{2L} w_i \left(\mathbf{Y}_i - \bar{y}\right)\left(\mathbf{Y}_i - \bar{y}\right)^T \tag{3.24}$$

$$P_{xy} \approx \sum_0^{2L} w_i \left(\mathbf{X}_i - \bar{x}\right)\left(\mathbf{Y}_i - \bar{y}\right)^T \tag{3.25}$$

These estimates of the mean and covariance are accurate to the second order (third order for true Gaussian priors) of the Taylor series expansion of $f(x)$ for any nonlinear function. [13] An example is shown here to illustrate the unscented transform:

$$y = f(x) \tag{3.26}$$

where $x = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T$, $y = \begin{bmatrix} |x_1|\left[\cos(x_2) + \sin(x_2)\right] \\ 2|x_1|\sin(x_2) \end{bmatrix}$ and the statistic property of x is known as $\bar{x} = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$ and $P_x = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}$.

10000 sample points of x is generated and their mean and covariance ellipse are shown in Figure 4-2 then these points are propagated through nonlinear function (3.26) and the points of y are shown Figure 4-3. As the number of y points is very large, the mean and covariance of y which are calculated by these points is considered as true value. Thus the true mean of y and its covariance ellipse are plotted in Figure 4-3 where the means and covariance ellipses derived from EKF and unscented transformation are compared. As a result, the UT almost captures the mean and covariance of y accurately while EKF results in larger errors.

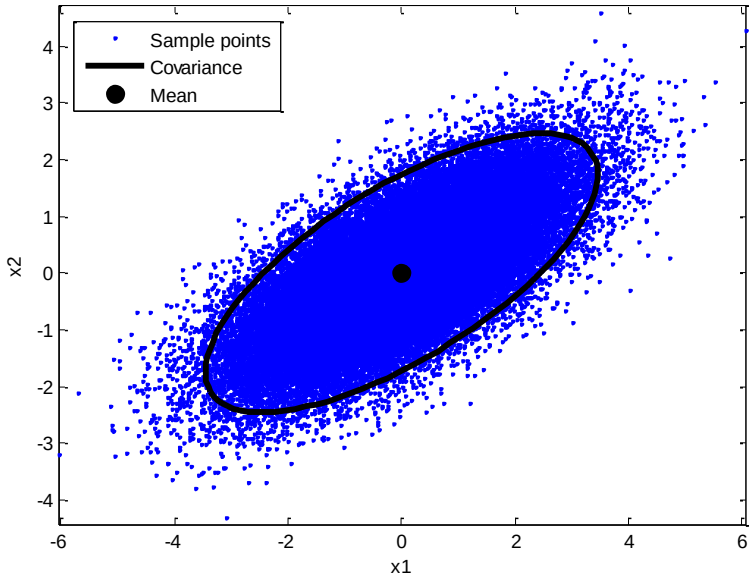


Figure 4-2: Mean and covariance of sample points

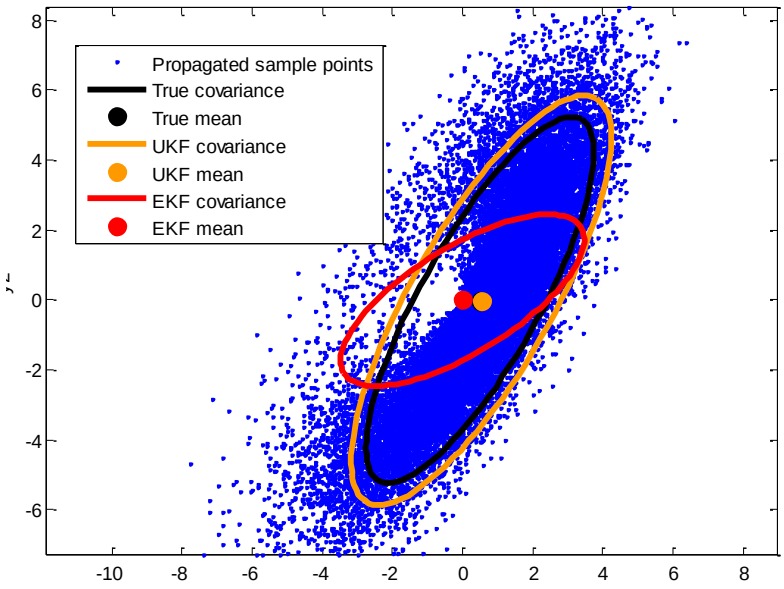


Figure 4-3: Mean and covariance propagation via nonlinear function by means of EKF and UKF

4.4.2 Unscented Kalman Filter Algorithm

The unscented Kalman filter algorithm is implemented as the following steps: [14]

1. Initialization:

$$\hat{\mathbf{x}}_0 = E\left[\mathbf{x}_0\right], \quad \mathbf{P}_0 = E\left[\left(\mathbf{x}_0 - \hat{\mathbf{x}}_0\right)\left(\mathbf{x}_0 - \hat{\mathbf{x}}_0\right)^T\right] \tag{3.27}$$

2. Calculation of sigma points:

   For $k \in \{1, \ldots, \infty\}$,

$$\mathbf{X}_{k-1} = \left[\hat{\mathbf{x}}_{k-1} \quad \hat{\mathbf{x}}_{k-1} + \gamma\sqrt{\mathbf{P}_{k-1}} \quad \hat{\mathbf{x}}_{k-1} - \gamma\sqrt{\mathbf{P}_{k-1}}\right] \tag{3.28}$$

3. Time-update equations:

$$\mathbf{X}_{k|k-1} = f\left(\mathbf{X}_{k-1}, \mathbf{u}_{k-1}\right) \tag{3.29}$$

$$\hat{\mathbf{x}}_k^- = \sum_{i=0}^{2L} w_i^{(m)} \mathbf{X}_{i,k|k-1} \tag{3.30}$$

$$\mathbf{P}_{\mathbf{x}_k}^- = \sum_{i=0}^{2L} w_i^{(c)} \left[\mathbf{X}_{i,k|k-1} - \hat{\mathbf{x}}_k^-\right]\left[\mathbf{X}_{i,k|k-1} - \hat{\mathbf{x}}_k^-\right]^T + \mathbf{R}^v \tag{3.31}$$

4. Measurement-update equations

$$\mathbf{Y}_{k|k-1} = \mathbf{H}\left[\mathbf{X}_{k|k-1}\right] \tag{3.32}$$

$$\hat{\mathbf{y}}_k^- = \sum_{i=0}^{2L} w_i^{(m)} \mathbf{Y}_{i,k|k-1} \tag{3.33}$$

$$\mathbf{P}_{\tilde{\mathbf{y}}_k\tilde{\mathbf{y}}_k} = \sum_{i=0}^{2L} w_i^{(c)} \left[\mathbf{Y}_{i,k|k-1} - \hat{\mathbf{y}}_k^-\right]\left[\mathbf{Y}_{i,k|k-1} - \hat{\mathbf{y}}_k^-\right]^T + \mathbf{R}^n \tag{3.34}$$

$$\mathbf{P}_{\mathbf{x}_k\mathbf{y}_k} = \sum_{i=0}^{2L} w_i^{(c)} \left[\mathbf{X}_{i,k|k-1} - \hat{\mathbf{x}}_k^-\right]\left[\mathbf{Y}_{i,k|k-1} - \hat{\mathbf{y}}_k^-\right]^T \tag{3.35}$$

$$\mathbf{K}_k = \mathbf{P}_{\mathbf{x}_k\mathbf{y}_k}\mathbf{P}^{-1}_{\tilde{\mathbf{y}}_k\tilde{\mathbf{y}}_k} \tag{3.36}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k\left(\tilde{\mathbf{y}}_k - \hat{\mathbf{y}}_k^-\right) \tag{3.37}$$

$$\mathbf{P}_{\mathbf{x}_k} = \mathbf{P}_{\mathbf{x}_k}^- - \mathbf{K}_k\mathbf{P}_{\tilde{\mathbf{y}}_k\tilde{\mathbf{y}}_k}\mathbf{K}_k^T \tag{3.38}$$

where $\lambda = \alpha^2\left(L+\kappa\right) - L$ and $\gamma = \sqrt{L+\lambda}$, α is the new sigma-point scaling parameter, which controls the size of the sigma-point distribution and should ideally be a small number to avoid sampling non-local effects when the nonlinearities are strong. [14] $\mathbf{R}^v$ and $\mathbf{R}^n$ are the process noise covariance and measurement noise covariance respectively.

w_i is calculated by

$$w_0^{(m)} = \frac{\lambda}{L+\lambda} \quad i = 0 \tag{3.39}$$

$$w_0^{(c)} = \frac{\lambda}{L+\lambda} + \left(1 - \alpha^2 + \beta\right) \quad i = 0 \tag{3.40}$$

$$w_i^{(m)} = w_i^{(c)} = \frac{\lambda}{2\left(L+\lambda\right)} \quad i = 1,\ldots,2L \tag{3.41}$$

where β is the parameter which affects the weight of the 0^{th} sigma point for the covariance.

There is a major flaw in UKF: the state error covariance matrix $\mathbf{P}$ might lose positive definiteness in the update step during UKF funning due to numerical reasons, resulting

in the failure of Cholesky factorization for the computation of sigma points. To resolve this problem, the eigenvalues of **P** should be checked and modified to keep this algorithm working properly. [18]

4.4.3 Square-root Unscented Kalman Filter

An alternative for UKF is square-root unscented Kalman filter (SRUKF) which overcomes the drawback of UKF in numerical calculation by avoiding the use of Cholesky factorization. The algorithms of UKF and SRUKF are quite similar except the method for determining the sigma points. Unlike the Cholesky factorization [41] in UKF, the QR decomposition [42] is used in the SRUKF which has improved numerical properties and robustness than the UKF. [18] The details of the SRUKF is described as following: [14]

1. Initialization:

$$\hat{\mathbf{x}}_0 = E[\mathbf{x}_0], \quad \mathbf{S}_0 = chol\left\{E\left[(\mathbf{x}_0 - \hat{\mathbf{x}}_0)(\mathbf{x}_0 - \hat{\mathbf{x}}_0)^T\right]\right\} \tag{3.42}$$

2. Calculation of sigma points:

   For $k \in \{1, \ldots, \infty\}$,

$$\mathbf{X}_{k-1} = \left[\hat{\mathbf{x}}_{k-1} \quad \hat{\mathbf{x}}_{k-1} + \gamma \mathbf{S}_k \quad \hat{\mathbf{x}}_{k-1} - \gamma \mathbf{S}_k\right] \tag{3.43}$$

3. Time-update equations:

$$\mathbf{X}_{k|k-1} = f\left(\mathbf{X}_{k-1}, \mathbf{u}_{k-1}\right) \tag{3.44}$$

$$\hat{\mathbf{x}}_k^- = \sum_{i=0}^{2L} w_i^{(m)} \mathbf{X}_{i,k|k-1} \tag{3.45}$$

$$\mathbf{S}_k^- = qr\left\{\left[\sqrt{w_i^{(c)}}\left(\mathbf{X}_{1:2L,k|k-1} - \hat{\mathbf{x}}_k^-\right) \quad \mathbf{R}^v\right]\right\} \tag{3.46}$$

$$\mathbf{S}_k^- = cholupdate\left\{\mathbf{S}_k^- \quad \mathbf{X}_{0,k} - \hat{\mathbf{x}}_k^- \quad w_0^{(c)}\right\} \tag{3.47}$$

4. Measurement-update equations

$$\mathbf{Y}_{k|k-1} = \mathbf{H}\left[\mathbf{X}_{k|k-1}\right] \tag{3.48}$$

$$\hat{\mathbf{y}}_k^- = \sum_{i=0}^{2L} w_i^{(m)} \mathbf{Y}_{i,k|k-1} \tag{3.49}$$

$$\mathbf{S}_{\tilde{\mathbf{y}}_k} = qr\left\{\left[\sqrt{w_1^{(c)}}\left(\mathbf{Y}_{1:2L,k|k-1} - \hat{\mathbf{y}}_k^-\right) \quad \sqrt{\mathbf{R}^n}\right]\right\} \tag{3.50}$$

$$\mathbf{S}_{\tilde{\mathbf{y}}_k} = cholupdate\left\{\mathbf{S}_{\tilde{\mathbf{y}}_k} \quad \mathbf{Y}_{0,k} - \hat{\mathbf{y}}_k^- \quad w_0^{(c)}\right\} \tag{3.51}$$

$$\mathbf{P}_{\mathbf{x}_k\mathbf{y}_k} = \sum_{i=0}^{2L} w_i^{(c)}\left[\mathbf{X}_{i,k|k-1} - \hat{\mathbf{x}}_k^-\right]\left[\mathbf{Y}_{i,k|k-1} - \hat{\mathbf{y}}_k^-\right]^T \tag{3.52}$$

$$\mathbf{K}_k = \left(\mathbf{P}_{\mathbf{x}_k\mathbf{y}_k} / \mathbf{S}_{\tilde{\mathbf{y}}_k}^T\right) / \mathbf{S}_{\tilde{\mathbf{y}}_k} \tag{3.53}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k\left(\tilde{\mathbf{y}}_k - \hat{\mathbf{y}}_k^-\right) \tag{3.54}$$

$$\mathbf{U} = \mathbf{K}_k \mathbf{S}_{\tilde{\mathbf{y}}_k} \tag{3.55}$$

$$\mathbf{S}_k = cholupdate\left\{\mathbf{S}_k^- \quad \mathbf{U} \quad -1\right\} \tag{3.56}$$

4.5 Comparison of Different Filters

Cascade of notch filter and low pass filter can filter out the high frequency signal and output the low frequency signal. However, the output has a phase delay that may cause oscillation in ship movement during dynamic positioning. Kalman filter offers several advantages over notch filter: [43] the Kalman filter eliminates only the harmonic noise without distorting the signal whereas the notch filter removes both the harmonic noise and the components of the signal falling in reject region; the Kalman filter is able to follow the frequency changes without a priori information whereas the changes of noise frequency results in bad performance of the notch filter.

For nonlinear problems, the UKF and EKF are considered because standard KF is only valid in linear cases. The UKF does not require the computation of the partial derivatives of the measurement transition matrix **H**, compared with EKF. However, its computational load is still higher than EKF due to the calculations for each sigma point. [44] As the UKF is accurate to the 2^{nd} order of nonlinear functions while the EKF is linearized at 1^{st} order, the accuracy of UKF is higher than that of EKF. [13] The SRUKF with the same accuracy of the UKF is an improved version of the UKF, aimed on increasing the numerical performance in practice.

4.6 Conclusion

As accuracy is an important factor to be considered in DP system and the computation load can be well handled in normal PC, the UKF is superior to the EKF. This thesis will use the UKF (SRUKF) algorithm to solve nonlinear estimation problem instead of the EKF whilst the KF will be implemented to solve linear problem in this thesis.

5. Sensor Fusion

Sensor fusion is a process to combine measurements from a number of sensors of same or different types and estimate the state of interest. It is very important in multi-sensor system as utilizing all available measurements from sensors will give an improved result compared with using single sensor. [4] Moreover, a system fusing sensors has a higher error tolerance capacity since multi-sensors contains redundant measurements which increases the reliability of the system in sensor failure case. [45] In practice it is quite normal that there is wanted but unknown information which cannot be measured directly nor described by mathematical model perfectly. But there may exist other measurable information which is related to the wanted information by known relations. Then it becomes possible to estimate the information of interest by fusing the sensors. Multi-sensor fusion using Kalman filter algorithm was discussed in [4] and four different methods were described. Thereinto, the simplest way of state estimation for a multi-sensor system is to combine all measurements in one measurement model and it will be used in this chapter. In order to fuse DP reference sensors, two different approaches of sensor fusion are discussed in the thesis: approach 1 is to transform all sensor measurements to CG and then calculate the weighted means as input of unscented Kalman filter; approach 2 is to fuse all measurements directly in unscented Kalman filter. Each approach has its merit and demerit, which will be presented later. Furthermore, UKF with/without acceleration measurement and the least measurement modes of each UKF are discussed at the end of this chapter.

5.1 Approach 1 for Sensor Fusion

Approach 1 consists of three steps as shown in Figure 5-1. At first step all linear motion measurements are transformed to CG. Then the weighted means and variances of same variables are calculated at step 2 so that there is only one measurement for each type of

variables in UKF implemented at step 3 regardless of the number of sensors in each type. Therefore, the number of measurements has no influence on the complexity of UKF and the computation load of the UKF always stays the same.

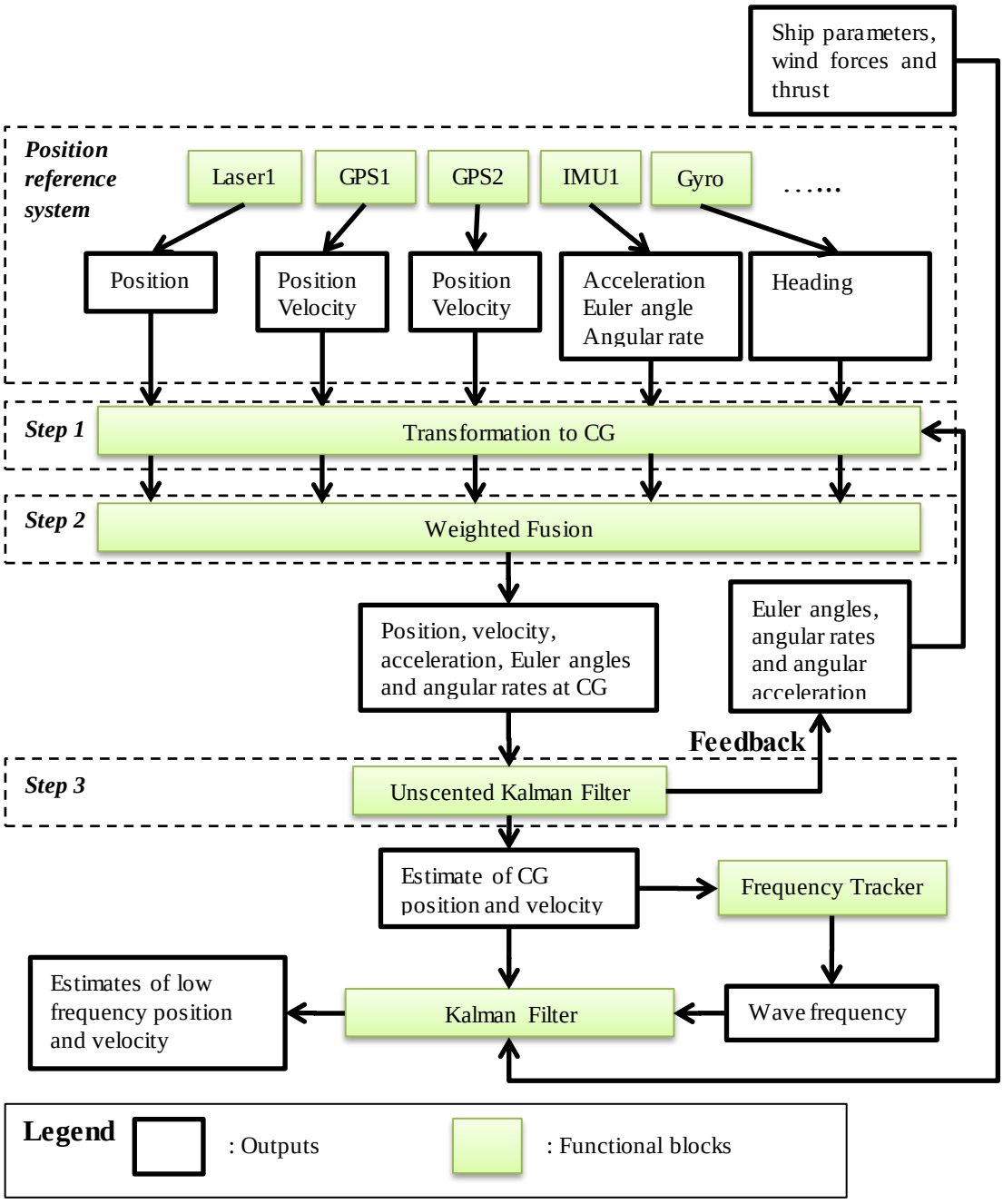


Figure 5-1: Diagram of Approach 1

5.1.1 Measurements Transformed to CG

Transformation between body-fixed frame and NED frame is described in section 2.4. The equation (2.13)(2.15) and (2.16) in single vector form can be rewritten in difference form of double vectors:

$$\mathbf{p}_{sensori|n} - \mathbf{p}_{CG|n} = \mathbf{R}(\mathbf{p}_{sensori|\mathrm{b}} - \mathbf{p}_{CG|\mathrm{b}}) \tag{4.1}$$

$$\mathbf{v}_{sensori|n} - \mathbf{v}_{CG|n} = \dot{\mathbf{R}}(\mathbf{p}_{sensori|\mathrm{b}} - \mathbf{p}_{CG|b}) \tag{4.2}$$

$$\mathbf{a}_{sensori|n} - \mathbf{a}_{CG|n} = \ddot{\mathbf{R}}(\mathbf{p}_{sensori|\mathrm{b}} - \mathbf{p}_{CG|\mathrm{b}}) \tag{4.3}$$

The notation for vectors is defined:

$\mathbf{p} \in \Re^{3\times1}$, $\mathbf{v} \in \Re^{3\times1}$ and $\mathbf{a} \in \Re^{3\times1}$ are the position vector, velocity vector and acceleration vector respectively, and their subscript *n* and *b* denote NED frame and body-fixed frame. For example, $\mathbf{p}_{CG|n} \in \Re^{3\times1}$, $\mathbf{v}_{CG|n} \in \Re^{3\times1}$ and $\mathbf{a}_{CG|n} \in \Re^{3\times1}$ are the position vector, velocity vector and acceleration vector of CG in NED frame respectively; Other variables are defined in the same way. $\mathbf{R} \in \Re^{3\times1}$ rotation transition matrix in section 2.4, and it is a function of $\mathbf{\Theta}_{E|n}$.

Then Linear motions of sensor *i* can be transformed to CG by using:

$$\mathbf{p}_{CG|n} = \mathbf{p}_{sensori|n} - \mathbf{R}(\mathbf{p}_{sensori|\mathrm{b}} - \mathbf{p}_{CG|\mathrm{b}}) \tag{4.4}$$

$$\mathbf{v}_{CG|n} = \mathbf{v}_{sensori|n} - \dot{\mathbf{R}}(\mathbf{p}_{sensori|\mathrm{b}} - \mathbf{p}_{CG|\mathrm{b}}) \tag{4.5}$$

$$\mathbf{a}_{CG|n} = \mathbf{a}_{sensori|n} - \ddot{\mathbf{R}}(\mathbf{p}_{sensori|\mathrm{b}} - \mathbf{p}_{CG|\mathrm{b}}) \tag{4.6}$$

The angular motions of sensor *i* are identical to that of any other points on the vessel which is assumed a rigid body. Hence, angular measurements can be used directly without transformation.

5.1.2 Inverse Variance Weighting

If there are *k* reference sensors to measure linear motions, *k* transformed measurements for CG can be obtained. Normally the measurements are redundant, so they can be fused to get a better estimation. Assume that $z_1, \ldots, z_k$ are independent measurements and have respective known variances $\sigma_1^2, \ldots, \sigma_k^2$, then the inverse variance weights is denoted by [46]

$$W_i = \frac{\frac{1}{\sigma_i^2}}{\sum_{j=1}^{j=k} \frac{1}{\sigma_j^2}} \tag{4.7}$$

The weighted mean $\hat{z}_{wm}$ is

$$\hat{z}_{wm} = \sum_{i=1}^{i=k} W_i z_i \tag{4.8}$$

The variance of the weighted mean $\hat{z}_{wm}$ is given by

$$\sigma_{wm}^2 = \frac{1}{\sum_{i=1}^{i=k} \frac{1}{\sigma_i^2}} \tag{4.9}$$

5.1.3 Sensor Fusion by Inverse Variance Weighting

According to (4.7)-(4.9) and (4.4) the mean CG position measurement at NED frame $\tilde{\mathbf{p}}_{CG|n}$ and its noise variance $\sigma^2_{\tilde{\mathbf{p}}_{CG|n}}$ can be written as

$$\tilde{\mathbf{p}}_{CG|n} = \sum_{i=1}^{i=k} \left\{ \frac{\frac{1}{\sigma^2_{\tilde{\mathbf{p}}_{sensori|n}}}}{\sum_{j=1}^{j=k} \frac{1}{\sigma^2_{\tilde{\mathbf{p}}_{sensorj|n}}}} \left[\tilde{\mathbf{p}}_{sensori|n} - \mathbf{R}\left(\mathbf{p}_{sensori|\mathrm{b}} - \mathbf{p}_{CG|\mathrm{b}} \right) \right] \right\} \tag{4.10}$$

$$\sigma^2_{\tilde{\mathbf{p}}_{CG|n}} = \frac{1}{\sum_{j=1}^{j=k} \frac{1}{\sigma^2_{\tilde{\mathbf{p}}_{sensorj|n}}}} \tag{4.11}$$

Similarly, the mean $\tilde{\mathbf{v}}_{CG|n}$ and noise variance $\sigma^2_{\tilde{\mathbf{v}}_{CG|n}}$ of velocity measurements at CG are given as

$$\tilde{\mathbf{v}}_{CG|n} = \sum_{i=1}^{i=k} \left\{ \frac{\frac{1}{\sigma^2_{\tilde{\mathbf{v}}_{sensori|n}}}}{\sum_{j=1}^{j=k} \frac{1}{\sigma^2_{\tilde{\mathbf{v}}_{sensorj|n}}}} \left[\tilde{\mathbf{v}}_{sensori|n} - \dot{\mathbf{R}}\left(\mathbf{p}_{sensori|\mathrm{b}} - \mathbf{p}_{CG|\mathrm{b}} \right) \right] \right\} \tag{4.12}$$

$$\sigma^2_{\tilde{\mathbf{v}}_{CG|n}} = \frac{1}{\sum_{j=1}^{j=k} \frac{1}{\sigma^2_{\tilde{\mathbf{v}}_{sensorj|n}}}} \tag{4.13}$$

And the mean $\tilde{\mathbf{a}}_{CG|\mathrm{b}}$ and noise variance $\sigma^2_{\tilde{\mathbf{a}}_{CG|\mathrm{b}}}$ of velocity measurements at CG are

$$\tilde{\mathbf{a}}_{CG|\mathrm{b}} = \sum_{i=1}^{i=k} \left\{ \frac{\dfrac{1}{\sigma^2_{\tilde{\mathbf{a}}_{sensori|n}}}}{\sum_{j=1}^{j=k} \dfrac{1}{\sigma^2_{\tilde{\mathbf{a}}_{sensorj|n}}}} \left[\tilde{\mathbf{a}}_{sensori|\mathrm{b}} - \mathbf{R}^T \ddot{\mathbf{R}} \left(\mathbf{p}_{sensori|\mathrm{b}} - \mathbf{p}_{CG|\mathrm{b}} \right) \right] \right\} \tag{4.14}$$

$$\sigma^2_{\tilde{\mathbf{a}}_{CG|\mathrm{b}}} = \frac{1}{\sum_{j=1}^{j=k} \dfrac{1}{\sigma^2_{\tilde{\mathbf{a}}_{sensorj|n}}}} \tag{4.15}$$

The transition matrix $\mathbf{R}$, $\dot{\mathbf{R}}$ and $\ddot{\mathbf{R}}$ are the functions of $\boldsymbol{\Theta}_{E|n}$, $\boldsymbol{\Omega}_{E|n}$ and $\mathbf{A}_{E|n}$ which are the feedbacks from unscented Kalman filter in Figure 5-1. The mean and variance of angular measurements are also derived:

$$\tilde{\boldsymbol{\Theta}}_{IMU} = \sum_{i=1}^{i=k} \left(\frac{\dfrac{1}{\sigma^2_{\tilde{\boldsymbol{\Theta}}_{IMUi}}}}{\sum_{j=1}^{j=k} \dfrac{1}{\sigma^2_{\tilde{\boldsymbol{\Theta}}_{IMUj}}}} \tilde{\boldsymbol{\Theta}}_{IMUi} \right) \tag{4.16}$$

$$\sigma^2_{\tilde{\boldsymbol{\Theta}}_{IMU}} = \frac{1}{\sum_{j=1}^{j=k} \dfrac{1}{\sigma^2_{\tilde{\boldsymbol{\Theta}}_{IMUj}}}} \tag{4.17}$$

$$\tilde{\boldsymbol{\Omega}}_{IMU} = \sum_{i=1}^{i=k} \left(\frac{\dfrac{1}{\sigma^2_{\tilde{\boldsymbol{\Omega}}_{IMUi}}}}{\sum_{j=1}^{j=k} \dfrac{1}{\sigma^2_{\tilde{\boldsymbol{\Omega}}_{IMUj}}}} \tilde{\boldsymbol{\Omega}}_{IMUi} \right) \tag{4.18}$$

$$\sigma^2_{\tilde{\boldsymbol{\Omega}}_{IMU}} = \frac{1}{\sum_{j=1}^{j=k} \dfrac{1}{\sigma^2_{\tilde{\boldsymbol{\Omega}}_{IMUj}}}} \tag{4.19}$$

where $\tilde{\boldsymbol{\Theta}}_{IMU} \in \Re^{3\times1}$ is the mean Euler angle measurements of IMU in NED frame, $\tilde{\boldsymbol{\Omega}}_{IMU} \in \Re^{3\times1}$ is the mean Euler angular rate measurements of IMU in NED frame. Notice that yaw measurement in $\tilde{\boldsymbol{\Theta}}_{IMU}$ is replaced by available measurement from gyroscope because gyro has a much higher accuracy. $\tilde{\boldsymbol{\Theta}}_{IMU}$ means roll, pitch measurement of IMU and yaw measurement of gyroscope instead of yaw from IMU. Note that $\tilde{\mathbf{a}}_{CG|\mathrm{b}}$ and $\tilde{\boldsymbol{\Omega}}_{IMU}$ contain the biases which is the inverse variance weighted mean of biases of corresponding sensors. Now multi-sensor measurements are transformed into the measurement vector:

$$\begin{bmatrix} \tilde{\mathbf{p}}_{CG|n} & \tilde{\mathbf{v}}_{CG|n} & \tilde{\mathbf{a}}_{CG|n} & \tilde{\boldsymbol{\Theta}}_{IMU} & \tilde{\boldsymbol{\Omega}}_{IMU} \end{bmatrix}^T \tag{4.20}$$

And (4.20) is the UKF (see Figure 5-1**)** measurement vector in (4.39).

5.1.4 Filter Design for Approach 1

The linear motion and rotational motion of a rigid body are given by kinematics

$$\dot{\mathbf{p}}_{CG|n} = \mathbf{v}_{CG|n} \tag{4.21}$$

$$\dot{\mathbf{v}}_{CG|n} = \mathbf{a}_{CG|n} \tag{4.22}$$

$$\dot{\boldsymbol{\Theta}}_{E|n} = \boldsymbol{\Omega}_{E|n} \tag{4.23}$$

$$\dot{\boldsymbol{\Omega}}_{E|n} = \mathbf{A}_{E|n} \tag{4.24}$$

However, the value of $\dot{\mathbf{a}}_{CG|n}$ is beyond knowledge and it is assumed to be white noise

$$\dot{\mathbf{a}}_{CG|n} = \boldsymbol{w}_{\mathbf{a}_{CG|n}} \tag{4.25}$$

Similarly,

$$\dot{\mathbf{A}}_{E|n} = \boldsymbol{w}_{\mathbf{A}_{CG|n}} \tag{4.26}$$

The bias of acceleration measurement $\dot{\mathbf{b}}_{\mathbf{a},IMU}$ and angular rate measurement $\dot{\mathbf{b}}_{\mathbf{\Omega},IMU}$ are both slowly varying terms and their time derivatives are approximately zero, so they are also assumed to be white noise:

$$\dot{\mathbf{b}}_{\tilde{\mathbf{a}}_{IMU}} = \boldsymbol{w}_{\mathbf{b}_{\tilde{\mathbf{a}}_{IMU}}} \tag{4.27}$$

$$\dot{\mathbf{b}}_{\mathbf{\Omega}_{IMU}} = \boldsymbol{w}_{\mathbf{b}_{\mathbf{\Omega}_{IMU}}} \tag{4.28}$$

Then the process model in matrix form is given by

$$\begin{bmatrix} \dot{\mathbf{p}}_{CG|n} \\ \dot{\mathbf{v}}_{CG|n} \\ \dot{\mathbf{a}}_{CG|n} \\ \dot{\mathbf{\Theta}}_{E|n} \\ \dot{\mathbf{\Omega}}_{E|n} \\ \dot{\mathbf{A}}_{E|n} \\ \dot{\mathbf{b}}_{\tilde{\mathbf{a}}_{IMU}} \\ \dot{\mathbf{b}}_{\tilde{\mathbf{\Omega}}_{IMU}} \end{bmatrix} = \begin{bmatrix} \mathbf{O}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} \end{bmatrix} \begin{bmatrix} \mathbf{p}_{CG|n} \\ \mathbf{v}_{CG|n} \\ \mathbf{a}_{CG|n} \\ \mathbf{\Theta}_{E|n} \\ \mathbf{\Omega}_{E|n} \\ \mathbf{A}_{E|n} \\ \mathbf{b}_{\tilde{\mathbf{a}}_{IMU}} \\ \mathbf{b}_{\tilde{\mathbf{\Omega}}_{IMU}} \end{bmatrix} + \begin{bmatrix} \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} \\ \boldsymbol{w}_{\mathbf{a}_{CG|n}} \\ \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} \\ \boldsymbol{w}_{\mathbf{A}_{CG|n}} \\ \boldsymbol{w}_{\mathbf{b}_{\tilde{\mathbf{a}}_{IMU}}} \\ \boldsymbol{w}_{\mathbf{b}_{\tilde{\mathbf{\Omega}}_{IMU}}} \end{bmatrix} \tag{4.29}$$

It is easier to implement a discrete form of UKF compared with the continuous form. In this case, $\mathbf{A}(T_s) = \mathrm{e}^{\mathbf{F}T_s}$ is exactly equal to $\mathbf{I} + \mathbf{F}T_s + \frac{(\mathbf{F}T_s)^2}{2!}$, so the differential equation (4.29) can be accurately discretized using (3.13) and (3.15) ,

$$\begin{bmatrix} \mathbf{p}_{CG|n,k} \\ \mathbf{v}_{CG|n,k} \\ \mathbf{a}_{CG|n,k} \\ \boldsymbol{\Theta}_{E|n,k} \\ \boldsymbol{\Omega}_{E|n,k} \\ \mathbf{A}_{E|n,k} \\ \mathbf{b}_{\tilde{\mathbf{a}}_{IMU},k} \\ \mathbf{b}_{\mathbf{b}_{\tilde{\Omega}_{IMU}},k} \end{bmatrix} = \begin{bmatrix} \mathbf{p}_{CG|n,k-1} + \mathbf{v}_{CG|n,k-1}t + \frac{1}{2}\mathbf{a}_{CG|b,k-1}t^2 \\ \mathbf{v}_{CG|n,k-1} + \mathbf{a}_{CG|b,k-1}t \\ \mathbf{a}_{CG|b,k-1} \\ \boldsymbol{\Theta}_{E|n,k-1} + \boldsymbol{\Omega}_{E|n,k-1}t + \frac{1}{2}\mathbf{A}_{E|n,k-1}t^2 \\ \boldsymbol{\Omega}_{E|n,k-1} + \mathbf{A}_{E|n,k-1}t \\ \mathbf{A}_{E|n,k-1} \\ \mathbf{b}_{\tilde{\mathbf{a}}_{IMU},k-1} \\ \mathbf{b}_{\mathbf{b}_{\tilde{\Omega}_{IMU}},k-1} \end{bmatrix} + \mathbf{Q}_k \tag{4.30}$$

$\mathbf{Q}_k \in \Re^{24\times 24}$ in (4.30) is defined as the diagonal matrix with submatrix $\mathbf{Q}_{k1,1} \in \Re^{9\times 9}$, $\mathbf{Q}_{k2,2} \in \Re^{9\times 9}$ and $\mathbf{Q}_{k3,3} \in \Re^{6\times 6}$ on its diagonal and it is written as $\mathbf{Q}_k = diag\left(\left[\mathbf{Q}_{k1,1}, \mathbf{Q}_{k2,2}, \mathbf{Q}_{k3,3}\right]\right)$ where the submatrixes are

$$\mathbf{Q}_{k1,1} = diag\left(\boldsymbol{\sigma}^2_{\boldsymbol{w}_{\mathbf{a}_{CG|n}}}\right) \cdot \begin{bmatrix} \frac{t^5}{20} & \frac{t^4}{8} & \frac{t^3}{6} \\ \frac{t^4}{8} & \frac{t^3}{3} & \frac{t^2}{2} \\ \frac{t^3}{6} & \frac{t^2}{2} & t \end{bmatrix} \tag{4.31}$$

$$\mathbf{Q}_{k2,2} = diag\left(\boldsymbol{\sigma}^2_{\boldsymbol{w}_{\mathbf{A}_{CG|n}}}\right) \cdot \begin{bmatrix} \frac{t^5}{20} & \frac{t^4}{8} & \frac{t^3}{6} \\ \frac{t^4}{8} & \frac{t^3}{3} & \frac{t^2}{2} \\ \frac{t^3}{6} & \frac{t^2}{2} & t \end{bmatrix} \tag{4.32}$$

$$\mathbf{Q}_{k3,3} = diag\left(\left[\boldsymbol{\sigma}^2_{\boldsymbol{w}_{\mathbf{b}_{\mathbf{a}},\mathrm{IMU}}} \quad \boldsymbol{\sigma}^2_{\boldsymbol{w}_{\boldsymbol{\Omega},\mathrm{IMU}}}\right]\right) \cdot t \tag{4.33}$$

Note that (4.31) is short for

$$\mathbf{Q}_{k1,1}=\begin{bmatrix} \frac{t^5}{20}\cdot diag\left(\boldsymbol{\sigma}^2_{w_{\mathbf{a}_{CG|n}}}\right) & \frac{t^4}{8}\cdot diag\left(\boldsymbol{\sigma}^2_{w_{\mathbf{a}_{CG|n}}}\right) & \frac{t^3}{6}\cdot diag\left(\boldsymbol{\sigma}^2_{w_{\mathbf{a}_{CG|n}}}\right) \\ \frac{t^4}{8}\cdot diag\left(\boldsymbol{\sigma}^2_{w_{\mathbf{a}_{CG|n}}}\right) & \frac{t^3}{3}\cdot diag\left(\boldsymbol{\sigma}^2_{w_{\mathbf{a}_{CG|n}}}\right) & \frac{t^2}{2}\cdot diag\left(\boldsymbol{\sigma}^2_{w_{\mathbf{a}_{CG|n}}}\right) \\ \frac{t^3}{6}\cdot diag\left(\boldsymbol{\sigma}^2_{w_{\mathbf{a}_{CG|n}}}\right) & \frac{t^2}{2}\cdot diag\left(\boldsymbol{\sigma}^2_{w_{\mathbf{a}_{CG|n}}}\right) & t\cdot diag\left(\boldsymbol{\sigma}^2_{w_{\mathbf{a}_{CG|n}}}\right) \end{bmatrix}$$ and the same rule

applies to $\mathbf{Q}_{k2,2}$ and $\mathbf{Q}_{k3,3}$.

The equations for measurement model is given by

$$\tilde{\mathbf{p}}_{CG|n}=\mathbf{p}_{CG|n}+\boldsymbol{\upsilon}_{\tilde{\mathbf{p}}_{CG|n}} \tag{4.34}$$

$$\tilde{\mathbf{v}}_{CG|n}=\mathbf{v}_{CG|n}+\boldsymbol{\upsilon}_{\tilde{\mathbf{v}}_{CG|n}} \tag{4.35}$$

$$\tilde{\boldsymbol{\Theta}}_{IMU}=\boldsymbol{\Theta}_{E|n}+\boldsymbol{\upsilon}_{\tilde{\boldsymbol{\Theta}}_{IMU}} \tag{4.36}$$

$$\tilde{\boldsymbol{\Omega}}_{IMU}=\boldsymbol{\Omega}_{E|n}+\mathbf{b}_{\tilde{\boldsymbol{\Omega}}_{IMU}}+\boldsymbol{\upsilon}_{\tilde{\boldsymbol{\Omega}}_{IMU}} \tag{4.37}$$

According to (4.3) acceleration equation is expressed as

$$\tilde{\mathbf{a}}_{CG|\mathrm{b}}=\mathbf{R}^T(\boldsymbol{\Theta}_{E|n})\left(\mathbf{a}_{CG|n}-\mathbf{g}\right)+\mathbf{b}_{\tilde{\mathbf{a}}_{IMU}}+\boldsymbol{\upsilon}_{\tilde{\mathbf{a}}_{CG|\mathrm{b}}} \tag{4.38}$$

Then the measurement model is of the form

$$\begin{bmatrix} \tilde{\mathbf{p}}_{CG|n} \\ \tilde{\mathbf{v}}_{CG|n} \\ \tilde{\mathbf{a}}_{CG|b} \\ \tilde{\boldsymbol{\Theta}}_{IMU} \\ \tilde{\boldsymbol{\Omega}}_{IMU} \end{bmatrix}=\begin{bmatrix} \mathbf{p}_{CG|n} \\ \mathbf{v}_{CG|n} \\ \mathbf{R}^T(\boldsymbol{\Theta}_{E|n})\left(\mathbf{a}_{CG|n}-\mathbf{g}\right)+\mathbf{b}_{\tilde{\mathbf{a}}_{IMU}} \\ \boldsymbol{\Theta}_{E|n} \\ \boldsymbol{\Omega}_{E|n}+\mathbf{b}_{\tilde{\boldsymbol{\Omega}}_{IMU}} \end{bmatrix}+\begin{bmatrix} \boldsymbol{\upsilon}_{\tilde{\mathbf{p}}_{CG|n}} \\ \boldsymbol{\upsilon}_{\tilde{\mathbf{v}}_{CG|n}} \\ \boldsymbol{\upsilon}_{\tilde{\mathbf{a}}_{CG|b}} \\ \boldsymbol{\upsilon}_{\tilde{\boldsymbol{\Theta}}_{IMU}} \\ \boldsymbol{\upsilon}_{\tilde{\boldsymbol{\Omega}}_{IMU}} \end{bmatrix} \tag{4.39}$$

where $\mathbf{g} \in \Re^{3\times1}$ is the local gravity in NED frame and it is positive; $\upsilon_{\tilde{\mathbf{p}}_{CG|n}}$, $\upsilon_{\tilde{\mathbf{v}}_{CG|n}}$, $\upsilon_{\tilde{\mathbf{a}}_{CG|b}}$, $\upsilon_{\tilde{\Theta}_{IMU}}$ and $\upsilon_{\tilde{\Omega}_{IMU}} \in \Re^{3\times1}$ are the sensor noises. All the measurements in (4.39) comes from (4.20) which is the result of weighted fusion block in Figure 5-1. The measurement model (4.39) are nonlinear, so unscented Kalman filter is implemented to estimate the state.

5.1.5 Approach 1 without Acceleration Measurement

Acceleration is not a must to determine the position, velocity and heading. It may be used to increase the accuracy of the estimate, whereas, is it very necessary to use acceleration measurement by adding more computation load and complexity on the system.

The UKF model (4.29) (4.39) and can be simplified by eliminating the acceleration term and its relevant term. The new process model and measurement model is respectively expressed as

$$\begin{bmatrix} \dot{\mathbf{p}}_{CG|n} \\ \dot{\mathbf{v}}_{CG|n} \\ \dot{\Theta}_{E|n} \\ \dot{\Omega}_{E|n} \\ \dot{\mathbf{b}}_{\tilde{\Omega}_{IMU}} \end{bmatrix} = \begin{bmatrix} \mathbf{O}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} \end{bmatrix} \begin{bmatrix} \mathbf{p}_{CG|n} \\ \mathbf{v}_{CG|n} \\ \Theta_{E|n} \\ \Omega_{E|n} \\ \mathbf{b}_{\tilde{\Omega}_{IMU}} \end{bmatrix} + \begin{bmatrix} \mathbf{O}_{3\times3} \\ w_{\mathbf{v}_{CG|n}} \\ \mathbf{O}_{3\times3} \\ w_{\Omega_{E|n}} \\ w_{\mathbf{b}_{\tilde{\Omega}_{IMU}}} \end{bmatrix} \tag{4.40}$$

$$\begin{bmatrix} \tilde{\mathbf{p}}_{CG|n} \\ \tilde{\mathbf{v}}_{CG|n} \\ \tilde{\Theta}_{IMU} \\ \tilde{\Omega}_{IMU} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{I}_{3\times3} \end{bmatrix} \begin{bmatrix} \mathbf{p}_{CG|n} \\ \mathbf{v}_{CG|n} \\ \Theta_{E|n} \\ \Omega_{E|n} \\ \mathbf{b}_{\tilde{\Omega}_{IMU}} \end{bmatrix} + \begin{bmatrix} \upsilon_{\tilde{\mathbf{p}}_{CG|n}} \\ \upsilon_{\tilde{\mathbf{v}}_{CG|n}} \\ \upsilon_{\tilde{\Theta}_{IMU}} \\ \upsilon_{\tilde{\Omega}_{IMU}} \end{bmatrix} \tag{4.41}$$

5.1.6 Least Measurement Mode

A failed sensor of one measurement type will be kicked out from weighted fusion block in Figure 5-1 and the UKF is still running properly in most cases except that there is no position measurement or no angular measurement. In approach 1 position measurement and angular measurement must be combined to determine the state of the UKF and the filter cannot work if one of the information is missing. The least measurements required are one position measurement and one yaw measurement. In least measurement mode, the measurement model for the UKF is

$$\begin{bmatrix} \tilde{\mathbf{p}}_{CG|n} \\ \tilde{\boldsymbol{\Theta}}_{IMU} \end{bmatrix} = \begin{bmatrix} \mathbf{p}_{CG|n} \\ \boldsymbol{\Theta}_{E|n} \end{bmatrix} + \begin{bmatrix} \upsilon_{\tilde{\mathbf{p}}_{CG|n}} \\ \upsilon_{\tilde{\Theta}_{IMU}} \end{bmatrix} \tag{4.42}$$

where $\tilde{\boldsymbol{\Theta}}_{IMU} = [0, 0, \psi]^T$ and ψ is the yaw measurement while pitch and roll measurements are not available. Then the measurement error covariance of $\tilde{\boldsymbol{\Theta}}_{IMU}$ corresponding to roll and pitch is set to a larger value than normal because they are assumed to be zero which is not accurate. If the least measurement mode cannot be satisfied, the UKF is not able to estimate the state vector. Then the Kalman filter in Figure 5-1 will do dead reckoning while the state of the UKF will keep unchanged until enough measurements are recovered to determine the state of the filter.

5.2 Approach 2 for Sensor Fusion

Unlike approach 1, all measurements of reference sensors are output to the unscented Kalman filter in approach 2 (see Figure 5-2), which means the size of UKF measurement vector of approach 2 is much larger compared with that of approach 1. All measurements are fused simultaneously in the UKF. Under this circumstance, the measurement model is variational other than invariable since the number of available sensors is probably changing due to sensor failure during DP operation. The loss and recovery of sensors should be seamless in order to increase reliability of the system, which requires feasible countermeasures to be done in UKF algorithm. The process model of UKF here is the same with that in approach 1.

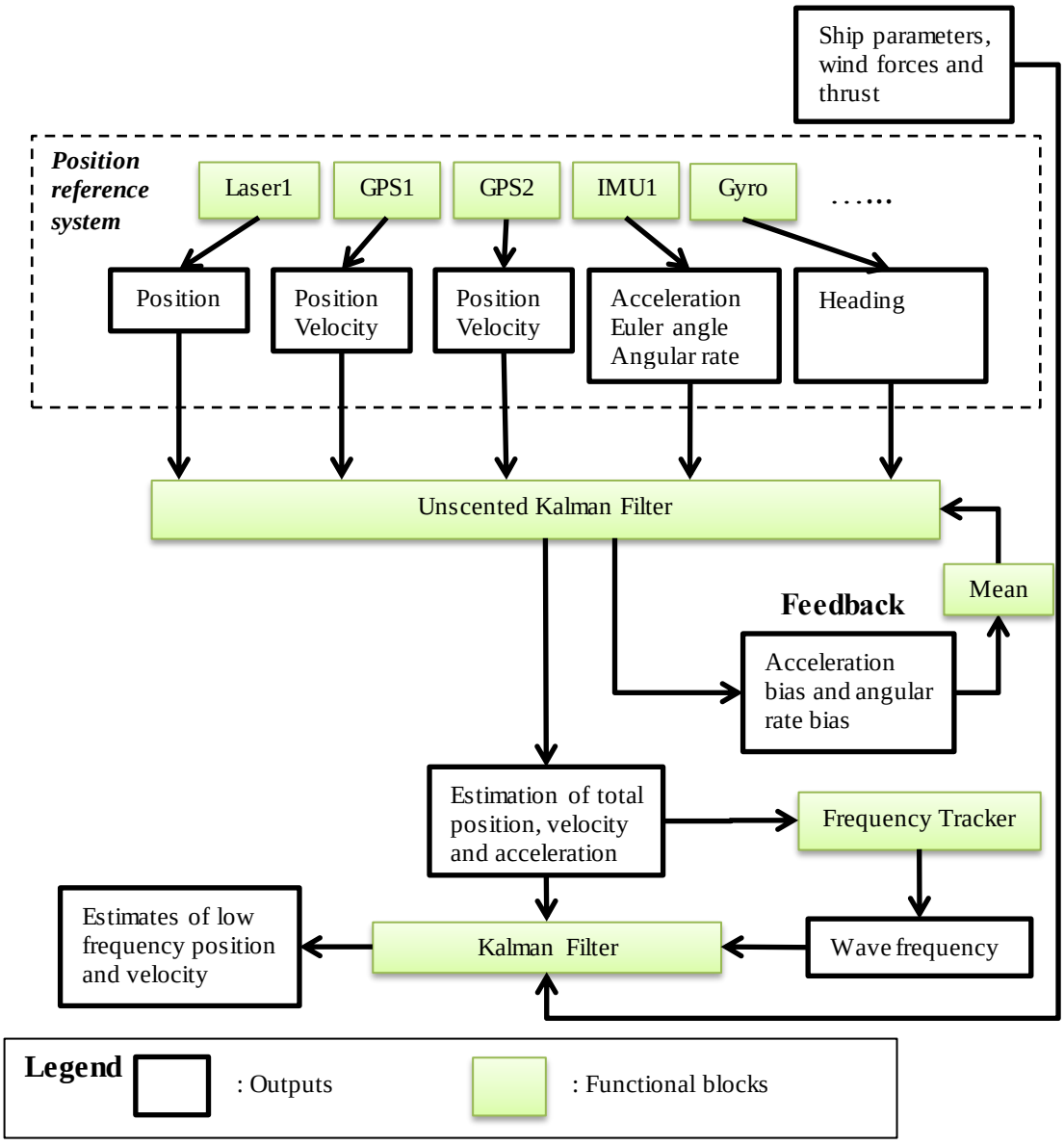


Figure 5-2: Diagram of Approach 2

5.2.1 Sensor Fusion Using Unscented Kalman Filter

The process model of approach 2 is also described by (4.29). However the measurement model is different from (4.39). Kinematics relates all the available measurements to state variables by equation (4.4)-(4.6). For position, velocity and acceleration measurements, the measurement equations are given by

$$\tilde{\mathbf{p}}_{GPSi} = \mathbf{p}_{CG|n} + \mathbf{R}(\boldsymbol{\Theta}_{E|n})(\mathbf{p}_{GPSi|b} - \mathbf{p}_{CG|b}) + \boldsymbol{\upsilon}_{\tilde{\mathbf{p}}_{GPSi}} \tag{4.43}$$

$$\tilde{\mathbf{v}}_{GPSi} = \mathbf{v}_{CG|n} + \dot{\mathbf{R}}(\boldsymbol{\Theta}_{E|n}, \boldsymbol{\Omega}_{E|n})(\mathbf{p}_{GPSi|b} - \mathbf{p}_{CG|b}) + \boldsymbol{\upsilon}_{\tilde{\mathbf{v}}_{GPSi}} \tag{4.44}$$

$$\tilde{\mathbf{p}}_{Ohteri} = \mathbf{p}_{CG|n} + \mathbf{R}(\boldsymbol{\Theta}_{E|n})(\mathbf{p}_{Otheri|b} - \mathbf{p}_{CG|b}) + \boldsymbol{\upsilon}_{\tilde{\mathbf{p}}_{Ohteri}} \tag{4.45}$$

$$\tilde{\mathbf{a}}_{IMUi} = \mathbf{R}^T(\boldsymbol{\Theta}_{E|n})\left[\mathbf{a}_{CG|n} - \mathbf{g} + \ddot{\mathbf{R}}(\boldsymbol{\Theta}_{E|n}, \boldsymbol{\Omega}_{E|n}, \mathbf{A}_{E|n})(\mathbf{p}_{\mathrm{IMU}\,i|b} - \mathbf{p}_{CG|b})\right] + \mathbf{b}_{\tilde{\mathbf{a}}_{IMUi}} + \boldsymbol{\upsilon}_{\tilde{\mathbf{a}}_{IMUi}} \tag{4.46}$$

where $\tilde{\mathbf{p}}_{GPSi} \in \Re^{3\times1}$ is the position measurements of GPS *i*; $\tilde{\mathbf{v}}_{GPSi} \in \Re^{3\times1}$ is the velocity measurements of GPS *i*; $\tilde{\mathbf{p}}_{Otheri} \in \Re^{3\times1}$ is the position measurements of other position sensor *i* except GPS; $\mathbf{R}(\boldsymbol{\Theta}_{E|n})$ is identical to the rotation transition matrix $\mathbf{R}$ in section 2.4; $\boldsymbol{\upsilon}_{\tilde{\mathbf{p}}_{GPSi}}$, $\boldsymbol{\upsilon}_{\tilde{\mathbf{v}}_{GPSi}}$, $\boldsymbol{\upsilon}_{\tilde{\mathbf{a}}_{IMUi}}$ and $\boldsymbol{\upsilon}_{\tilde{\mathbf{p}}_{Ohteri}} \in \Re^{3\times1}$ are the corresponding measurement noises respectively; $\mathbf{b}_{\tilde{\mathbf{a}}_{IMUi}} \in \Re^{3\times1}$ is the bias of IMU *i* acceleration measurements.

The measurement equations for rotational motion can be written directly:

$$\tilde{\boldsymbol{\Theta}}_{IMUi} = \boldsymbol{\Theta}_{E|n} + \boldsymbol{\upsilon}_{\tilde{\boldsymbol{\Theta}}_{IMUi}} \tag{4.47}$$

$$\tilde{\boldsymbol{\Omega}}_{IMUi} = \boldsymbol{\Omega}_{E|n} + \mathbf{b}_{\tilde{\boldsymbol{\Omega}}_{IMUi}} + \boldsymbol{\upsilon}_{\tilde{\boldsymbol{\Omega}}_{IMUi}} \tag{4.48}$$

where $\tilde{\boldsymbol{\Theta}}_{IMUi} \in \Re^{3\times1}$ is the mean Euler angle measurements of IMU *i*; $\tilde{\boldsymbol{\Omega}}_{IMUi} \in \Re^{3\times1}$ is the mean Euler angular rate measurements of IMU *i*; $\mathbf{b}_{\tilde{\boldsymbol{\Omega}}_{IMUi}} \in \Re^{3\times1}$ is the bias of IMU *i*

angular rate measurements; $\upsilon_{\tilde{\Theta}_{IMUi}}$ and $\upsilon_{\tilde{\Omega}_{IMUi}} \in \Re^{3\times 1}$ are the corresponding measurement noises respectively.

Then the complete form of measurement model is:

$$\begin{bmatrix} \tilde{\mathbf{p}}_{GPS1} \\ \tilde{\mathbf{v}}_{GPS1} \\ \vdots \\ \tilde{\mathbf{p}}_{GPSi} \\ \tilde{\mathbf{v}}_{GPSi} \\ \tilde{\mathbf{a}}_{IMU1} \\ \tilde{\mathbf{\Theta}}_{IMU1} \\ \tilde{\mathbf{\Omega}}_{IMU1} \\ \vdots \\ \tilde{\mathbf{a}}_{IMUi} \\ \tilde{\mathbf{\Theta}}_{IMUi} \\ \tilde{\mathbf{\Omega}}_{IMUi} \\ \tilde{\mathbf{p}}_{Other1} \\ \vdots \\ \tilde{\mathbf{p}}_{Ohteri} \end{bmatrix} = \begin{bmatrix} \mathbf{p}_{CG|n} + \mathbf{R}(\mathbf{\Theta}_{E|n})(\mathbf{p}_{GPS1|b} - \mathbf{p}_{CG|b}) \\ \mathbf{v}_{CG|n} + \dot{\mathbf{R}}(\mathbf{\Theta}_{E|n}, \mathbf{\Omega}_{E|n})(\mathbf{p}_{GPS1|b} - \mathbf{p}_{CG|b}) \\ \vdots \\ \mathbf{p}_{CG|n} + \mathbf{R}(\mathbf{\Theta}_{E|n})(\mathbf{p}_{GPSi|b} - \mathbf{p}_{CG|b}) \\ \mathbf{v}_{CG|n} + \dot{\mathbf{R}}(\mathbf{\Theta}_{E|n}, \mathbf{\Omega}_{E|n})(\mathbf{p}_{GPSi|b} - \mathbf{p}_{CG|b}) \\ \mathbf{a}_{CG|\mathrm{b}} + \mathbf{R}^T(\mathbf{\Theta}_{E|n})\left[\ddot{\mathbf{R}}(\mathbf{\Theta}_{E|n}, \mathbf{\Omega}_{E|n}, \mathbf{A}_{E|n})(\mathbf{p}_{\mathrm{IMU1}|b} - \mathbf{p}_{CG|b}) - \mathbf{g}\right] + \mathbf{b}_{\tilde{\mathbf{a}}_{IMU1}} \\ \mathbf{\Theta}_{E|n} \\ \mathbf{\Omega}_{E|n} + \mathbf{b}_{\tilde{\Omega}_{IMU1}} \\ \vdots \\ \mathbf{a}_{CG|\mathrm{b}} + \mathbf{R}^T(\mathbf{\Theta}_{E|n})\left[\ddot{\mathbf{R}}(\mathbf{\Theta}_{E|n}, \mathbf{\Omega}_{E|n}, \mathbf{A}_{E|n})(\mathbf{p}_{\mathrm{IMU}\,i|b} - \mathbf{p}_{CG|b}) - \mathbf{g}\right] + \mathbf{b}_{\tilde{\mathbf{a}}_{IMUi}} \\ \mathbf{\Theta}_{E|n} \\ \mathbf{\Omega}_{E|n} + \mathbf{b}_{\tilde{\Omega}_{IMUi}} \\ \mathbf{p}_{CG|n} + \mathbf{R}(\mathbf{\Theta}_{E|n})(\mathbf{p}_{Other1|b} - \mathbf{p}_{CG|b}) \\ \vdots \\ \mathbf{p}_{CG|n} + \mathbf{R}(\mathbf{\Theta}_{E|n})(\mathbf{p}_{Otheri|b} - \mathbf{p}_{CG|b}) \end{bmatrix} + \begin{bmatrix} \upsilon_{\tilde{\mathbf{p}}_{GPS1}} \\ \upsilon_{\tilde{\mathbf{v}}_{GPS1}} \\ \vdots \\ \upsilon_{\tilde{\mathbf{p}}_{GPSi}} \\ \upsilon_{\tilde{\mathbf{v}}_{GPSi}} \\ \upsilon_{\tilde{\mathbf{a}}_{IMU1}} \\ \upsilon_{\tilde{\mathbf{\Theta}}_{IMU1}} \\ \upsilon_{\tilde{\mathbf{\Omega}}_{IMU1}} \\ \vdots \\ \upsilon_{\tilde{\mathbf{a}}_{IMUi}} \\ \upsilon_{\tilde{\mathbf{\Theta}}_{IMUi}} \\ \upsilon_{\tilde{\mathbf{\Omega}}_{IMUi}} \\ \upsilon_{\tilde{\mathbf{p}}_{Other1}} \\ \vdots \\ \upsilon_{\tilde{\mathbf{p}}_{Ohteri}} \end{bmatrix} \quad (4.49)$$

Normally, the bias term $\mathbf{b}_{\tilde{\mathbf{a}}_{IMUi}}$ and $\mathbf{b}_{\tilde{\Omega}_{IMUi}}$ are unknown variables to be estimated in state vector. However, they will greatly increase the size of matrixes by 6 rows per IMU in process model if placed in state vector. Moreover, the size of matrixes in process model becomes variable while the number of available IMU changes, which makes it very difficult to implement the filter and it easily causes instability during UKF running.

In order to avoid the above situation, a feasible solution is to estimate the bias $\mathbf{b}_{\tilde{\mathbf{a}}_{IMUi}}$ and $\mathbf{b}_{\tilde{\Omega}_{IMUi}}$ outside the UKF instead, by transforming the estimated state variable $\mathbf{a}_{CG|\mathrm{b}}$ and

$\boldsymbol{\Omega}_{E|n}$ to each IMU location and calculating the mean of difference between measured value and transformed value as shown in Figure 5-2.

After the state is estimated at the end of each running loop of UKF, the bias $\mathbf{b}_{\mathbf{a},IMUi,out}$ and $\mathbf{b}_{\boldsymbol{\Omega},IMUi,out}$ used for output are severally calculated by

$$\mathbf{b}_{\mathbf{a},IMUi,out} = \tilde{\mathbf{a}}_{IMUi} - \mathbf{a}_{CG|\mathrm{b}} + \mathbf{R}^T(\boldsymbol{\Theta}_{E|n})\left[\mathbf{g} - \ddot{\mathbf{R}}(\boldsymbol{\Theta}_{E|n}, \boldsymbol{\Omega}_{E|n}, \mathbf{A}_{E|n})(\mathbf{p}_{\mathrm{IMU}i|b} - \mathbf{p}_{CG|b})\right] \quad (4.50)$$

$$\mathbf{b}_{\boldsymbol{\Omega},IMUi,out} = \tilde{\boldsymbol{\Omega}}_{IMUi} - \boldsymbol{\Omega}_{E|n} \quad (4.51)$$

And their means of all previous loops become the estimated bias $\mathbf{b}_{\tilde{\mathbf{a}}_{IMUi}}$ and $\mathbf{b}_{\tilde{\Omega}_{IMUi}}$ respectively which are the feedbacks for next UKF loop

$$\mathbf{b}_{\tilde{\mathbf{a}}_{IMUi}} = \frac{1}{n}\sum_{1}^{n^{th}\,loop} \mathbf{b}_{\mathbf{a},IMUi,out} \quad (4.52)$$

$$\mathbf{b}_{\tilde{\Omega}_{IMUi}} = \frac{1}{n}\sum_{1}^{n^{th}\,loop} \mathbf{b}_{\boldsymbol{\Omega},IMUi,out} \quad (4.53)$$

5.2.2 Approach 2 without Acceleration Measurement

Acceleration is not a must to determine the position, velocity and heading. It may be used to increase the accuracy of the estimate, whereas, is it necessary to use acceleration measurement by adding more computation load on the system?

The UKF model (4.29) (4.49) and can be simplified by eliminating the acceleration term and its relevant term. The new process model and measurement model is respectively expressed as

$$\begin{bmatrix} \dot{\mathbf{p}}_{CG|n} \\ \dot{\mathbf{v}}_{CG|n} \\ \dot{\boldsymbol{\Theta}}_{E|n} \\ \dot{\boldsymbol{\Omega}}_{E|n} \\ \dot{\mathbf{b}}_{\tilde{\Omega}_{IMUi}} \end{bmatrix} = \begin{bmatrix} \mathbf{O}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} \end{bmatrix} \begin{bmatrix} \mathbf{p}_{CG|n} \\ \mathbf{v}_{CG|n} \\ \boldsymbol{\Theta}_{E|n} \\ \boldsymbol{\Omega}_{E|n} \\ \mathbf{b}_{\tilde{\Omega}_{IMUi}} \end{bmatrix} + \begin{bmatrix} \mathbf{O}_{3\times3} \\ \boldsymbol{w}_{\mathbf{v}_{CG|n}} \\ \mathbf{O}_{3\times3} \\ \boldsymbol{w}_{\boldsymbol{\Omega}_{E|n}} \\ \boldsymbol{w}_{\mathbf{b}_{\tilde{\Omega}_{IMUi}}} \end{bmatrix} \tag{4.54}$$

$$\begin{bmatrix} \tilde{\mathbf{p}}_{GPS1} \\ \tilde{\mathbf{v}}_{GPS1} \\ \vdots \\ \tilde{\mathbf{p}}_{GPSi} \\ \tilde{\mathbf{v}}_{GPSi} \\ \tilde{\boldsymbol{\Theta}}_{IMU1} \\ \tilde{\boldsymbol{\Omega}}_{IMU1} \\ \vdots \\ \tilde{\boldsymbol{\Theta}}_{IMUi} \\ \tilde{\boldsymbol{\Omega}}_{IMUi} \\ \tilde{\mathbf{p}}_{Other1} \\ \vdots \\ \tilde{\mathbf{p}}_{Ohteri} \end{bmatrix} = \begin{bmatrix} \mathbf{p}_{CG|n} + \mathbf{R}(\boldsymbol{\Theta}_{E|n})(\mathbf{p}_{GPS1|b} - \mathbf{p}_{CG|b}) \\ \mathbf{v}_{CG|n} + \dot{\mathbf{R}}(\boldsymbol{\Theta}_{E|n}, \boldsymbol{\Omega}_{E|n})(\mathbf{p}_{GPS1|b} - \mathbf{p}_{CG|b}) \\ \vdots \\ \mathbf{p}_{CG|n} + \mathbf{R}(\boldsymbol{\Theta}_{E|n})(\mathbf{p}_{GPSi|b} - \mathbf{p}_{CG|b}) \\ \mathbf{v}_{CG|n} + \dot{\mathbf{R}}(\boldsymbol{\Theta}_{E|n}, \boldsymbol{\Omega}_{E|n})(\mathbf{p}_{GPSi|b} - \mathbf{p}_{CG|b}) \\ \boldsymbol{\Theta}_{E|n} \\ \boldsymbol{\Omega}_{E|n} + \mathbf{b}_{\tilde{\Omega}_{IMU1}} \\ \vdots \\ \boldsymbol{\Theta}_{E|n} \\ \boldsymbol{\Omega}_{E|n} + \mathbf{b}_{\tilde{\Omega}_{IMUi}} \\ \mathbf{p}_{CG|n} + \mathbf{R}(\boldsymbol{\Theta}_{E|n})(\mathbf{p}_{Other1|b} - \mathbf{p}_{CG|b}) \\ \vdots \\ \mathbf{p}_{CG|n} + \mathbf{R}(\boldsymbol{\Theta}_{E|n})(\mathbf{p}_{Otheri|b} - \mathbf{p}_{CG|b}) \end{bmatrix} + \begin{bmatrix} \boldsymbol{\upsilon}_{\tilde{\mathbf{p}}_{GPS1}} \\ \boldsymbol{\upsilon}_{\tilde{\mathbf{v}}_{GPS1}} \\ \vdots \\ \boldsymbol{\upsilon}_{\tilde{\mathbf{p}}_{GPSi}} \\ \boldsymbol{\upsilon}_{\tilde{\mathbf{v}}_{GPSi}} \\ \boldsymbol{\upsilon}_{\tilde{\boldsymbol{\Theta}}_{IMU1}} \\ \boldsymbol{\upsilon}_{\tilde{\boldsymbol{\Omega}}_{IMU1}} \\ \vdots \\ \boldsymbol{\upsilon}_{\tilde{\boldsymbol{\Theta}}_{IMUi}} \\ \boldsymbol{\upsilon}_{\tilde{\boldsymbol{\Omega}}_{IMUi}} \\ \boldsymbol{\upsilon}_{\tilde{\mathbf{p}}_{Other1}} \\ \vdots \\ \boldsymbol{\upsilon}_{\tilde{\mathbf{p}}_{Ohteri}} \end{bmatrix} \tag{4.55}$$

5.2.3 Least Measurement Mode

If failure of one or several sensors is detected, the failed sensors will be kicked out from (4.49) and the UKF is still running properly in most cases. However, if there is only one position sensor or one angular sensor left, there is no enough information to determine the state of the UKF and it will output wrong estimate diverged quickly. The measurement model of minimum sensors for the UKF is

$$\begin{bmatrix} \tilde{\mathbf{p}}_{sensorm} \\ \tilde{\mathbf{p}}_{sensorn} \end{bmatrix} = \begin{bmatrix} \mathbf{p}_{CG|n} + \mathbf{R}(\boldsymbol{\Theta}_{E|n})(\mathbf{p}_{sensorm|b} - \mathbf{p}_{CG|b}) \\ \mathbf{p}_{CG|n} + \mathbf{R}(\boldsymbol{\Theta}_{E|n})(\mathbf{p}_{sensornb} - \mathbf{p}_{CG|b}) \end{bmatrix} + \begin{bmatrix} \boldsymbol{\upsilon}_{\tilde{\mathbf{p}}_{sensorm}} \\ \boldsymbol{\upsilon}_{\tilde{\mathbf{p}}_{sensorn}} \end{bmatrix} \tag{4.56}$$

or

$$\begin{bmatrix} \tilde{\mathbf{p}}_{sensorm} \\ \tilde{\boldsymbol{\Theta}}_{sensorn} \end{bmatrix} = \begin{bmatrix} \mathbf{p}_{CG|n} + \mathbf{R}(\boldsymbol{\Theta}_{E|n})(\mathbf{p}_{sensorm|b} - \mathbf{p}_{CG|b}) \\ \boldsymbol{\Theta}_{E|n} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\upsilon}_{\tilde{\mathbf{p}}_{sensorm}} \\ \boldsymbol{\upsilon}_{\tilde{\Theta}_{sensorn}} \end{bmatrix} \tag{4.57}$$

There are only two position measurements available in (4.56) , however, at least three position measurements at different locations are needed to determine the motion of a rigid body in three dimensional world. Therefore (4.56) seems incomplete to estimate the state, whereas, some extra information is known depending on the understanding of the physical model, which makes it possible to estimate the heading and position of a floating vessel.

In operational mode, the vessel is floating on sea surface and it is in the state of stable equilibrium in *z*, roll and pitch direction in NED frame. Then it is a good assumption that the measurement of roll and pitch is zero, which gives an extra virtual measurement for the UKF

$$\tilde{\boldsymbol{\Theta}}_{sensor\_virtual} = \boldsymbol{\Theta}_{E|n} + \boldsymbol{\upsilon}_{\tilde{\Theta}_{sensor\_virtual}} \tag{4.58}$$

where $\tilde{\boldsymbol{\Theta}}_{sensor\_virtual} = [0, 0, \psi^-]^T$ and ψ^- is the value of ψ in the last loop and the measurement error covariance of $\tilde{\boldsymbol{\Theta}}_{sensor\_virtual}$ is set to a larger value than normal because this virtual measurement is inaccurate. Then the augmented measurement model of (4.56) is written as

$$\begin{bmatrix} \tilde{\mathbf{p}}_{sensorm} \\ \tilde{\mathbf{p}}_{sensorn} \\ \tilde{\boldsymbol{\Theta}}_{sensor\_virtual} \end{bmatrix} = \begin{bmatrix} \mathbf{p}_{CG|n} + \mathbf{R}(\boldsymbol{\Theta}_{E|n})(\mathbf{p}_{sensorm|b} - \mathbf{p}_{CG|b}) \\ \mathbf{p}_{CG|n} + \mathbf{R}(\boldsymbol{\Theta}_{E|n})(\mathbf{p}_{sensornb} - \mathbf{p}_{CG|b}) \\ \boldsymbol{\Theta}_{E|n} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\upsilon}_{\mathbf{p},sensorm} \\ \boldsymbol{\upsilon}_{\mathbf{p},sensorn} \\ \boldsymbol{\upsilon}_{\tilde{\Theta}_{sensor\_virtual}} \end{bmatrix} \tag{4.59}$$

For the second situation (4.57), the least measurements required are one position measurement and one yaw measurement. $\tilde{\boldsymbol{\Theta}}_{sensorn} = [0, 0, \psi]^T$ and ψ is the yaw

measurement and the measurement error covariance of $\tilde{\boldsymbol{\Theta}}_{IMU}$ corresponding to roll and pitch is set to a larger value than normal just like what is done for (4.58). In this way, the redundancy of the DP system is increased since the failure of heading measurements will not result in position loss if there are at least two available position sensors.

If the least sensor mode cannot be satisfied, the UKF is not able to estimate the state vector. Then the Kalman filter in Figure 5-1 will start dead reckoning while the state of the UKF will keep unchanged until enough measurements are recovered to determine the state of the filter.

As mentioned above, at least three position measurements at different locations are needed to determine the motion of a rigid body in three-dimensional world. Theoretically, the filter is able to work in the condition where there are three position measurements of different point on the vessel without angular measurement. However, there is abrupt change of the output of the UKF at the moment when angular measurements are switched between available and unavailable. The solution is to put (4.58) in the measurement model when angular measurements are all lost so that there is always angular measurement in the measurement model.

5.3 Tuning

The general tuning of the Kalman filter is to set up the values of noise covariance matrixes $\mathbf{Q}$ and $\mathbf{R}$ so that the filter is robust enough to work properly regardless of initial errors. [47] $\mathbf{R}$ can be simply found according to the covariance of sensor noise. However, the value of $\mathbf{Q}$ is more difficult to tune despite the model knowledge is more or less known. It is tuned by trial or using adaptive methods. Three additional parameters should be tuned for the UKF. α determines the spread of the sigma points around $\hat{\mathbf{x}}$ and its value is chosen between 0 and 1. For Gaussian noise, it is the optimal

choice that $\beta = 2$. Choose $\kappa \geq 0$ to guarantee positive definiteness of the covariance matrix and $\kappa = 0$ is a good default choice.

5.4 Conclusion

The process models of UKF in approach 1 and approach 2 are identical; however, the measurement models are different. Obviously, the measurement model in approach 2 is more complex than that in approach 1 and its complexity is proportional to the number of measurements, which means the computational load of approach 2 is much higher if there are lots of sensors. In practice, there are indeed lots of sensors working during DP operation for the sake of redundancy. Nevertheless, if taking accuracy into account, approach 2 is expected to be more accurate because all measurements are combined by kinematics in measurement model while only the mean of measurements are combined in approach 1. Furthermore, the least sensor mode in approach 2 is easier to be satisfied and this will increase the redundancy of the measurement system. As a conclusion, approach 2 is a better choice under comprehensive consideration since the computation load can be handled well by current computers. Considering the numerical performance, SRUKF is used instead of UKF in this work.

6. Observer Design for DP

The motions of a floating vessel consist of low frequency motions and wave frequency motions. In low to medium sea states, the wave frequency motion has little influence on the performance of DP operation. In order to reduce power consumption and wear of the actuators, only low frequency motions are controlled by DP system. [5] The total motions of low frequency motions and wave frequency motions have already been estimated in the previous chapter. Hence, the work of this chapter is focused on estimating low frequency motions based on Kalman filter with the knowledge of the vessel mathematical model as well as the output of the SRUKF in the previous chapter. For safety reasons this observer must be able to keep working after the position measurements are lost. Thus, dead reckoning is discussed in this chapter as well, which will keep the position after the entire reference system fails.

6.1 Vessel Mathematical Model

6.1.1 Motion of a Floating Vessel

There are six degrees of freedom (DOF) defined as surge, sway, heave, roll, pitch and yaw [19] as shown in Figure 2-3 and Table 6-1 where the forces and velocities are also defined. Floating vessels are exposed to external forces caused by current, wind and waves and these forces have high frequency components as well as low frequency components. However, only low frequency forces are of interest to motion control because the frequency of oscillations of the first order wave forces has little influence on the operational performance of a vessel. Therefore, the dynamic positioning system will not counteract the first order wave forces. Besides, if the controller responds to the first order wave frequency, the actuators would react in a high frequency which results

in unwanted power consumption and actuator wear. [5] For operation of a vessel, three degrees of freedom (surge, sway and yaw) are taken into account.

Table 6-1 Notation for marine vessels

DOF	Description	Forces and moments	Velocities and angluar rates	Positions and Euler angles
1	Translation in x direction (surge)	X	u	x
2	Translation in y direction (sway)	Y	v	y
3	Translation in z direction (heave)	Z	w	z
4	Rotation about x axis (roll)	K	p	ϕ
5	Rotation about y axis (pitch)	M	q	θ
6	Rotation about z axis (yaw)	N	r	ψ

6.1.2 Low Frequency Motion

The model of vessel kinetics is built by [48] and the equation of low frequency motion of a vessel in matrix form can be written as [5]

$$\dot{\boldsymbol{\eta}} = \mathbf{R}_z \mathbf{v}_b \tag{5.1}$$

$$\dot{\mathbf{v}}_b = -\mathbf{M}^{-1}\mathbf{D}\mathbf{v}_b + \mathbf{M}^{-1}\boldsymbol{\tau} + \mathbf{M}^{-1}\mathbf{R}_z{}^T\mathbf{b} \tag{5.2}$$

$$\dot{\mathbf{b}} = -\mathbf{T}^{-1}\mathbf{b} + \boldsymbol{\Psi}\boldsymbol{w}_b \tag{5.3}$$

In (5.1) the matrix $\boldsymbol{\eta} = \begin{bmatrix} x_n & y_n & \psi \end{bmatrix}^T$ represents the position x, y and heading ψ in NED frame and $\mathbf{v}_b = \begin{bmatrix} u & v & r \end{bmatrix}^T$ describe the velocities in the body-fixed frame and $\mathbf{R}_z$ is the rotation transformation matrix about z-axis of NED. $\mathbf{b}$ in (5.2) is a vector of

bias forces, which contains all unknown low frequency forces such as current and wave drift forces as well as pipe tension, etc. $\mathbf{T}$ in (5.3) is a diagonal matrix of positive bias time constants. $\boldsymbol{\Psi}$ is a diagonal matrix scaling the noise term $\boldsymbol{w}_b$. It is considered more appropriate to use $\dot{\mathbf{b}} = \boldsymbol{\Psi}\boldsymbol{w}_b$ under the assumption that the bias forces are almost constant and driven by noise because wave drift forces and current forces are changing slowly. The force matrix $\boldsymbol{\tau}$ is the sum of known and measurable forces including control forces and wind forces acting on the vessel in body-fixed frame. The inertial matrix $\mathbf{M}$ in (5.2) is the sum of rigid-body mass matrix $\mathbf{M}_{RB}$ and hydrodynamic added mass matrix $\mathbf{M}_A$ and they are written as

$$\mathbf{M}_{RB} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & mx_g \\ 0 & mx_g & I_z \end{bmatrix} \tag{5.4}$$

$$\mathbf{M}_A = \begin{bmatrix} -X_{\dot{u}} & 0 & 0 \\ 0 & -Y_{\dot{v}} & -Y_{\dot{r}} \\ 0 & -Y_{\dot{r}} & -N_{\dot{r}} \end{bmatrix} \tag{5.5}$$

$$\mathbf{M} = \begin{bmatrix} m - X_{\dot{u}} & 0 & 0 \\ 0 & m - Y_{\dot{v}} & mx_g - Y_{\dot{r}} \\ 0 & mx_g - Y_{\dot{r}} & I_z - N_{\dot{r}} \end{bmatrix} \tag{5.6}$$

where m is the mass of the vessel, x_g is the longitudinal position of center of gravity of the vessel in body-fixed frame, I_z denotes the moment of inertia about z-axis of body-fixed frame. The hydrodynamic added mass terms are defined as

$$X_{\dot{u}} = \frac{\partial X}{\partial \dot{u}}, \quad Y_{\dot{v}} = \frac{\partial Y}{\partial \dot{v}}, \quad Y_{\dot{r}} = \frac{\partial Y}{\partial \dot{r}}, \quad N_{\dot{r}} = \frac{\partial N}{\partial \dot{r}} \tag{5.7}$$

In low speed applications such as dynamic positioning, the linear damping matrix **D** in (5.2) is given by

$$\mathbf{D} = \begin{bmatrix} -X_u & 0 & 0 \\ 0 & -Y_v & -Y_r \\ 0 & -Y_r & -N_r \end{bmatrix} \tag{5.8}$$

where linear damping coefficients are defined as

$$X_u = \frac{\partial X}{\partial u}, \quad Y_v = \frac{\partial Y}{\partial v}, \quad Y_r = \frac{\partial Y}{\partial r}, \quad N_r = \frac{\partial N}{\partial r} \tag{5.9}$$

6.1.3 First Order Wave Frequency Motion

The first order wave-induced motion in surge, sway and yaw direction can be modeled as second order harmonic oscillations driven by Gaussian white noises. [49] The wave frequency motion of frequency domain is written as [50]

$$\xi_{(i)}(s) = h_{\omega i}(s) w_i(s) \tag{5.10}$$

and

$$h_{\omega i}(s) = \frac{\sigma_i s}{s^2 + 2\zeta_i \omega_{0i} + \omega_{0i}^2} \tag{5.11}$$

where i=1,2,3 representing surge, sway and yaw respectively. w_i is a zero-mean white noise, ω_{0i} is the dominating wave frequency which can be tracked using the method from [51] or [20], ζ_i is the relative damping ratio and σ_i is a parameter related to wave intensity.

The inverse Laplace transform of (5.10) to time domain is given by

$$\frac{d^2 \xi_{(i)}}{dt^2} + 2\zeta_i \omega_{0i} \frac{d\xi_{(i)}}{dt} + \omega_{0i}^2 \xi_{(i)} = \sigma_i w_i \tag{5.12}$$

Let

$$\xi_1 = \begin{bmatrix} \xi_{(1)} & \xi_{(2)} & \xi_{(3)} \end{bmatrix}^T \tag{5.13}$$

$$\xi_2 = \begin{bmatrix} \frac{d\xi_{(1)}}{dt} & \frac{d\xi_{(2)}}{dt} & \frac{d\xi_{(3)}}{dt} \end{bmatrix}^T \tag{5.14}$$

$$\mathbf{\Omega} = diag(\begin{bmatrix} \omega_{01} & \omega_{02} & \omega_{03} \end{bmatrix}) \tag{5.15}$$

$$\mathbf{Z}_w = diag(\begin{bmatrix} \zeta_1 & \zeta_2 & \zeta_3 \end{bmatrix}) \tag{5.16}$$

$$\mathbf{\Sigma} = diag(\begin{bmatrix} \sigma_1 & \sigma_2 & \sigma_3 \end{bmatrix}) \tag{5.17}$$

$$\boldsymbol{w}_\xi = \begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix}^T \tag{5.18}$$

Then (5.12)-(5.18) compact

$$\begin{bmatrix} \dot{\boldsymbol{\xi}}_1 \\ \dot{\boldsymbol{\xi}}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{O}_{3\times3} & \mathbf{I}_{3\times3} \\ \mathbf{-\Omega^2} & \mathbf{-2Z_w\Omega} \end{bmatrix} \begin{bmatrix} \boldsymbol{\xi}_1 \\ \boldsymbol{\xi}_2 \end{bmatrix} + \begin{bmatrix} \mathbf{O}_{3\times3} \\ \mathbf{\Sigma} \end{bmatrix} \boldsymbol{w}_\xi \tag{5.19}$$

6.1.4 Summary of the Observer

By combining (5.1), (5.2), (5.3) and (5.19) in one matrix form, the low frequency model and wave frequency model can be written as

$$\begin{bmatrix} \dot{\boldsymbol{\eta}} \\ \dot{\mathbf{v}}_\mathbf{b} \\ \dot{\mathbf{b}} \\ \dot{\boldsymbol{\xi}}_1 \\ \dot{\boldsymbol{\xi}}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{O}_{3\times3} & \mathbf{R}_z & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} & \mathbf{-M^{-1}D} & \mathbf{M^{-1}R}_z^T & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{I}_{3\times3} \\ \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{-\Omega^2} & \mathbf{-2Z_w\Omega} \end{bmatrix} \begin{bmatrix} \boldsymbol{\eta} \\ \mathbf{v}_\mathbf{b} \\ \mathbf{b} \\ \boldsymbol{\xi}_1 \\ \boldsymbol{\xi}_2 \end{bmatrix} + \begin{bmatrix} \mathbf{O}_{3\times3} \\ \mathbf{M^{-1}} \\ \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} \end{bmatrix} \boldsymbol{\tau} + \begin{bmatrix} \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} \\ \mathbf{\Psi} & \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} & \mathbf{\Sigma} \end{bmatrix} \begin{bmatrix} \boldsymbol{w}_\mathbf{b} \\ \boldsymbol{w}_\xi \end{bmatrix} \tag{5.20}$$

and its measurement model is expressed by

$$\begin{bmatrix} \tilde{\boldsymbol{\eta}} \\ \tilde{\mathbf{v}}_n \\ \tilde{\mathbf{a}}_n \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} & \mathbf{R}_z & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{I}_{3\times3} \\ \mathbf{O}_{3\times3} & -\mathbf{R}_z\mathbf{M}^{-1}\mathbf{D} & \mathbf{R}_z\mathbf{M}^{-1}\mathbf{R}_z^T & -\boldsymbol{\Omega}^2 & -2\mathbf{Z}_w\boldsymbol{\Omega} \end{bmatrix} \begin{bmatrix} \boldsymbol{\eta} \\ \mathbf{v}_b \\ \mathbf{b} \\ \boldsymbol{\xi}_1 \\ \boldsymbol{\xi}_2 \end{bmatrix} + \begin{bmatrix} \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} \\ \mathbf{R}_z\mathbf{M}^{-1} \end{bmatrix} \boldsymbol{\tau} + \begin{bmatrix} \upsilon_{\tilde{\boldsymbol{\eta}}} \\ \upsilon_{\tilde{\mathbf{v}}_n} \\ \upsilon_{\tilde{\mathbf{a}}_n} \end{bmatrix} \quad (5.21)$$

where the measurement vector $\begin{bmatrix} \tilde{\boldsymbol{\eta}} & \tilde{\mathbf{v}}_n & \tilde{\mathbf{a}}_n \end{bmatrix}^T$ comes from the estimated state of the UKF or SRUKF.

6.2 Observability and Stabilizability

The Kalman filter for state estimate for the DP system is designed in the previous sector and the important properties of the model including observability and stabilizability should also be studied. All these properties has been discussed in [7], conclusions of which are used as reference here.

The following Hautus test [52] is used to test the observability of linear time-invariant systems.

Theorem 7.1 The *n* state linear time-invariant system in the form of (3.10) and (3.11) is observable if and only if

$$rank\begin{bmatrix} \mathbf{F} - \lambda\mathbf{I} \\ \mathbf{H} \end{bmatrix} = n, \quad \forall \lambda \in \mathbb{C} \quad (5.22)$$

where λ is an eigenvalue of matrix $\mathbf{F}$.

Theorem 7.2 The *n* state linear time-invariant system in the form of (3.10) and (3.11) is stabilizable if and only if [52]

$$rank\begin{bmatrix} \mathbf{F} - \lambda\mathbf{I} & \mathbf{B} \end{bmatrix} = n, \quad \forall \lambda \in \mathbb{C} : \Re(\lambda) \geq 0. \quad (5.23)$$

In this case, (5.20) and (5.21) is the linear time-invariant system to be tested, which means

$$\mathbf{F}=\begin{bmatrix} \mathbf{O}_{3\times3} & \mathbf{R}_z & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} & \mathbf{-M^{-1}D} & \mathbf{M^{-1}R}_z^T & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{I}_{3\times3} \\ \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{-\Omega^2} & \mathbf{-2Z_w\Omega} \end{bmatrix} \tag{5.24}$$

$$\mathbf{H}=\begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} & \mathbf{R}_z & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{I}_{3\times3} \\ \mathbf{O}_{3\times3} & \mathbf{-R_zM^{-1}D} & \mathbf{R_zM^{-1}R}_z^T & \mathbf{-\Omega^2} & \mathbf{-2Z_w\Omega} \end{bmatrix} \tag{5.25}$$

$$\mathbf{B}=\begin{bmatrix} \mathbf{O}_{3\times3} \\ \mathbf{M^{-1}} \\ \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} \end{bmatrix} \tag{5.26}$$

and λ is calculated from matrix $\mathbf{F}$.

Thus, condition (5.22) and (5.23) can be easily tested in Matlab and this system is proved to be observable and stabilizable as a result.

6.3 Discretization of KF

Kalman filter is used as the observer for the DP system in this thesis. The continuous form of Kalman filter is not convenient for the computation in practice and it need to be discretized. The differential form (5.20) can be discretized by the method presented in section 4.2.2, which yields

$$\mathbf{x}_k = \mathbf{A}_k\mathbf{x}_{k-1} + \mathbf{B}_k\boldsymbol{\tau}_{k-1} + \boldsymbol{w}_{k-1} \tag{5.27}$$

where $\mathbf{x}_k = \begin{bmatrix} \boldsymbol{\eta} & \mathbf{v_b} & \mathbf{b} & \boldsymbol{\xi}_1 & \boldsymbol{\xi}_2 \end{bmatrix}_k^T$, $\mathbf{A}_k = \mathrm{e}^{\mathbf{F}T_s} = \mathbf{I} + \mathbf{F}T_s + \frac{(\mathbf{F}T_s)^2}{2!} + \cdots + \frac{(\mathbf{F}T_s)^n}{n!} + \cdots$ and it is simplified as $\mathbf{A}_k = \mathbf{I} + \mathbf{F}T_s + \frac{(\mathbf{F}T_s)^2}{2}$ by ignoring the terms higher than 2nd order, however, it causes inaccuracy in this discrete model. If the system has dynamics that are dominantly faster than T_s, divergence problems, i.e., instability, can occur. To avoid this problem, two typical methods to increase numerical accuracy and stability [47], are to shorten the sample time or use more complex discretization schemes such as Runge-Kutta methods [53].

$\mathbf{B}_k$ and $\mathbf{Q}_k$ are calculated based on the discretized process transition matrix $\mathbf{A}_k$. Thus, $\mathbf{B}_k$ is obtained from

$$\mathbf{B}_k = \int_0^{T_s} \mathbf{A}_k(\tau) \begin{bmatrix} \mathbf{O}_{3\times3} \\ \mathbf{M}^{-1} \\ \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} \end{bmatrix} d\tau \tag{5.28}$$

and $\mathbf{Q}_k$ is given by

$$\mathbf{Q}_k = \int_0^{T_s} \mathbf{A}_k(\tau)\mathbf{Q}\mathbf{A}_k{}^T(\tau)d\tau \tag{5.29}$$

where

$$\mathbf{Q} = E\left\{ \begin{bmatrix} \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} \\ \boldsymbol{\Psi} & \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} & \boldsymbol{\Sigma} \end{bmatrix} \begin{bmatrix} \boldsymbol{w}_\mathbf{b} \\ \boldsymbol{w}_\xi \end{bmatrix} \begin{bmatrix} \boldsymbol{w}_\mathbf{b} \\ \boldsymbol{w}_\xi \end{bmatrix}^T \begin{bmatrix} \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} \\ \boldsymbol{\Psi} & \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} & \boldsymbol{\Sigma} \end{bmatrix}^T \right\} \tag{5.30}$$

Then this discrete Kalman filter can be implemented by using (3.5)-(3.9) but the prediction equation $\hat{\mathbf{x}}_k^- = \mathbf{A}_k\hat{\mathbf{x}}_{k-1} + \mathbf{B}_k\mathbf{u}_{k-1}$ is replaced by fourth order Runge-Kutta method, and the procedure is shown below:

1) Initialization:

$$\hat{\mathbf{x}}_0 = E(\mathbf{x}_0), \mathbf{P}_0 = E\left[(\mathbf{x}_0 - \hat{\mathbf{x}}_0)(\mathbf{x}_0 - \hat{\mathbf{x}}_0)^T\right] \tag{5.31}$$

2) Time update equations:

$$k_1 = \mathbf{F}\hat{\mathbf{x}}_{k-1} + \mathbf{G}\mathbf{u} \tag{5.32}$$

$$k_2 = \mathbf{F}\left(\hat{\mathbf{x}}_{k-1} + \frac{T_s}{2}k_1\right) + \mathbf{G}\mathbf{u} \tag{5.33}$$

$$k_3 = \mathbf{F}\left(\hat{\mathbf{x}}_{k-1} + \frac{T_s}{2}k_2\right) + \mathbf{G}\mathbf{u} \tag{5.34}$$

$$k_4 = \mathbf{F}\left(\hat{\mathbf{x}}_{k-1} + T_s k_3\right) + \mathbf{G}\mathbf{u} \tag{5.35}$$

$$\hat{\mathbf{x}}_k^- = \hat{\mathbf{x}}_{k-1} + \frac{T_s}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) \tag{5.36}$$

$$\mathbf{P}_k^- = \mathbf{A}_k\mathbf{P}_{k-1}\mathbf{A}_k^T + \mathbf{Q}_k \tag{5.37}$$

3) Measurement update equations:

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^T(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T + \mathbf{R}_k)^{-1} \tag{5.38}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_k^-) \tag{5.39}$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^- \tag{5.40}$$

6.4 Tuning

The signal-to-noise ratio (SNR) $\|\mathbf{Q}\|/\|\mathbf{R}\|$, which sets the filter speed, is the most crucial in tuning. $\mathbf{R}$ can be set according to sensor specification. Then $\mathbf{Q}$ is tuned after $\mathbf{R}$ is fixed. The filter is able to adapt to changes fast with high SNR in the model, but with larger uncertainty, vice versa. $\mathbf{P}_0$ indicates the belief in the prior $\mathbf{x}_0$. Therefore, choose $\mathbf{P}_0$ very large if no prior information of $\mathbf{x}_0$ exists.

6.5 Dead Reckoning under Reference System Failure

This observer must be able to keep working after position is lost for sake of safety. When position is lost, the UFK will keep the current state without update and inform the observer that this is no position information available. Then the observer will do dead reckoning by estimating the state based on process model.

An activation matrix $\mathbf{f} \in \Re^{n\times n}$ (n is the size of measurement vector z) is defined to indicate which measurement variable in measurement vector is missing [54]:

$$\mathbf{f}_{i,i} = \begin{cases} 0 & \textit{measurement missing} \\ 1 & \textit{measurement available} \end{cases} \tag{5.41}$$

where $\mathbf{f}$ is diagonal matrix and $\mathbf{f}_{i,i}$ is the element in i^{th} row *and* i^{th} column.

The measurement update equation (5.39) is modified by

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k\mathbf{f}(\mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_k^-) \tag{5.42}$$

As a result, the position and heading of the vessel are estimated in the worst case in which the whole reference system fails. The vessel is still capable of maintaining its position for a period the length of which depends on the external environment. If the

external forces are changing slowly, then the dead reckoning accuracy decreases slowly and vice versa.

6.6 Conclusion

In this chapter, the observer for low frequency motions estimation is designed and the estimated motions can be output to DP controller when the result of the UKF (SRUKF) is connected to this observer. By now the whole algorithm for DP observer is done and the next step is to simulate the system in Simulink and check the performance of this system.

7. Simulation and Results

The simulation setup and results of Matlab/Simulink model are discussed in this chapter. The simulation program (Figure 7-1) in this thesis is developed by combining the author's modules and some modules from the existing Simulink toolbox Marine Systems Simulator (MSS) which provides the necessary resources for implementation of mathematical models of marine systems. [55] Environmental module and marine vessel module are directly from the library of MSS and the controller module from MSS is modified in the simulation program to achieve a better performance. The reference system is developed by the author to simulate measurement sensors at different locations of the vessel. Then sensor fusion module and observer module are built based on the algorithms described in this thesis. The results of both SRUKF and KF estimates will be analyzed at the end of this chapter.

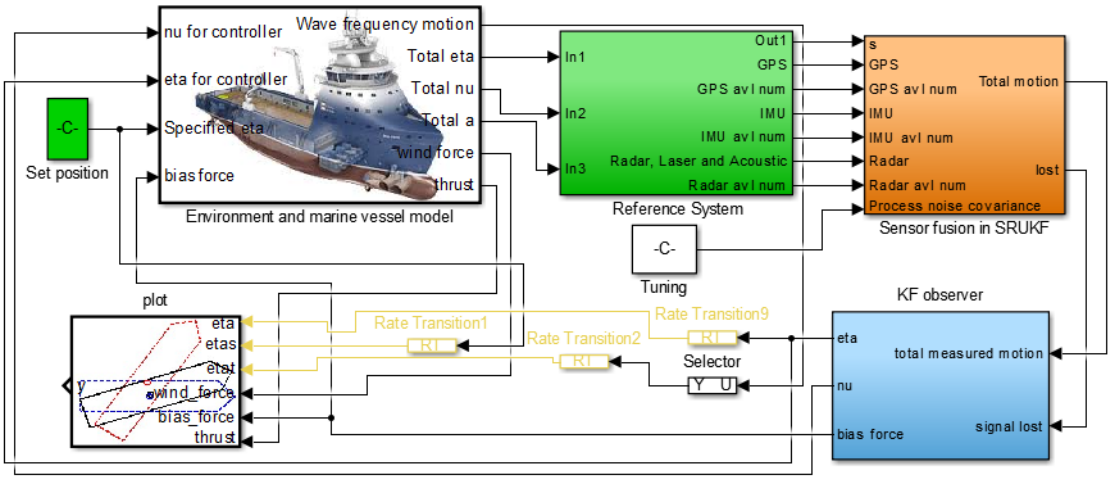


Figure 7-1: Overview of the simulation program

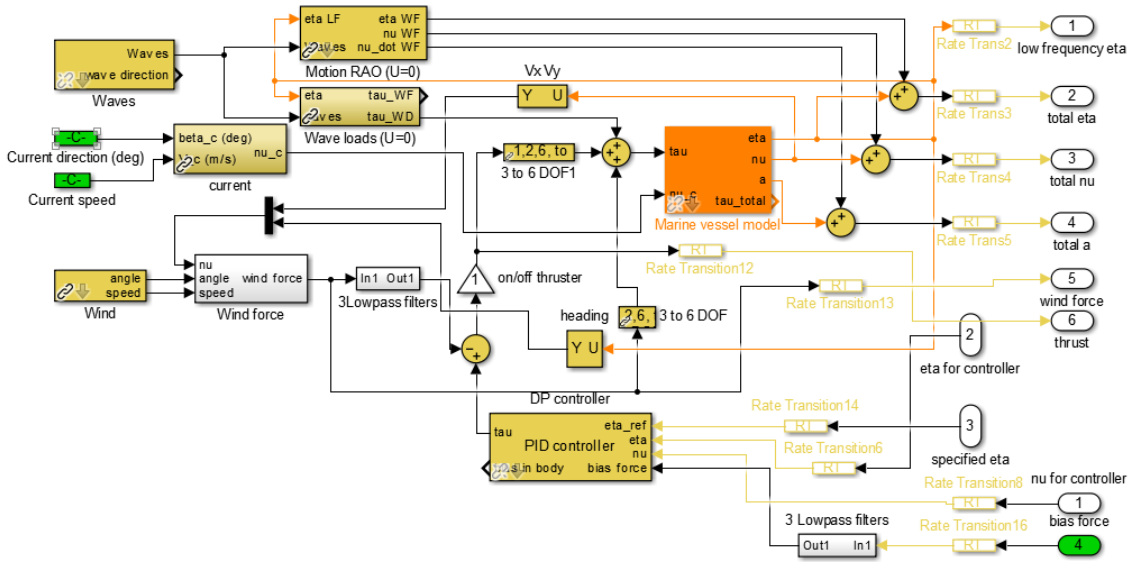


Figure 7-2: Blocks inside 'Environment and marine vessel model' in Figure 7-1

7.1 Simulation Program Description

7.1.1 Environmental Module

An irregular sea state is simulated by environmental module in Figure 7-2 where waves, current and wind are generated. The wave parameters such as significant wave height, peak wave frequency, mean wave directions, etc. are set and then N harmonic wave components are generated and superposed to simulate irregular waves. The wave spectrum and wave realization are plotted in Figure 7-3 and Figure 7-4 respectively. The current is generated by specifying the current direction and speed. The wind is generated based on the mean angle and mean wind speed at 10 meters height and both the total velocity and direction of the wind are slowly varying. [56]

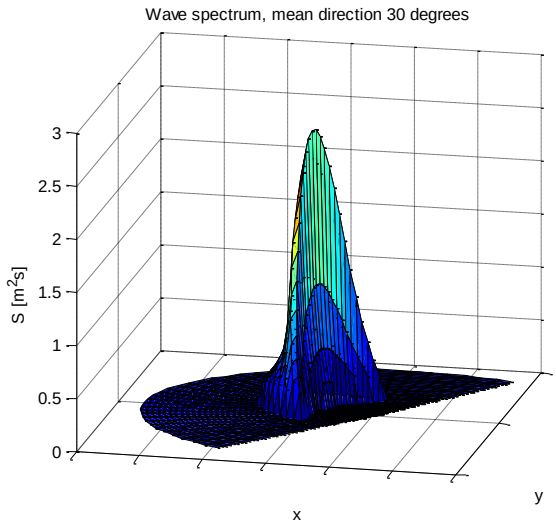


Figure 7-3: Wave spectrum for simulation

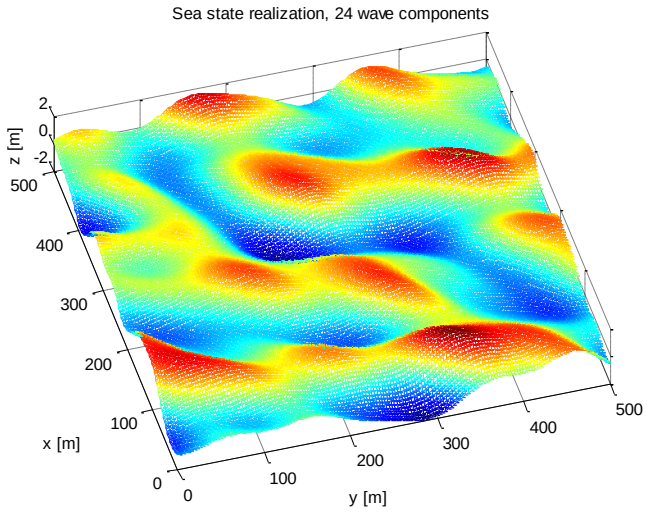


Figure 7-4: Sea state in Simulink

7.1.2 Marine Vessel Module

The marine vessel module in Figure 7-2 is based on equations of 6 DOF motion of a marine system, see [57] for details. The low frequency motion of the vessel is calculated by the equations in [57] and the wave frequency motion is calculated based on the motion response amplitude operators (RAO). [58] The vessel model used for the simulation is a platform supply vessel (PSV) the parameters of which are shown in Table 7-1. This module will output low frequency motion and wave frequency motion of the vessel while wave, wind and current module are set up and thrust forces and moments are input from the controller.

Table 7-1 Ship parameters of the PSV

Lpp	82.8	Length between perpendiculars (m)
T	6	draught (m)
B	19.2	breadth (m)

m	6.3622085×10^{6}	mass (kg)
rho	1025	density of water (kg/m^3)
g	9.81	acceleration of gravity (m/s^2)
k44	6.84459	radius of gyration in roll (m)
k55	20.74078	radius of gyration in pitch (m)
k66	20.7	radius of gyration in yaw (m)
nabla	$6.207032682926830\times10^{3}$	volume displacement (m^3)
CB	[-5.38600145 0 3.334]	center of buoyancy w.r.t baseline and Lpp/2 (m)
S	1.8201601524×10^{3}	wetted surface (m^2)
GM_T	1.03628×10^{2}	transverse metacentric height (m)

GM_L	2.144	lateral metacentric height (m)
M	$\begin{bmatrix} 7.010149\times10^{6} & 0 & 0 \\ 0 & 8.519007\times10^{6} & 4.718726\times10^{5} \\ 0 & -2.595509\times10^{6} & 3.797291\times10^{9} \end{bmatrix}$	Inertial matrix
D	$\begin{bmatrix} 2.648609\times10^{5} & 0 & 0 \\ 0 & 8.816423\times10^{5} & -1\times10^{7} \\ 0 & -1\times10^{7} & 3.377437\times10^{8} \end{bmatrix}$	Linear damping matrix

7.1.3 Controller Module

The DP controller module in Figure 7-2 is a nonlinear PID set-point controller from MSS. [56] The set-point, estimated low frequency vessel position, heading, velocity and heading rate are input to this controller which will then determine the desired force and moment to keep the position at the set-point. However, the PID parameters must be tuned well to realize a good performance. [59]

7.1.4 Reference System Module

From the marine vessel module in Figure 7-1 the true 6 DOF motions of the vessel CG are output so the motions of any points with known body-fixed frame coordinates can be calculated according to (4.1)-(4.3), which means the true motions of different sensors are known because of the knowledge of sensors location on the vessel. By adding white noises on the specific motions corresponding to the sensor type, the measurements are simulated and then the reference system is built.

7.1.5 Sensor Fusion Module and Observer Module

All available measurements are input to sensor fusion module (by means of SRUKF) and the total 6 DOF motions of CG are estimated by the algorithm in chapter 5. The Kalman filter algorithm discussed in chapter 6 is applied in the observer module which outputs the estimated low frequency 3 DOF motions and bias forces into the controller.

7.2 Simulation

The parameters in the environmental module are set up as an example shown in Table 7-2. The total 6 DOF motions of the vessel in waves are visualized in the animation window in Figure 7-5, and the 3 DOF low frequency motions of the vessel and vectors of the forces as well as moments on the vessel are plotted in Figure 7-6. The initial position of the vessel is [0, 0, 0] (x position, y position and heading) and the velocities are also initialized as zero. The set-point is [-3, 7, -20]. The reference system contains 5 position reference sensors (two DGPS, one hydro-acoustic, one radar scan, and one laser cyscan) and 2 IMUs (and 2 gyroscopes for yaw). The simulation runs for the first 1000s in normal mode where no reference system fails, and the results are shown from Figure 7-8 to Figure 7-20. Subsequently, the simulation runs until 2800s in dead reckoning mode where reference system fails at 1000s.

Table 7-2 Environmental parameters

Wave spectrum type	Jonswap
Significant wave height Hs (m)	4
Wave peak frequency w_0 (rad/s)	0.6
Mean wave direction (degree)	30
Current direction (degree)	20
Current speed (m/s)	0.2
Mean wind speed (m/s)	20
Mean wind angle (degree)	26

Figure 7-5: Animation of the PSV in waves

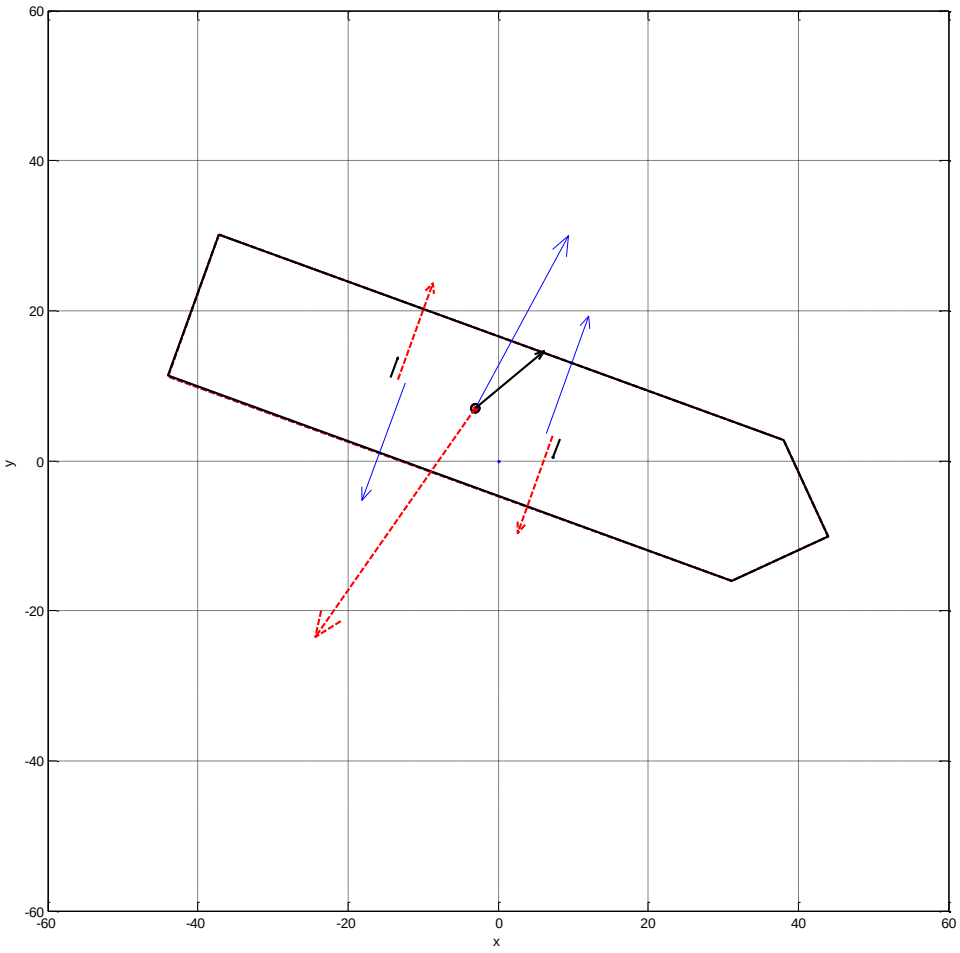


Figure 7-6: DP with available reference system

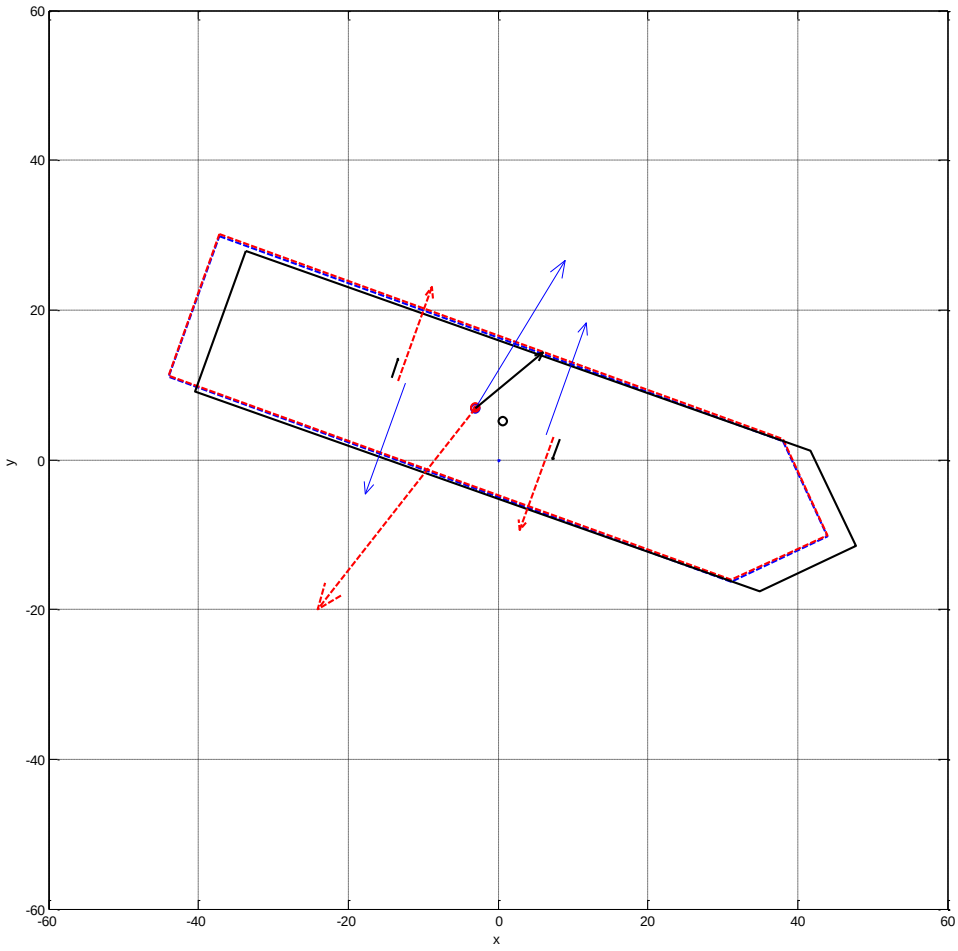


Figure 7-7: Dead reckoning for 1800s after reference system fails

In Figure 7-6 and Figure 7-7, the solid vessel outline is the true position; the blue dotted vessel outline is the estimated position; the red dotted vessel outline is the set point. The three vessels are overlapped in Figure 7-6 because the position and heading are estimated well. Figure 7-7 shows the results of dead reckoning for 1800s. Arrows represent force vectors and the moments are represented by the equivalent coupled forces. The blue arrows are the wind force and moment; the black arrows are the bias force and moment; the dotted red arrows are the thruster force and moment.

The estimated total position, velocity and acceleration of CG are shown from Figure 7-8 to Figure 7-13. The three dimensional errors ($\sqrt{x_{error}^2 + y_{error}^2 + z_{error}^2}$) of translational motions and rotational motions are presented under the conditions of 2 sensor (1 position sensor, 1 IMU), 3 sensor (2 position sensors, 1 IMU), and 7 sensors (5 position sensors, 2 IMUs) in Figure 7-14, Figure 7-15 and Table 7-3 where it is shown that the system with more sensors has more accuracy and the errors of the linear motions are under certain limit without increasing overtime, which means the estimation of the filter is stable.

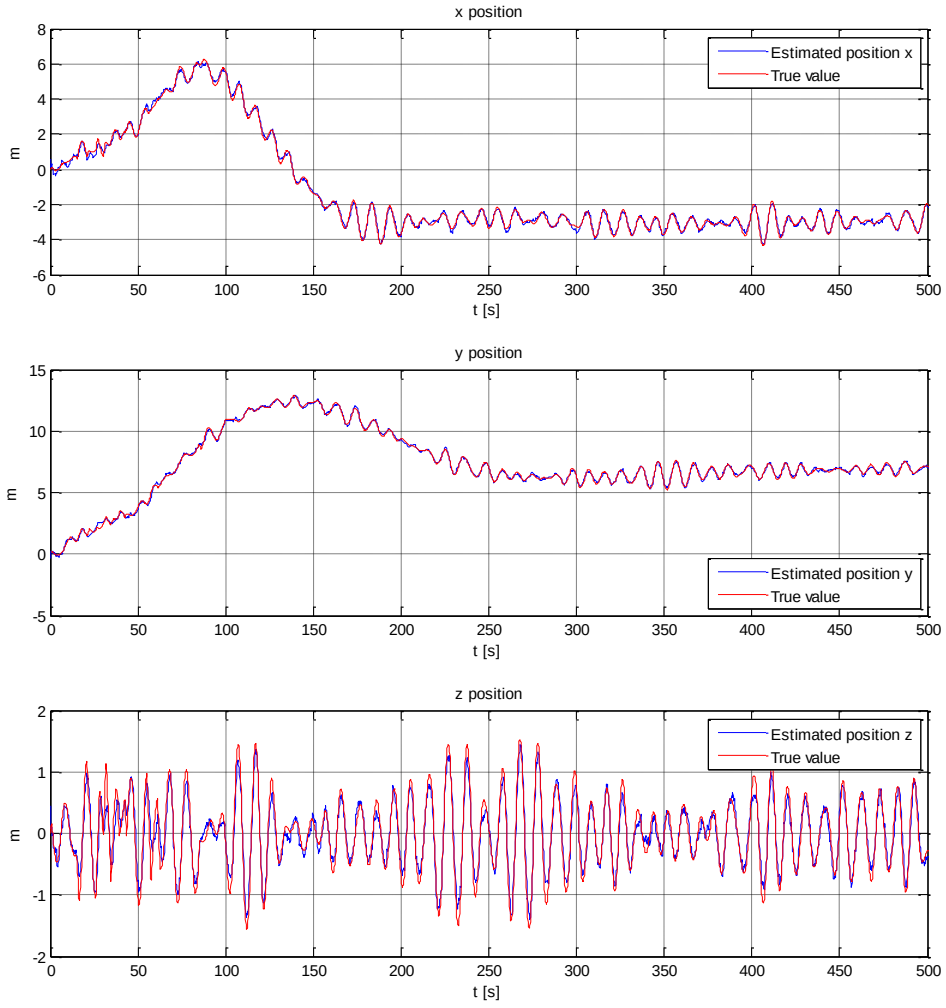


Figure 7-8: SRUKF for the estimation of the total postion of CG in NED frame. At beginning the x, y position of the vessel drifts away under environmental forces because the sytem needs time to start up. Then the vessel moves to the wanted position after a while. The z position is oscillating aroud 0 m due to the waves.

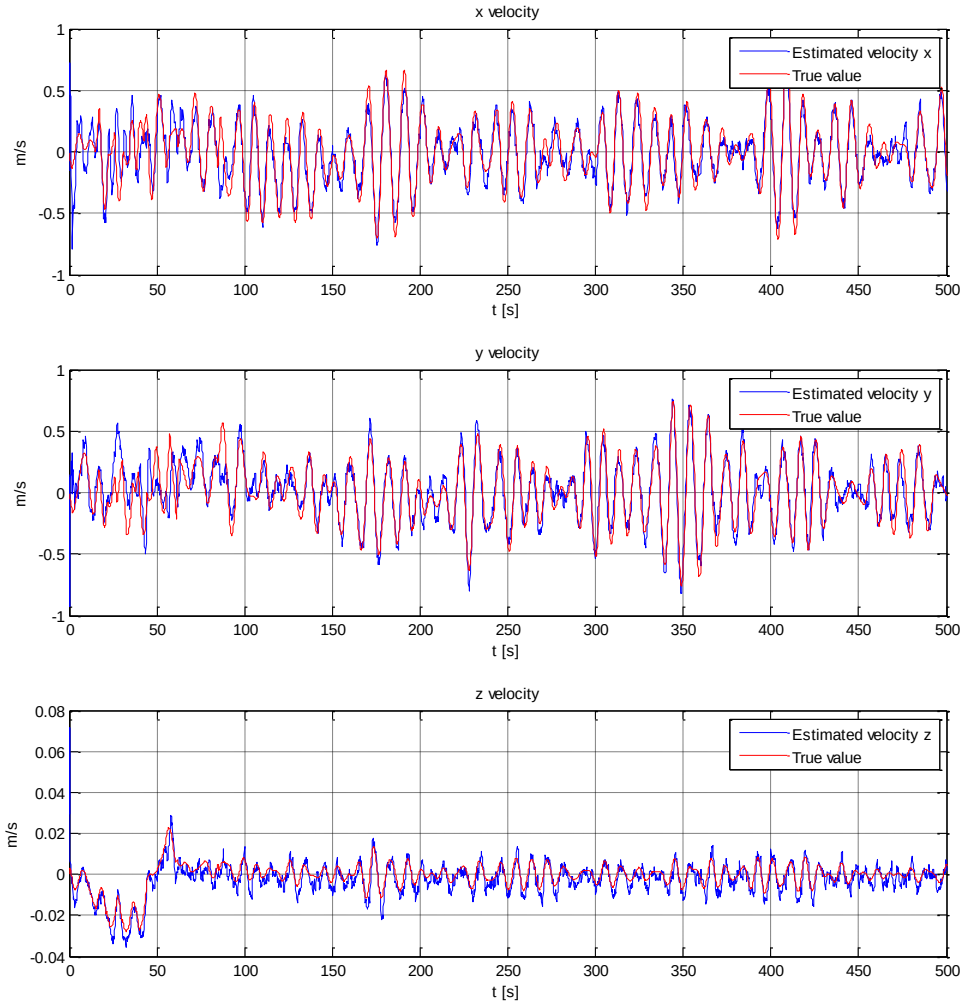


Figure 7-9: SRUKF for the estimation of the total velocity of CG in NED frame. When the vessel is maintaining its position, the velocities in x, y and z direction are oscillating around zero because of first order wave frequency forces.

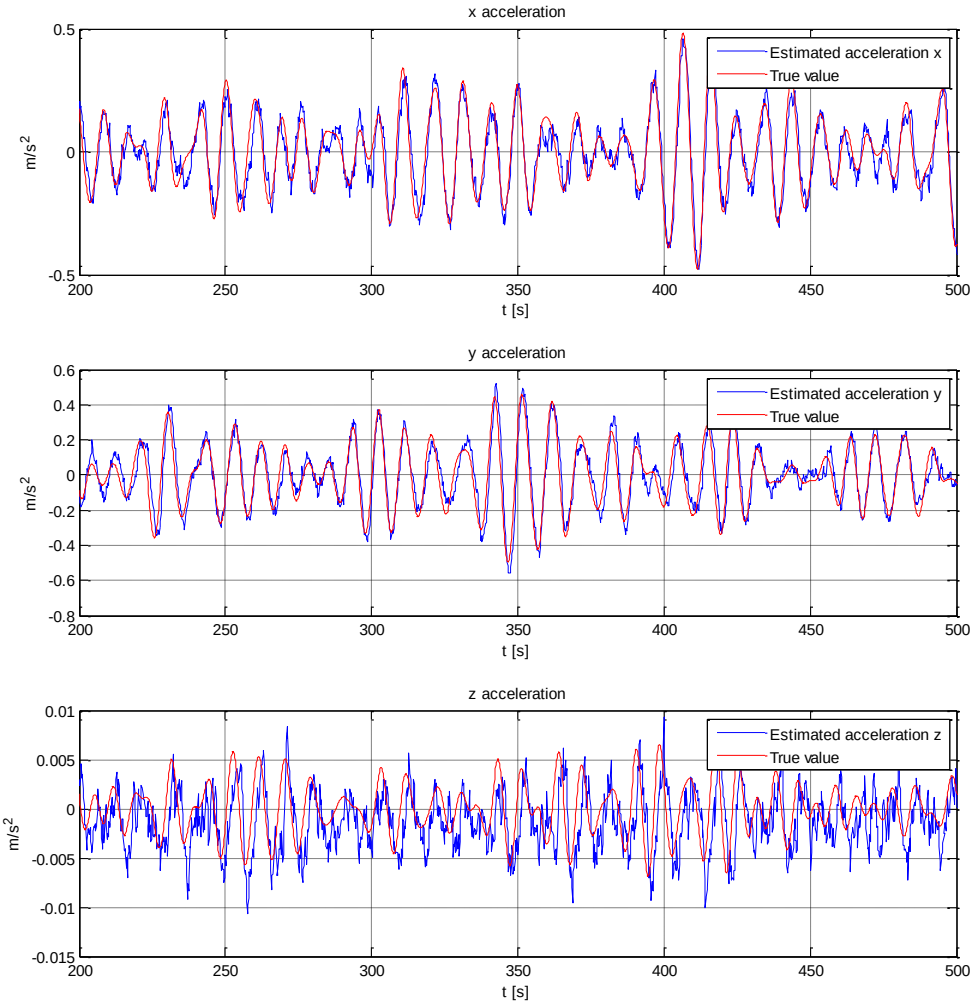


Figure 7-10: SRUKF for estimation of the total acceleration of CG in NED frame

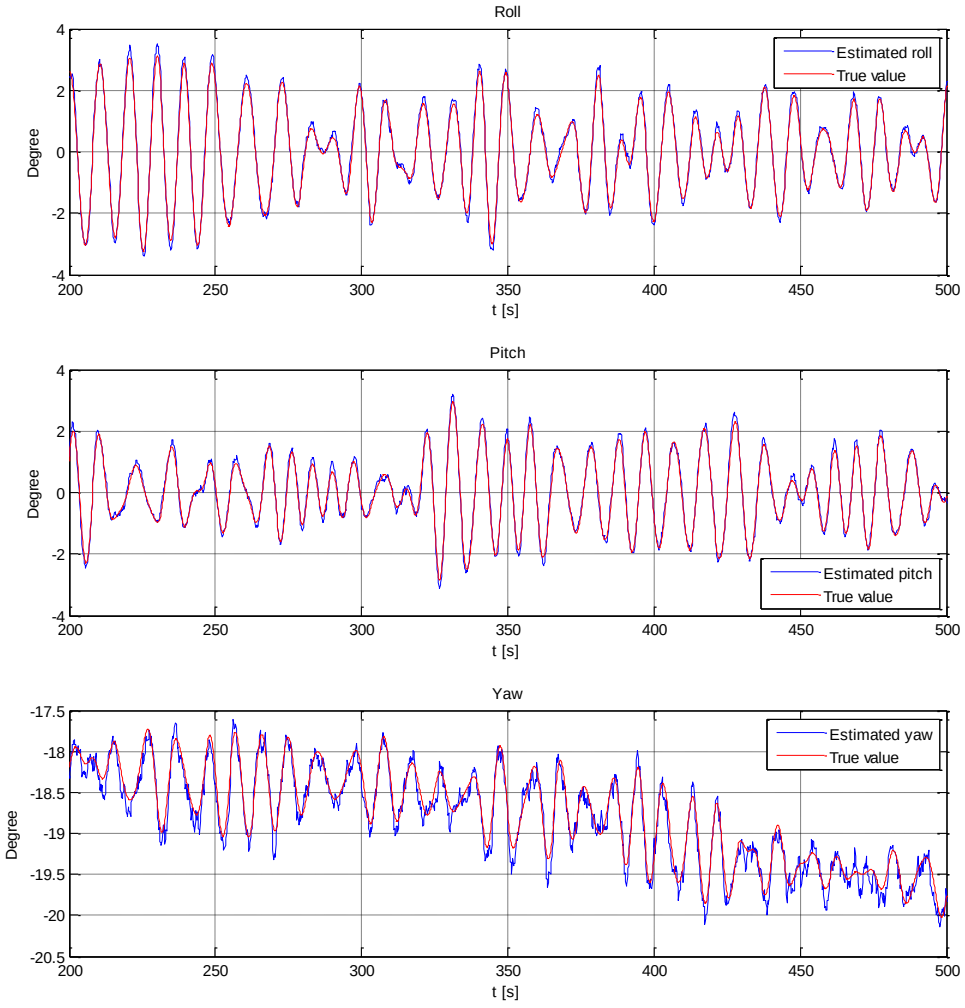


Figure 7-11: SRUKF for total Euler angles estimation

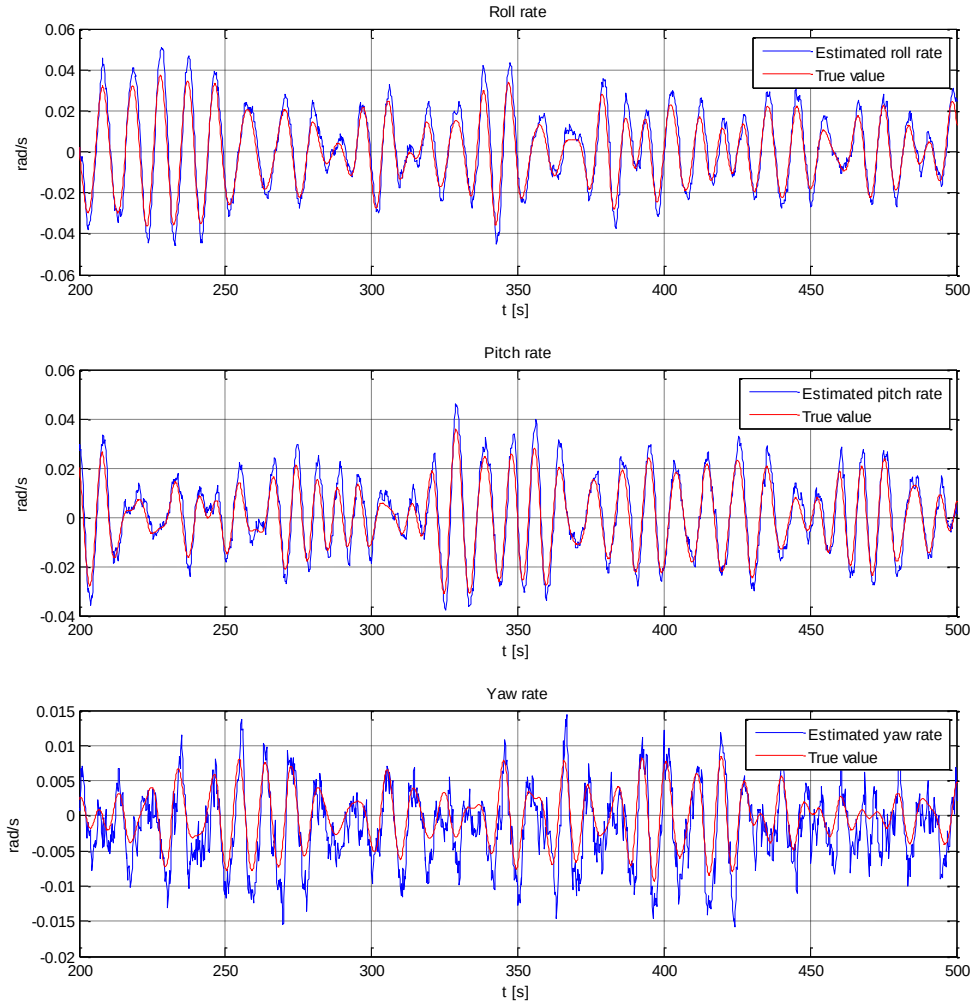


Figure 7-12: SRUKF for total anglar rates estimation

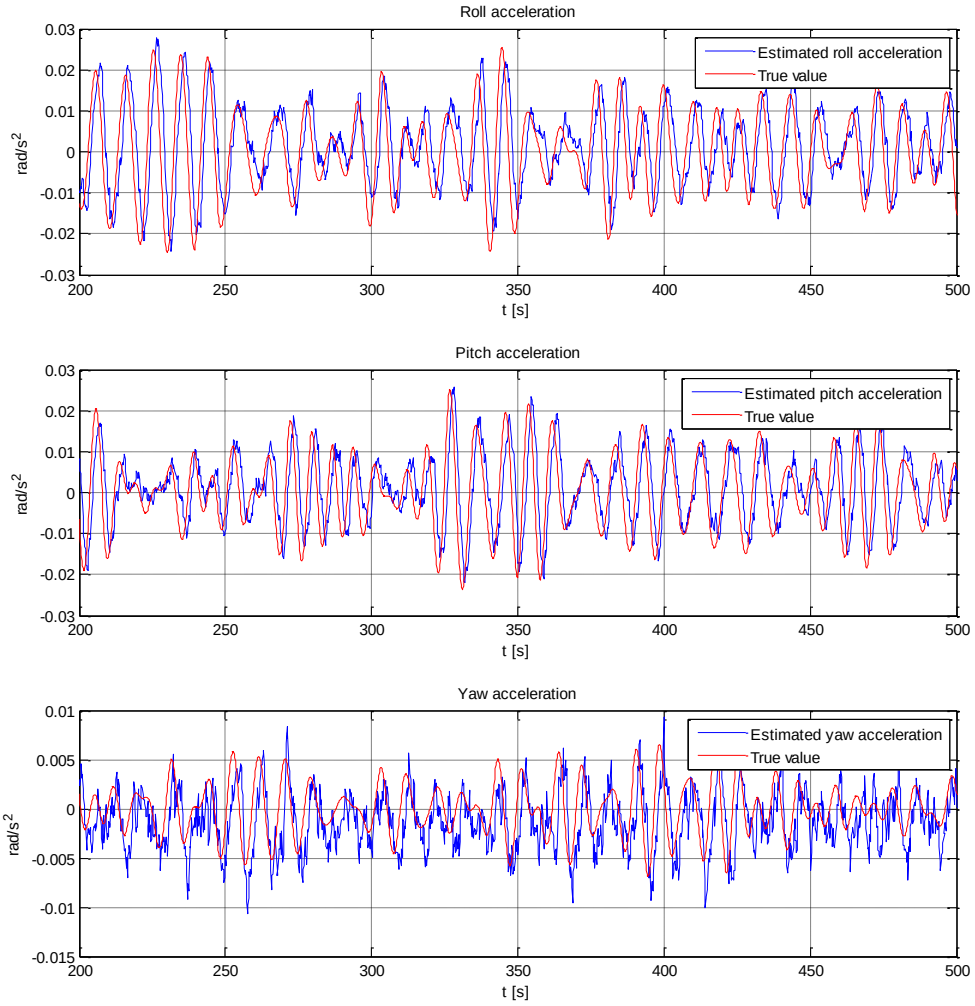


Figure 7-13: SRUKF for total anglar accelerations estimation

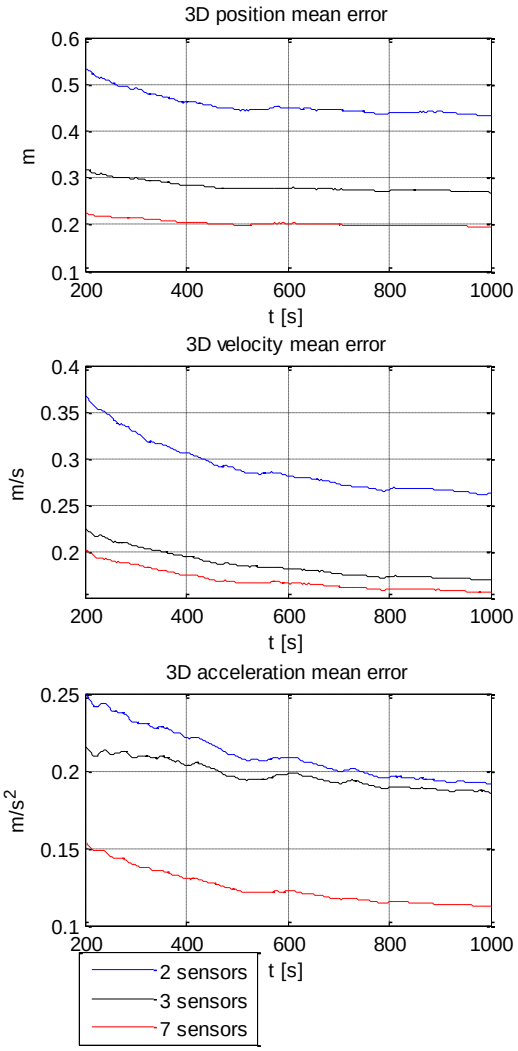


Figure 7-14: 3D mean errors of the estimation for position, velocity and acceleration

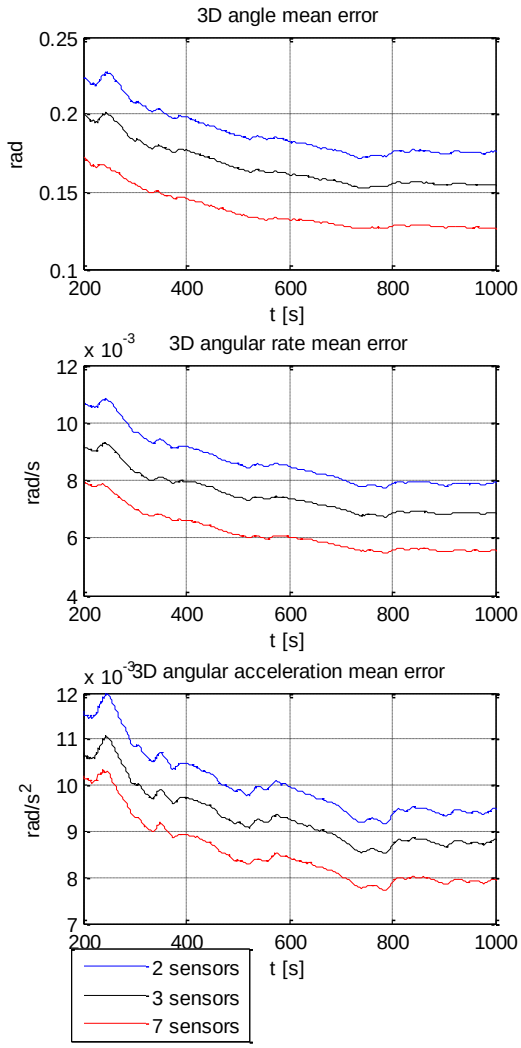
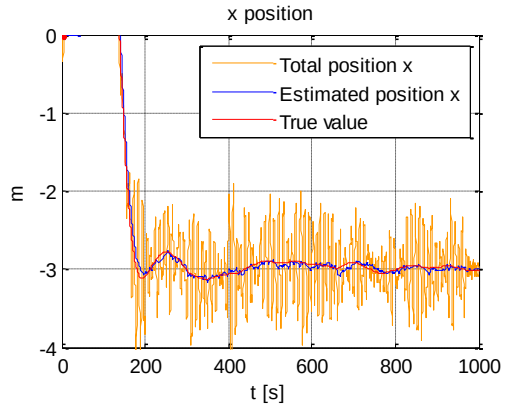


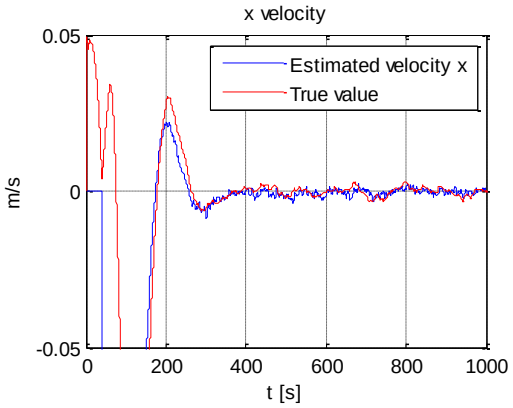
Figure 7-15: 3D mean errors of the estimation for Euler angle, angular rate and angular acceleration

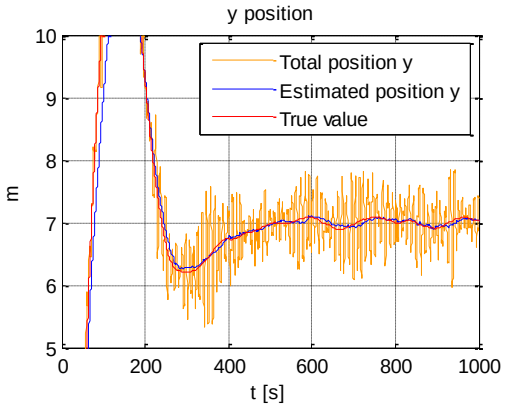
The low frequency motions including positions and velocities are estimated quite good as most wave frequency components are eliminated, see Figure 7-16 and Figure 7-17. The mean two dimensional errors ($\sqrt{x_{error}^2 + y_{error}^2}$) of estimates for position are below 0.15 m and the heading error is below 0.2 degree in Figure 7-18; the 2D errors of estimates for the velocity and heading rate are below 0.004 m/s and 0.01 rad/s respectively in Figure 7-19. The more number of sensors also results in better estimation of the KF, similar with the UKF, since the input of the KF is from the results of the UKF (see Table 7-3). The estimated bias forces are shown in Figure 7-20 where the bias forces estimates gradually capture the true values as the vessel gets stationary.

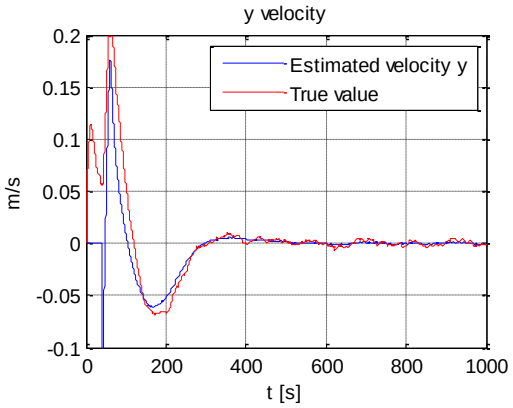
Table 7-3 Mean errors of the SRUKF and the KF for 1000s

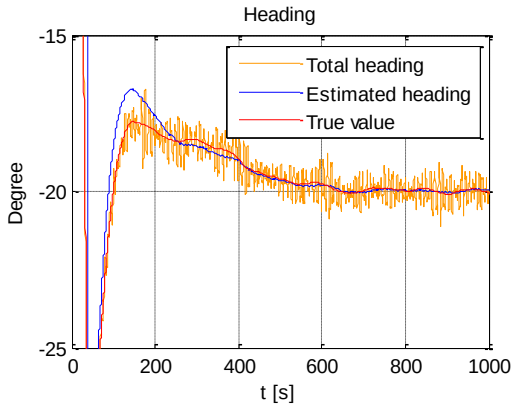
Filters	Errors	1 position sensor 1 IMU	2 position sensors 1 IMU	5 position sensors 2 IMUs
SRUKF	3D position mean error (m)	0.4335	0.2688	0.1944
	3D velocity mean error (m/s)	0.2621	0.1696	0.1555
	3D acceleration mean error (m/s^2)	0.1913	0.1860	0.1123
	3D angular mean error (degree)	0.1762	0.1548	0.1265
	3D angular rate mean error (rad/s)	0.007926	0.006889	0.005550
	3D angular acceleration mean error (rad/s^2)	0.009497	0.008818	0.007957
KF	2D position mean error (m)	0.1094	0.09204	0.07494
	2D velocity mean error (m/s)	0.004413	0.003999	0.003723
	Heading mean error (degree)	0.1167	0.1123	0.1114
	Heading rate mean error (rad/s)	0.004243	0.004218	0.004170











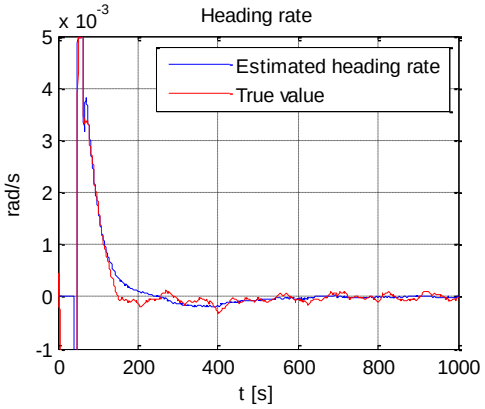
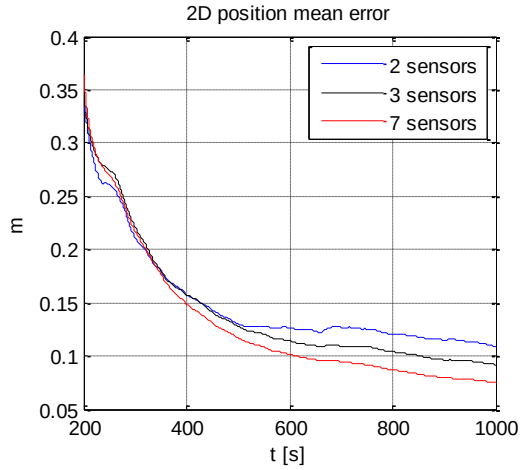
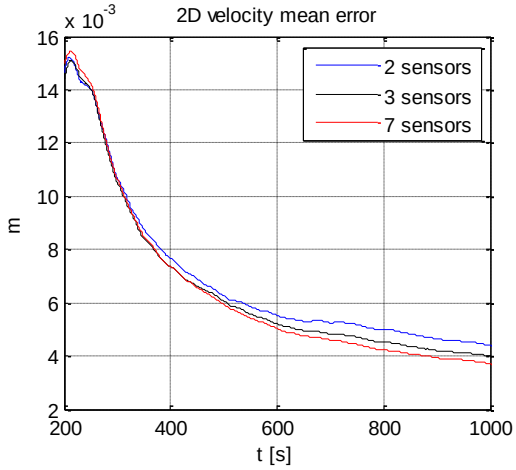
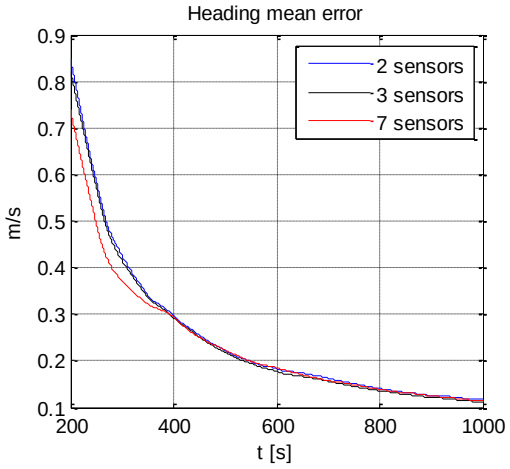


Figure 7-16: KF estimates for low frequency position and heading

Figure 7-17: KF estimates for low frequency velocity and heading rate







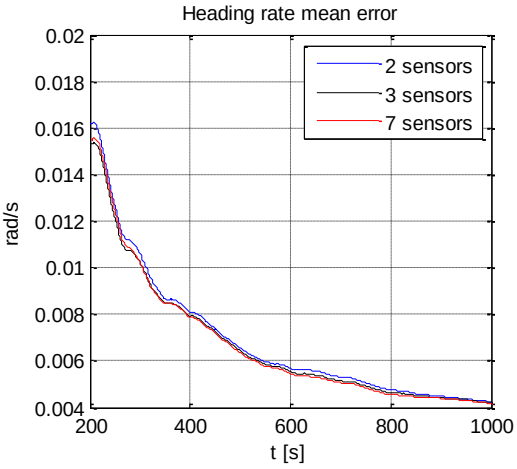


Figure 7-18: Position and heading mean error of KF

Figure 7-19: Velocity and heading rate mean error of KF

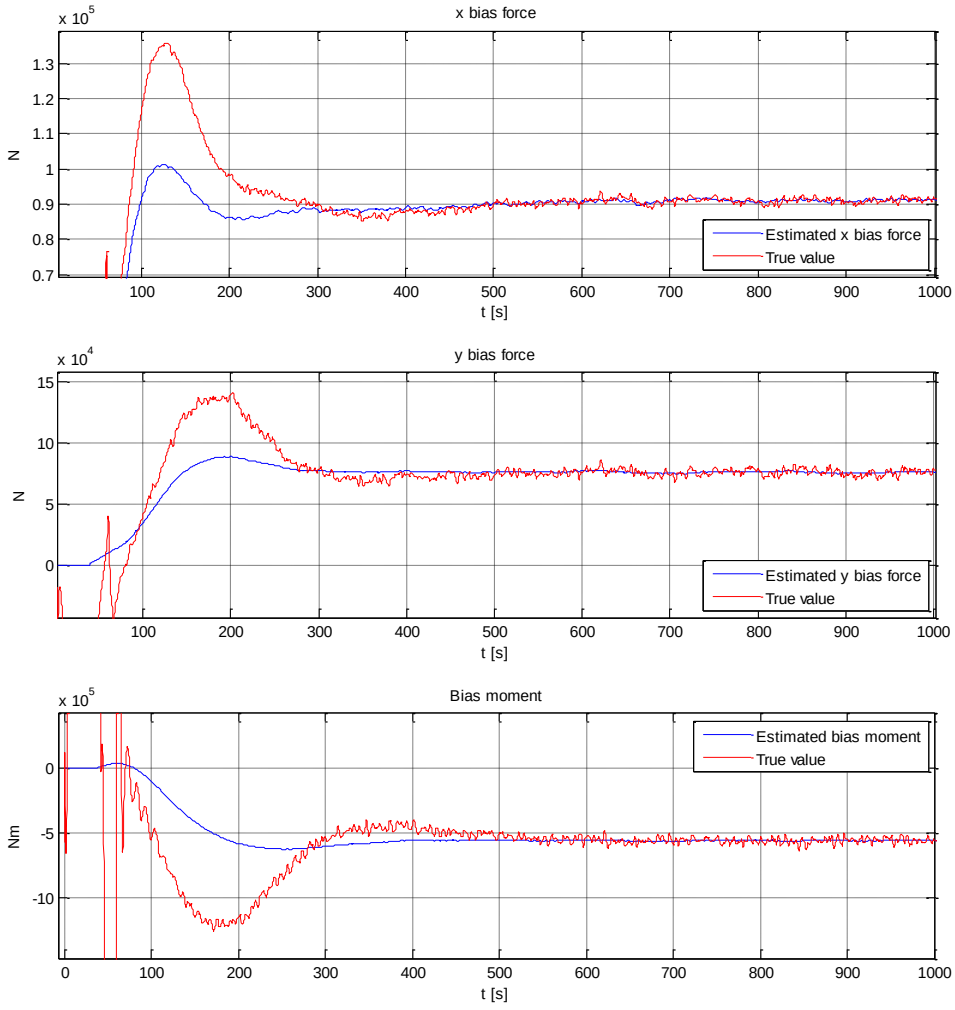


Figure 7-20: Bias forces estimated by KF. The figure shows that the true bias forces are quite noisy and the KF is able to eliminate the noises and ouputs smooth results.

7.3 Conclusion

The 6 DOF motions are estimated correctly in the SRUKF just a few seconds after the simulation starts. More sensors increase the accuracy and redundancy of the DP system if they are fused by the SRUKF. The low frequency 3 DOF motions are estimated well in the KF based on the total motions estimated in SRUKF after approximate 200s and it takes longer before the bias forces are estimated correctly, which indicates that it takes some time before the estimate converges to the correct value. The dead reckoning works well after estimated bias forces become stable as shown in Figure 7-7; the error of the position is only about 5m after reference system failure for 1800s, which reduces the probability of collision with other surrounding structures at sea.

8. Conclusions and Recommendations

In this chapter, the main conclusions of this work are summarized due to the algorithms and simulation result presented in previous chapters. Recommendations for the future work are also presented based on the experience and knowledge obtained in this work. The drawbacks of the algorithms in this thesis are expected to get fixed in the future work.

8.1 Conclusions

The goal of this thesis is to fuse all available reference sensors and design an observer to estimate low frequency motion for dynamic positioning system. Therefore, there are two major procedures to achieve the target. At first step, all sensors are fused base on square-root unscented Kalman filter, which gives an estimate of total motion in 6 DOF about CG. The sensor fusion in the SRUKF improves the accuracy and redundancy of the reference system because all sensors are combined by the mathematical model in the filter. Since the results of the SRUKF contain wave frequency motions which should not be compensated by actuators, an observer based on Kalman filter is implemented at the next step. As the ship mathematical model is assumed linear for rotational speed that is the case in DP operation, standard Kalman filter is a good choice in this condition. Besides, ship mathematical model is combined with measurements such that this observer is capable of dead reckoning using the ship mathematical model when the reference system fails.

In order to test the algorithms in this thesis, a simulation program is built in Simulink. The external environment (wind, current and waves) is generated and the motions about CG of a PSV with known parameters can be calculated as the true motions. Hence, motions of any points on the vessel are also known via reference frame transformation

described in this thesis. Adding noise on the motion of any specified point generates a sensor measurement. Therefore, any type of reference sensor can be simulated in this way. Then the algorithms for sensor fusion and observer are tested by connecting them to the measurements, the results of which are presented in Chapter 7. The simulation results show good performance of cascade of the SRUKF and the KF for redundancy improvement and the low frequency vessel motion estimation in DP.

8.2 Recommendations for Future Work

The performance of both SRUKF and KF is affected by the parameters such as measurement noise covariance and process noise covariance. Consequently, it is crucial to tune these parameters based on our knowledge of the system model and sensor properties. Nevertheless, it is of difficulty to tune the process noise covariance. Furthermore, the filters should be tuned every time they are implemented on different vessels and the sea state changes in order to reach the best performance. Apparently, manual tuning is inconvenient and it should be replaced by other algorithms like self-tuning and adaptive filter in the future work to reduce the cost for DP system installation.

9. Appendices

9.1 Appendix 1 References

[1]. http://dpmarine.dk/dynamic-positioning/.

[2]. Lindegaard, K.-P., *Acceleration feedback in dynamic positioning.* Trondheim, Norway: Norwegian University of Science and Technology, 2003.

[3]. http://www.praxis-automation.nl/.

[4]. Durrant-Whyte, H. and T.C. Henderson, *Multisensor data fusion.* Springer handbook of robotics, 2008: p. 585-610.

[5]. Fossen, T.I. and T. Perez, *Kalman filtering for positioning and heading control of ships and offshore rigs.* Control Systems, IEEE, 2009. **29**(6): p. 32-46.

[6]. Muhammad, S., *Dynamic Positioning of Ships: A nonlinear control design study*. 2012, TU Delft, Delft University of Technology.

[7]. Simon, D., *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. 2006: John Wiley & Sons.

[8]. Paul, Z. and M. Howard, *Fundamentals of Kalman Filtering-A Practical Approach.* Virginia, Published by the American Institute of Aeronautics and Astronautics, 2005.

[9]. Mohinder, S.G. and P.A. Angus, *Kalman filtering: theory and practice using Matlab.* John Wiley & Sons, Inc., New York, 2001: p. 178-181.

[10]. Mayhew, D.M., *Multi-rate sensor fusion for GPS navigation using Kalman filtering*. 1999, Virginia Polytechnic Institute and State University.

[11]. Nassar, S. and N. El-Sheimy, *A combined algorithm of improving INS error modeling and sensor measurements for accurate INS/GPS navigation.* GPS solutions, 2006. **10**(1): p. 29-39.

[12]. Han, S. and J. Wang, *Integrated GPS/INS navigation system with dual-rate Kalman Filter.* GPS solutions, 2012. **16**(3): p. 389-404.

[13]. Van Der Merwe, R., *Sigma-point Kalman filters for probabilistic inference in dynamic state-space models*. 2004, Oregon Health & Science University.

[14]. Van Der Merwe, R. and E.A. Wan. *The square-root unscented Kalman filter for state and parameter-estimation*. in *Acoustics, Speech, and*

Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on. 2001. IEEE.
[15]. Van Der Merwe, R., E.A. Wan, and S. Julier. *Sigma-point Kalman filters for nonlinear estimation and sensor-fusion: Applications to integrated navigation*. in *Proceedings of the AIAA Guidance, Navigation & Control Conference*. 2004.
[16]. Zhang, L.-G., H.-B. Ma, and Y.-Z. Chen. *Square-root unscented Kalman filter for vehicle integrated navigation*. in *Machine Learning and Cybernetics, 2007 International Conference on*. 2007. IEEE.
[17]. Tang, X., J. Yan, and D. Zhong, *Square-root sigma-point Kalman filtering for spacecraft relative navigation.* Acta Astronautica, 2010. **66**(5): p. 704-713.
[18]. Hovland, G., et al., *Nonlinear estimation methods for parameter tracking in power plants.* Control Engineering Practice, 2005. **13**(11): p. 1341-1355.
[19]. Fossen, T.I., *Handbook of marine craft hydrodynamics and motion control*. 2011: John Wiley & Sons.
[20]. Torsetnes, G., *Nonlinear control and observer design for dynamic positioning using contraction theory*. 2004, Master's thesis, Norwegian University of Science and Technology.
[21]. Farrell, J., *Aided navigation: GPS with high rate sensors*. 2008: McGraw-Hill New York.
[22]. *http://what-when-how.com/*.
[23]. *http://logpedia.com/*.
[24]. Diebel, J., *Representing attitude: Euler angles, unit quaternions, and rotation vectors.* Matrix, 2006. **58**: p. 15-16.
[25]. *http://www.damen.nl/*.
[26]. *http://www.shipmotion.se/*.
[27]. *http://www.sonardyne.com/products/positioning.html*.
[28]. *http://www.km.kongsberg.com/*.
[29]. Vickery, K. *Acoustic positioning systems. A practical overview of current systems*. in *Autonomous Underwater Vehicles, 1998. AUV'98. Proceedings of the 1998 Workshop on*. 1998. IEEE.
[30]. Rhodes, M., *Lasers: Position Reference Sensors for DP* DYNAMIC POSITIONING CONFERENCE, 2000.
[31]. *http://marine.guidance.eu.com/RadaScan*.
[32]. *http://en.wikipedia.org/wiki/Low-pass_filter*.
[33]. *http://en.wikipedia.org/wiki/Band-stop_filter*.

[34]. Aguiar, A.P., et al. *A Linear Design Model for Wave Filtering and Dynamic Positioning*. in *CONTROLO'2012*. 2012.
[35]. *http://en.wikipedia.org/wiki/Kalman_filter.*
[36]. Welch, G. and G. Bishop, *An introduction to the Kalman filter*. 1995.
[37]. Crassidis, J.L., *Sigma-point Kalman filtering for integrated GPS and inertial navigation.* Aerospace and Electronic Systems, IEEE Transactions on, 2006. **42**(2): p. 750-756.
[38]. György, K., A. Kelemen, and L. Dávid, *Unscented Kalman Filters and Particle Filter Methods for Nonlinear State Estimation.* Procedia Technology, 2014. **12**: p. 65-74.
[39]. Julier, S.J. and J.K. Uhlmann. *A new extension of the Kalman filter to nonlinear systems*. in *Int. symp. aerospace/defense sensing, simul. and controls*. 1997. Orlando, FL.
[40]. Press, W. and S. Flannery, *Teukolsky, and WT Vetterling. 1988. Numerical Recipes in C; The Art of Scientific Computing*. 1971, Cambridge University Press, Cambridge.
[41]. Wilkinson, J.H., J.H. Wilkinson, and J.H. Wilkinson, *The algebraic eigenvalue problem*. Vol. 87. 1965: Clarendon Press Oxford.
[42]. Gander, W. *Algorithms for the QR decomposition*. in *Seminar für Angewandte Mathematik: Research report*. 1980.
[43]. Buttkus, B., *Spectral Analysis and Filter Theory in Applied Geophysics: With 23 Tables*. 2000: Springer.
[44]. Lemon, K. and B.W. Welch, *Comparison of Nonlinear Filtering Techniques for Lunar Surface Roving Navigation.* Glenn Research Center, NASA, Cleveland, Ohio, NASA/TM-2008-215152, 2008.
[45]. Bachmann, C., *Multi-Sensor Data Fusion for Traffic Speed and Travel Time Estimation*. 2011.
[46]. Sanchez-Meca, J. and F. Marín-Martínez, *Weighting by inverse variance or by sample size in meta-analysis: a simulation study.* Educational and Psychological Measurement, 1998. **58**(2): p. 211-220.
[47]. Fannemel, Å.V., *Dynamic positioning by nonlinear model predictive control.* 2008.
[48]. Fossen, T.I., *Guidance and control of ocean vehicles*. Vol. 199. 1994: Wiley New York.
[49]. Balchen, J.G., N.A. Jenssen, and S. Sælid. *Dynamic positioning using Kalman filtering and optimal control theory*. in *IFAC/IFIP symposium on automation in offshore oil field operation*. 1976.
[50]. Snijders, J., *Wave filtering and Thruster allocation for dynamic positioned ships.* Delft University of Technology, 2005.

[51]. Andersson, T., *Parameter Estimation and Waveform Fitting for Narrowband Signals.* 2005.
[52]. Hespanha, J.P., *Linear systems theory*. 2009: Princeton university press.
[53]. Butcher, J.C., *The numerical analysis of ordinary differential equations: Runge-Kutta and general linear methods*. 1987: Wiley-Interscience.
[54]. Jinli, X., L. Mingjiun, and X. Yanmin, *A modified square-root unscented Kalman filter restraining outliers.* 2012.
[55]. T. Perez, Ø.N.S., T.I. Fossen, and A.J. Sørensen, *An Overview of the Marine Systems Simulator (MISS): A Simulink Toolbox for Marine Control Systems.*
[56]. *MSS, Marine Systems Simulator. Available online at http://www.marinecontrol.org.*
[57]. Brodtkorb, A.H., *Dynamic Positioning in Extreme Sea States.* 2014.
[58]. Journée, J. and W. Massie, *Offshore hydromechanics*. 2000: TU Delft.
[59]. Cominos, P. and N. Munro, *PID controllers: recent tuning methods and design to specification.* IEE Proceedings-Control Theory and Applications, 2002. **149**(1): p. 46-53.

9.2 Appendix 2 Matlab/Simulink

9.2.1 Signal Generator

```
function out = signal(t,tt,eta,nu,a)
g=[0;0;9.81];
psi=eta(4);
theta=eta(5);
phi=eta(6);

dpsi=nu(4);
dtheta=nu(5);
dphi=nu(6);

ddpsi=a(4);
ddtheta=a(5);
ddphi=a(6);

R=[ cos(phi)*cos(theta), cos(phi)*sin(psi)*sin(theta) -
cos(psi)*sin(phi), sin(phi)*sin(psi) + cos(phi)*cos(psi)*sin(theta)
 cos(theta)*sin(phi), cos(phi)*cos(psi) + sin(phi)*sin(psi)*sin(theta),
cos(psi)*sin(phi)*sin(theta) - cos(phi)*sin(psi)
         -sin(theta),                                  cos(theta)*sin(psi),
cos(psi)*cos(theta)];

 invR=[                                cos(phi)*cos(theta),
cos(theta)*sin(phi),          -sin(theta)
 cos(phi)*sin(psi)*sin(theta) - cos(psi)*sin(phi), cos(phi)*cos(psi) +
sin(phi)*sin(psi)*sin(theta), cos(theta)*sin(psi)
 sin(phi)*sin(psi) + cos(phi)*cos(psi)*sin(theta),
cos(psi)*sin(phi)*sin(theta) - cos(phi)*sin(psi), cos(psi)*cos(theta)];


dR=[ - dphi*cos(theta)*sin(phi) - dtheta*cos(phi)*sin(theta),
dpsi*sin(phi)*sin(psi) - dphi*cos(phi)*cos(psi) +
dpsi*cos(phi)*cos(psi)*sin(theta) + dtheta*cos(phi)*cos(theta)*sin(psi)
- dphi*sin(phi)*sin(psi)*sin(theta), dphi*cos(phi)*sin(psi) +
dpsi*cos(psi)*sin(phi) + dtheta*cos(phi)*cos(psi)*cos(theta) -
dphi*cos(psi)*sin(phi)*sin(theta) - dpsi*cos(phi)*sin(psi)*sin(theta)
   dphi*cos(phi)*cos(theta) - dtheta*sin(phi)*sin(theta),
dphi*cos(phi)*sin(psi)*sin(theta) - dpsi*cos(phi)*sin(psi) -
dphi*cos(psi)*sin(phi) + dpsi*cos(psi)*sin(phi)*sin(theta) +
dtheta*cos(theta)*sin(phi)*sin(psi), dphi*sin(phi)*sin(psi) -
dpsi*cos(phi)*cos(psi) + dphi*cos(phi)*cos(psi)*sin(theta) +
dtheta*cos(psi)*cos(theta)*sin(phi) - dpsi*sin(phi)*sin(psi)*sin(theta)
                                     -dtheta*cos(theta),
dpsi*cos(psi)*cos(theta) - dtheta*sin(psi)*sin(theta),
- dpsi*cos(theta)*sin(psi) - dtheta*cos(psi)*sin(theta)];
```

```
ddR=[ dphi*dtheta*sin(phi)*sin(theta) - ddtheta*cos(phi)*sin(theta) -
dphi*dphi*cos(phi)*cos(theta) - dtheta*dtheta*cos(phi)*cos(theta) -
ddphi*cos(theta)*sin(phi) + dtheta*dphi*sin(phi)*sin(theta),
ddpsi*sin(phi)*sin(psi) - ddphi*cos(phi)*cos(psi) +
ddpsi*cos(phi)*cos(psi)*sin(theta) +
ddtheta*cos(phi)*cos(theta)*sin(psi) -
ddphi*sin(phi)*sin(psi)*sin(theta) + dphi*dphi*cos(psi)*sin(phi) +
dphi*dpsi*cos(phi)*sin(psi) + dpsi*dphi*cos(phi)*sin(psi) +
dpsi*dpsi*cos(psi)*sin(phi) - dphi*dphi*cos(phi)*sin(psi)*sin(theta) -
dphi*dpsi*cos(psi)*sin(phi)*sin(theta) -
dphi*dtheta*cos(theta)*sin(phi)*sin(psi) -
dpsi*dphi*cos(psi)*sin(phi)*sin(theta) -
dpsi*dpsi*cos(phi)*sin(psi)*sin(theta) -
dtheta*dphi*cos(theta)*sin(phi)*sin(psi) -
dtheta*dtheta*cos(phi)*sin(psi)*sin(theta) +
dpsi*dtheta*cos(phi)*cos(psi)*cos(theta) +
dtheta*dpsi*cos(phi)*cos(psi)*cos(theta), ddphi*cos(phi)*sin(psi) +
ddpsi*cos(psi)*sin(phi) + ddtheta*cos(phi)*cos(psi)*cos(theta) -
ddphi*cos(psi)*sin(phi)*sin(theta) - ddpsi*cos(phi)*sin(psi)*sin(theta)
+ dphi*dpsi*cos(phi)*cos(psi) + dpsi*dphi*cos(phi)*cos(psi) -
dphi*dphi*sin(phi)*sin(psi) - dpsi*dpsi*sin(phi)*sin(psi) +
dphi*dpsi*sin(phi)*sin(psi)*sin(theta) +
dpsi*dphi*sin(phi)*sin(psi)*sin(theta) -
dphi*dphi*cos(phi)*cos(psi)*sin(theta) -
dphi*dtheta*cos(psi)*cos(theta)*sin(phi) -
dpsi*dpsi*cos(phi)*cos(psi)*sin(theta) -
dpsi*dtheta*cos(phi)*cos(theta)*sin(psi) -
dtheta*dphi*cos(psi)*cos(theta)*sin(phi) -
dtheta*dpsi*cos(phi)*cos(theta)*sin(psi) -
dtheta*dtheta*cos(phi)*cos(psi)*sin(theta)
ddphi*cos(phi)*cos(theta) - ddtheta*sin(phi)*sin(theta) -
dphi*dphi*cos(theta)*sin(phi) - dphi*dtheta*cos(phi)*sin(theta) -
dtheta*dphi*cos(phi)*sin(theta) - dtheta*dtheta*cos(theta)*sin(phi),
ddphi*cos(phi)*sin(psi)*sin(theta) - ddpsi*cos(phi)*sin(psi) -
ddphi*cos(psi)*sin(phi) + ddpsi*cos(psi)*sin(phi)*sin(theta) +
ddtheta*cos(theta)*sin(phi)*sin(psi) - dphi*dphi*cos(phi)*cos(psi) -
dpsi*dpsi*cos(phi)*cos(psi) + dphi*dpsi*sin(phi)*sin(psi) +
dpsi*dphi*sin(phi)*sin(psi) - dphi*dphi*sin(phi)*sin(psi)*sin(theta) -
dpsi*dpsi*sin(phi)*sin(psi)*sin(theta) -
dtheta*dtheta*sin(phi)*sin(psi)*sin(theta) +
dphi*dpsi*cos(phi)*cos(psi)*sin(theta) +
dphi*dtheta*cos(phi)*cos(theta)*sin(psi) +
dpsi*dphi*cos(phi)*cos(psi)*sin(theta) +
dpsi*dtheta*cos(psi)*cos(theta)*sin(phi) +
dtheta*dphi*cos(phi)*cos(theta)*sin(psi) +
dtheta*dpsi*cos(psi)*cos(theta)*sin(phi), ddphi*sin(phi)*sin(psi) -
ddpsi*cos(phi)*cos(psi) + ddphi*cos(phi)*cos(psi)*sin(theta) +
ddtheta*cos(psi)*cos(theta)*sin(phi) -
ddpsi*sin(phi)*sin(psi)*sin(theta) + dphi*dphi*cos(phi)*sin(psi) +
dphi*dpsi*cos(psi)*sin(phi) + dpsi*dphi*cos(psi)*sin(phi) +
dpsi*dpsi*cos(phi)*sin(psi) - dphi*dphi*cos(psi)*sin(phi)*sin(theta) -
dphi*dpsi*cos(phi)*sin(psi)*sin(theta) -
```

```
dpsi*dphi*cos(phi)*sin(psi)*sin(theta) -
dpsi*dpsi*cos(psi)*sin(phi)*sin(theta) -
dpsi*dtheta*cos(theta)*sin(phi)*sin(psi) -
dtheta*dpsi*cos(theta)*sin(phi)*sin(psi) -
dtheta*dtheta*cos(psi)*sin(phi)*sin(theta) +
dphi*dtheta*cos(phi)*cos(psi)*cos(theta) +
dtheta*dphi*cos(phi)*cos(psi)*cos(theta)

dtheta*dtheta*sin(theta) - ddtheta*cos(theta),
ddpsi*cos(psi)*cos(theta) - ddtheta*sin(psi)*sin(theta) -
dpsi*dpsi*cos(theta)*sin(psi) - dpsi*dtheta*cos(psi)*sin(theta) -
dtheta*dpsi*cos(psi)*sin(theta) - dtheta*dtheta*cos(theta)*sin(psi),
dpsi*dtheta*sin(psi)*sin(theta) - ddtheta*cos(psi)*sin(theta) -
dpsi*dpsi*cos(psi)*cos(theta) - dtheta*dtheta*cos(psi)*cos(theta) -
ddpsi*cos(theta)*sin(psi) + dtheta*dpsi*sin(psi)*sin(theta)];


P0_0=eta(1:3);
V0_0=R*nu(1:3);
a0_0=R*a(1:3);

P0_1=[-5.3859196;0;7.3000002];%CoG coordinate. 1 is body frame
Pmru_1=[-2;-2.8;-3.7];
Pgps_1=[3;6;30];

Pmru_0=R*(Pmru_1-P0_1)+P0_0;
Pgps_0=R*(Pgps_1-P0_1)+P0_0;

Vmru_0=dR*(Pmru_1-P0_1)+V0_0;
Vgps_0=dR*(Pgps_1-P0_1)+V0_0;

amru_0=ddR*(Pmru_1-P0_1)+a0_0;
agps_0=ddR*(Pgps_1-P0_1)+a0_0;
amru_1=invR*(amru_0-g);%gravity


out.ts=t;
out.tt=tt;

out.varPGPS=[0.2^2;0.2^2;0.2^2];%GPS position error variance
out.varVGPS=[0.5^2;0.5^2;0.5^2];
out.varA=[0.1^2;0.1^2;0.1^2];
out.varAngle=[0.001^2;0.001^2;0.001^2]*10;
out.varDangle=[0.02^2;0.02^2;0.02^2];

out.Pmru_1.x=Pmru_1(1);
out.Pmru_1.y=Pmru_1(2);
out.Pmru_1.z=Pmru_1(3);

out.Pgps_1.x=Pgps_1(1);
```

```
out.Pgps_1.y=Pgps_1(2);
out.Pgps_1.z=Pgps_1(3);

out.P0_1.x=P0_1(1);
out.P0_1.y=P0_1(2);
out.P0_1.z=P0_1(3);

out.angle.x=psi;
out.angle.y=theta;
out.angle.z=phi;

out.anglen.x=psi+sqrt(out.varAngle(1))*randn;
out.anglen.y=theta+sqrt(out.varAngle(1))*randn;
out.anglen.z=phi+sqrt(out.varAngle(1))*randn;

out.dangle.x=dpsi;
out.dangle.y=dtheta;
out.dangle.z=dphi;

out.danglen.x=dpsi+sqrt(out.varDangle(1))*randn+1;
out.danglen.y=dtheta+sqrt(out.varDangle(1))*randn+1;
out.danglen.z=dphi+sqrt(out.varDangle(1))*randn-1;

out.ddangle.x=ddpsi;
out.ddangle.y=ddtheta;
out.ddangle.z=ddphi;

% % % % % % % % % % % % % % % %

out.P0_0.x=P0_0(1,1);
out.P0_0.y=P0_0(2,1);
out.P0_0.z=P0_0(3,1);

out.V0_0.x=V0_0(1,1);
out.V0_0.y=V0_0(2,1);
out.V0_0.z=V0_0(3,1);

% out.a0_0.x=a0_0(1,1);
% out.a0_0.y=a0_0(2,1);
% out.a0_0.z=a0_0(3,1);
out.a0_0.x=a0_0(1,1);
out.a0_0.y=a0_0(2,1);
out.a0_0.z=a0_0(3,1);
% % % % % % % % % % % % % % % % % % % % % % % % % %

% % % % % % % % % % % % % % % % % % % % % % % % % % % % 
out.amru_1.x=amru_1(1,1);
out.amru_1.y=amru_1(2,1);
out.amru_1.z=amru_1(3,1);
```

```
out.anmru_1.x=amru_1(1,1)+sqrt(out.varA(1))*randn+0.2;%bias
out.anmru_1.y=amru_1(2,1)+sqrt(out.varA(2))*randn+0.001;%bias
out.anmru_1.z=amru_1(3,1)+sqrt(out.varA(3))*randn+0.003;%bias

out.amru_0.x=amru_0(1,1);
out.amru_0.y=amru_0(2,1);
out.amru_0.z=amru_0(3,1);
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
out.Pgps_0.x=Pgps_0(1,1);
out.Pgps_0.y=Pgps_0(2,1);
out.Pgps_0.z=Pgps_0(3,1);

out.Pngps_0.x=Pgps_0(1,1)+sqrt(out.varPGPS(1))*randn;
out.Pngps_0.y=Pgps_0(2,1)+sqrt(out.varPGPS(2))*randn;
out.Pngps_0.z=Pgps_0(3,1)+sqrt(out.varPGPS(3))*randn;

out.Vgps_0.x=Vgps_0(1,1);
out.Vgps_0.y=Vgps_0(2,1);
out.Vgps_0.z=Vgps_0(3,1);

out.Vngps_0.x=Vgps_0(1,1)+sqrt(out.varVGPS(1))*randn;
out.Vngps_0.y=Vgps_0(2,1)+sqrt(out.varVGPS(2))*randn;
out.Vngps_0.z=Vgps_0(3,1)+sqrt(out.varVGPS(3))*randn;
```

9.2.2 SRUKF Algorithm for Sensor Fusion

```
function [y,IMUbias,lost] =
UNKF(s,GPS,GPS_avl_num,IMU,IMU_avl_num,Radar,Radar_avl_num,IMUbias_in,
Process_ncov)

assert (GPS_avl_num <= 10);
assert (IMU_avl_num <= 10);
assert (Radar_avl_num <= 10);
nGPS=GPS_avl_num;
nIMU=IMU_avl_num;
nRadar=Radar_avl_num;
if nIMU==0
    nIMU=1;%assume there is 1 IMU, its roll and pitch =0
end

if nGPS+nRadar==0
    nGPS=1;%assume there is 1 GPS, when there is no position sensor
end
g=[0;0;9.81];
persistent P;
persistent P1;
persistent P2;
persistent Xe;
```

```
persistent Xe_;%prior Xe

persistent flag;%for pos
persistent flag1;%for heading
persistent count;%for pos
persistent count1;%for heading
persistent lost1;
persistent lost2;
persistent lost3;
persistent lost4;
persistent NUM;
persistent All_position_UKF;
persistent All_IMU_UKF
persistent flag_yaw
if isempty(flag_yaw)
flag_yaw=0;
end;
if isempty(All_position_UKF)
All_position_UKF=1;
end;
if isempty(All_IMU_UKF)
All_IMU_UKF=1;
end;
if isempty(NUM)
NUM=0;
end;
NUM=NUM+1;
if isempty(lost1)
lost1=0;
end;
if isempty(lost2)
lost2=0;
end;
if isempty(lost3)
lost3=0;
end;
if isempty(lost4)
lost4=0;
end;
if isempty(count)
count=0;
end;
if isempty(count1)
count1=0;
end;
if isempty(flag)
flag=0;
end;
if isempty(flag1)
flag1=0;
end;
```

```
nXe=24;

nRn=nGPS*6+nIMU*9+nRadar*3;
nPsub1=12;
nPsub2=nRn;
nPsub=nPsub1+nPsub2;
assert(nRn <= 200);
assert(nPsub <= 200);

if isempty(Xe)
Xe=zeros(nXe,1);
end;
if isempty(Xe_)
Xe_=zeros(nXe,1);
end;

sPsub=eye(nPsub,nPsub);
if isempty(P2)
P2=eye(nXe,nXe);
end;
if isempty(P)
P=10*eye(nXe,nXe);
P=chol(P,'lower');
end;
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
f
% determin lost signal
f=eye(nRn);




% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% %

alfa =0.05;%alfa =0.05;I found that alfa has a large influence on the
result,in this case, the smaller the better

kappa =0;
beta =2;
Lsigma =nXe;%+nPsub;
lambada =alfa*alfa*(Lsigma+kappa)-Lsigma;
gamma =alfa*(Lsigma+kappa)^0.5;
sqrt_gamma =gamma;


% % % % % % % % % % % INTI% % % % % % % % % % %
Ym=zeros(1,nRn);
t=s.tt;
if s.ts<20%warm up
    All_position_UKF=1;
```

```
    All_IMU_UKF=1;
end


if nGPS>=1
 for i=1:nGPS
  sPsub(nPsub1+1+6*(i-1):nPsub1+3+6*(i-1),nPsub1+1+6*(i-
1):nPsub1+3+6*(i-1))=diag(GPS(i).varPGPS); %GPS position

  sPsub(nPsub1+4+6*(i-1):nPsub1+6+6*(i-1),nPsub1+4+6*(i-
1):nPsub1+6+6*(i-1))=diag(GPS(i).varVGPS); %GPS velocity

Ym(1+(i-1)*6)=GPS(i).Pngps_0.x;
Ym(2+(i-1)*6)=GPS(i).Pngps_0.y;
Ym(3+(i-1)*6)=GPS(i).Pngps_0.z;
Ym(4+(i-1)*6)=GPS(i).Vngps_0.x;
Ym(5+(i-1)*6)=GPS(i).Vngps_0.y;
Ym(6+(i-1)*6)=GPS(i).Vngps_0.z;

f(1+(i-1)*6:6+(i-1)*6,1+(i-1)*6:6+(i-1)*6)=eye(6)*GPS(i).avl;%if sigal
available

if GPS(i).avl==0
    Ym(1+(i-1)*6:2+(i-1)*6)=Xe_(1:2,1)'+([cos(Xe_(12,1)) -
sin(Xe_(12,1));sin(Xe_(12,1)) cos(Xe_(12,1))]...
                                        *[GPS(i).Pgps_1.x-s.P0_1.x;
GPS(i).Pgps_1.y-s.P0_1.y])';%!!!!!!!!!!!!!!do not forget Rz
    Ym(3+(i-1)*6)=GPS(i).Pgps_1.z-s.P0_1.z;
    Ym(4+(i-1)*6:6+(i-1)*6)=zeros(1,3);
    f(1+(i-1)*6:6+(i-1)*6,1+(i-1)*6:6+(i-
1)*6)=eye(6)*(All_position_UKF);
else
    Xe_(1:6)=Xe(1:6);

end

 end

end


 if nIMU>=1

 for i=1:nIMU
  sPsub(nPsub1+1+nGPS*6+9*(i-1):nPsub1+3+nGPS*6+9*(i-
1),nPsub1+1+nGPS*6+9*(i-1):nPsub1+3+nGPS*6+9*(i-
1))=diag(IMU(i).varA); % accel
```

```
  sPsub(nPsub1+4+nGPS*6+9*(i-1):nPsub1+6+nGPS*6+9*(i-
1),nPsub1+4+nGPS*6+9*(i-1):nPsub1+6+nGPS*6+9*(i-
1))=diag(IMU(i).varAngle); %angle

  sPsub(nPsub1+7+nGPS*6+9*(i-1):nPsub1+9+nGPS*6+9*(i-
1),nPsub1+7+nGPS*6+9*(i-1):nPsub1+9+nGPS*6+9*(i-
1))=diag(IMU(i).varDangle); %dangle

Ym(1+nGPS*6+9*(i-1))=IMU(i).anmru_1.x;
Ym(2+nGPS*6+9*(i-1))=IMU(i).anmru_1.y;
Ym(3+nGPS*6+9*(i-1))=IMU(i).anmru_1.z;

Ym(4+nGPS*6+9*(i-1))=IMU(i).anglen.x;
Ym(5+nGPS*6+9*(i-1))=IMU(i).anglen.y;
Ym(6+nGPS*6+9*(i-1))=IMU(i).anglen.z;

Ym(7+nGPS*6+9*(i-1))=IMU(i).danglen.x;
Ym(8+nGPS*6+9*(i-1))=IMU(i).danglen.y;
Ym(9+nGPS*6+9*(i-1))=IMU(i).danglen.z;

f(1+nGPS*6+9*(i-1):9+nGPS*6+9*(i-1),1+nGPS*6+9*(i-1):9+nGPS*6+9*(i-
1))=eye(9)*IMU(i).avl;
    if IMU(i).avl==0
  sPsub(nPsub1+1+nGPS*6+9*(i-1):nPsub1+3+nGPS*6+9*(i-
1),nPsub1+1+nGPS*6+9*(i-1):nPsub1+3+nGPS*6+9*(i-
1))=diag(IMU(i).varA); % accel

  sPsub(nPsub1+4+nGPS*6+9*(i-1):nPsub1+6+nGPS*6+9*(i-
1),nPsub1+4+nGPS*6+9*(i-1):nPsub1+6+nGPS*6+9*(i-
1))=diag(IMU(i).varAngle); %angle

  sPsub(nPsub1+7+nGPS*6+9*(i-1):nPsub1+9+nGPS*6+9*(i-
1),nPsub1+7+nGPS*6+9*(i-1):nPsub1+9+nGPS*6+9*(i-
1))=diag(IMU(i).varDangle); %dangle
    Ym(1+nGPS*6+9*(i-1):3+nGPS*6+9*(i-1))=-
g';%!!!!!!!!!!!!!!!!!!!!!!!!not zero here
    Ym(4+nGPS*6+9*(i-1):5+nGPS*6+9*(i-1))=zeros(2,1)';%roll pitch
forced to 0
    Ym(6+nGPS*6+9*(i-1))=Xe(12,1);%yaw not Xe_
    Ym(7+nGPS*6+9*(i-1):9+nGPS*6+9*(i-1))=zeros(3,1)';
    IMUbias_in(i).ab=zeros(3,1);
    IMUbias_in(i).wb=zeros(3,1);
    f(1+nGPS*6+9*(i-1):9+nGPS*6+9*(i-1),1+nGPS*6+9*(i-
1):9+nGPS*6+9*(i-1))=eye(9)*All_IMU_UKF;
         if nGPS+nRadar>1 %2 position sensor determin yaw
         sPsub(nPsub1+6+nGPS*6+9*(i-1),nPsub1+6+nGPS*6+9*(i-
1))=sPsub(nPsub1+6+nGPS*6+9*(i-1),nPsub1+6+nGPS*6+9*(i-1))*1e10; %yaw
Changing cov will cause abruption

         flag_yaw=flag_yaw+1;
         else
```

```
        flag_yaw=0;
        end
    else

    Xe_(7:15)=Xe(7:15);
    flag_yaw=0;
    end
 end

 end

 if nRadar>=1

 for i=1:nRadar
  sPsub(nPsub1+1+nGPS*6+nIMU*9+3*(i-1):nPsub1+3+nGPS*6+nIMU*9+3*(i-
1),...
      nPsub1+1+nGPS*6+nIMU*9+3*(i-1):nPsub1+3+nGPS*6+nIMU*9+3*(i-
1))=diag(Radar(i).varPGPS); %Radar or laser or acoustics position

%   sPsub(nPsub1+4+6*(i-1):nPsub1+6+6*(i-1),nPsub1+4+6*(i-
1):nPsub1+6+6*(i-1))=diag(GPS(i).varVGPS); %GPS velocity

Ym(1+nGPS*6+nIMU*9+(i-1)*3)=Radar(i).Pngps_0.x;
Ym(2+nGPS*6+nIMU*9+(i-1)*3)=Radar(i).Pngps_0.y;
Ym(3+nGPS*6+nIMU*9+(i-1)*3)=Radar(i).Pngps_0.z;


f(1+nGPS*6+nIMU*9+(i-1)*3:3+nGPS*6+nIMU*9+(i-1)*3,1+nGPS*6+nIMU*9+(i-
1)*3:3+nGPS*6+nIMU*9+(i-1)*3)=eye(3)*Radar(i).avl;%if sigal available
    if Radar(i).avl==0
    Ym(1+nGPS*6+nIMU*9+(i-1)*3:2+nGPS*6+nIMU*9+(i-
1)*3)=Xe_(1:2,1)'+([cos(Xe_(12,1)) -sin(Xe_(12,1));sin(Xe_(12,1))
cos(Xe_(12,1))]...

*[Radar(i).Pgps_1.x-s.P0_1.x; Radar(i).Pgps_1.y-s.P0_1.y])';%!!!do not
forget Rz
    Ym(3+nGPS*6+nIMU*9+(i-1)*3)=Radar(i).Pgps_1.z-s.P0_1.z;
    f(1+nGPS*6+nIMU*9+(i-1)*3:3+nGPS*6+nIMU*9+(i-
1)*3,1+nGPS*6+nIMU*9+(i-1)*3:3+nGPS*6+nIMU*9+(i-
1)*3)=eye(3)*All_position_UKF;
    else
    Xe_(1:3)=Xe(1:3);

    end

 end

 end

temp1=0;
```

```
if nGPS>=1
for i=1:nGPS
    temp1=GPS(i).avl+temp1;
end
end

if nRadar>=1
for i=1:nRadar
    temp1=Radar(i).avl+temp1;
end
end

temp2=0;

if nIMU>=1
for i=1:nIMU
    temp2=IMU(i).avl+temp2;
end
end



 if temp1==0
    %all position lost
     All_position_UKF=1;
 else
     All_position_UKF=0;
 end

  if temp1==0||(temp1<=1&&temp2==0)
     All_position_lost=1;%without IMU, 1 position sensors canot give a
good result of position. it is worse than dead reckoning

 else
     All_position_lost=0;

 end

  if temp2==0 %with 3 position sensors, euler angles can be calculated.
so IMU lost doesn't mean heading lost

     All_IMU_UKF=1;%all IMU lost
 else

     All_IMU_UKF=0;
  end

  if temp2==0&&temp1<=1 %with 3 position sensors, euler angles can be
calculated. so IMU lost doesn't mean heading lost
    All_heading_lost=1;
```

```
 else
     All_heading_lost=0;

  end

 if flag_yaw==1
     All_heading_lost=1;%to avoid jump in Yaw
     All_position_lost=1;
 end


sPsub(1:12,1:12)=diag(Process_ncov);




% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % %
P1=P;
Xa=zeros(Lsigma,1);
lost=[0 0];
% % % % % % % % % % % % % % % % % % %position recovered, do not inform
KF immediately, but wait a few
minutes % % % % % % % % % % % % % % % % % % % % % % % % % % % %
lost1=All_position_lost;
if lost1==0&&lost2==1
    flag=1;
else

end

if flag==1
   count=count+1;
end

%position is lost, then recovered. output to next KF block after
30(can be ajusted) rounds of UKF to avoid jump in next KF output.
if count==300||count==0;
    lost(1)=0;
    flag=0;
    count=0;
else
    lost(1)=1;
end
if All_position_lost==1
   lost(1)=1;
end

lost2=All_position_lost;
```

```
 %heading recovered, do not inform KF immediately, but wait a few
minutes
lost3=All_heading_lost;
if lost3==0&&lost4==1
    flag1=1;
else

end

if flag1==1
   count1=count1+1;
end

if count1==300||count1==0;%heading is lost, then recovered. output to
next KF block after 30(can be ajusted) rounds of UKF to avoid jump in
next KF output.
    lost(2)=0;
    flag1=0;
    count1=0;
else
    lost(2)=1;
end
if All_heading_lost==1
   lost(2)=1;
end

lost4=All_heading_lost;

% % % % % % % % % % % % % %
SRUKF % % % % % % % % % % % % % % % % % % % %
wm=zeros(Lsigma*2+1,1);
wc=zeros(Lsigma*2+1,1);
X=zeros(Lsigma,Lsigma*2+1);
gamma_P=zeros(Lsigma,1);

q1=sPsub(1,1); q2=q1; q3=q1;
q4=sPsub(4,4); q5=q4; q6=q4;
q7=sPsub(7,7); q8=q7; q9=q7;
q10=sPsub(10,10); q11=q10; q12=q10;

Rv=zeros(Lsigma,Lsigma);
A=zeros(Lsigma,Lsigma);
F=zeros(Lsigma,Lsigma);
A(1,1)=1;A(1,4)=t;A(1,7)=t^2/2;
A(2,2)=1;A(2,5)=t;A(2,8)=t^2/2;
A(3,3)=1;A(3,6)=t;A(3,9)=t^2/2;

A(4,4)=1;A(4,7)=t;
A(5,5)=1;A(5,8)=t;
```

```
A(6,6)=1;A(6,9)=t;

A(7,7)=1;A(8,8)=1;A(9,9)=1;

A(10,10)=1;A(10,13)=t;A(10,16)=t^2/2;
A(11,11)=1;A(11,14)=t;A(11,17)=t^2/2;
A(12,12)=1;A(12,15)=t;A(12,18)=t^2/2;

A(13,13)=1;A(13,16)=t;
A(14,14)=1;A(14,17)=t;
A(15,15)=1;A(15,18)=t;
for i=16:24
A(i,i)=1;
end

F(1:3,4:6)=eye(3);
F(4:6,7:9)=eye(3);
F(10:12,13:15)=eye(3);
F(13:15,16:18)=eye(3);

Rv=[ (q1*t^5)/20,           0,           0, (q1*t^4)/8,           0,
0, (q1*t^3)/6,           0,           0,           0,           0,
0,           0,           0,           0,           0,           0,
0,    0,    0,    0,    0,    0,    0
           0, (q2*t^5)/20,           0,           0, (q2*t^4)/8,
0,           0, (q2*t^3)/6,           0,           0,           0,
0,           0,           0,           0,           0,           0,
0,    0,    0,    0,    0,    0,    0
           0,           0, (q3*t^5)/20,           0,           0,
(q3*t^4)/8,           0,           0, (q3*t^3)/6,           0,
0,           0,           0,           0,           0,           0,
0,           0,    0,    0,    0,    0,    0,    0
  (q1*t^4)/8,           0,           0, (q1*t^3)/3,           0,
0, (q1*t^2)/2,           0,           0,           0,           0,
0,           0,           0,           0,           0,           0,
0,    0,    0,    0,    0,    0,    0
           0,  (q2*t^4)/8,           0,           0, (q2*t^3)/3,
0,           0, (q2*t^2)/2,           0,           0,           0,
0,           0,           0,           0,           0,           0,
0,    0,    0,    0,    0,    0,    0
           0,           0,  (q3*t^4)/8,           0,           0,
(q3*t^3)/3,           0,           0, (q3*t^2)/2,           0,
0,           0,           0,           0,           0,           0,
0,           0,    0,    0,    0,    0,    0,    0
  (q1*t^3)/6,           0,           0, (q1*t^2)/2,           0,
0,       q1*t,           0,           0,           0,           0,
0,           0,           0,           0,           0,           0,
0,    0,    0,    0,    0,    0,    0
           0,  (q2*t^3)/6,           0,           0, (q2*t^2)/2,
0,           0,       q2*t,           0,           0,           0,
```

```
0,           0,           0,           0,           0,           0,
0,    0,     0,     0,     0,     0,     0
             0,           0,  (q3*t^3)/6,           0,           0,
(q3*t^2)/2,           0,           0,        q3*t,           0,
0,           0,           0,           0,           0,           0,
0,           0,    0,     0,     0,     0,     0,     0
            0,           0,           0,           0,           0,
0,           0,           0,           0, (q4*t^5)/20,           0,
0, (q4*t^4)/8,           0,           0, (q4*t^3)/6,           0,
0,    0,     0,     0,     0,     0,     0
            0,           0,           0,           0,           0,
0,           0,           0,           0,           0, (q5*t^5)/20,
0,           0, (q5*t^4)/8,           0,           0, (q5*t^3)/6,
0,    0,     0,     0,     0,     0,     0
            0,           0,           0,           0,           0,
0,           0,           0,           0,           0,           0,
(q6*t^5)/20,           0,           0, (q6*t^4)/8,           0,
0, (q6*t^3)/6,     0,     0,     0,     0,     0,     0
            0,           0,           0,           0,           0,
0,           0,           0,           0,  (q4*t^4)/8,           0,
0, (q4*t^3)/3,           0,           0, (q4*t^2)/2,           0,
0,    0,     0,     0,     0,     0,     0
            0,           0,           0,           0,           0,
0,           0,           0,           0,           0,  (q5*t^4)/8,
0,           0, (q5*t^3)/3,           0,           0, (q5*t^2)/2,
0,    0,     0,     0,     0,     0,     0
            0,           0,           0,           0,           0,
0,           0,           0,           0,           0,           0,
(q6*t^4)/8,           0,           0, (q6*t^3)/3,           0,           0,
(q6*t^2)/2,     0,     0,     0,     0,     0,     0
            0,           0,           0,           0,           0,
0,           0,           0,           0,  (q4*t^3)/6,           0,
0, (q4*t^2)/2,           0,           0,        q4*t,           0,
0,    0,     0,     0,     0,     0,     0
            0,           0,           0,           0,           0,
0,           0,           0,           0,           0,  (q5*t^3)/6,
0,           0, (q5*t^2)/2,           0,           0,        q5*t,
0,    0,     0,     0,     0,     0,     0
            0,           0,           0,           0,           0,
0,           0,           0,           0,           0,           0,
(q6*t^3)/6,           0,           0, (q6*t^2)/2,           0,           0,
q6*t,     0,     0,     0,     0,     0,     0
            0,           0,           0,           0,           0,
0,           0,           0,           0,           0,           0,
0,           0,           0,           0,           0,           0,
0, q7*t,     0,     0,     0,     0,     0
            0,           0,           0,           0,           0,
0,           0,           0,           0,           0,           0,
0,           0,           0,           0,           0,           0,
0,    0, q8*t,     0,     0,     0,     0
            0,           0,           0,           0,           0,
0,           0,           0,           0,           0,           0,
```

```
0,          0,          0,          0,          0,          0,
0,    0,    0, q9*t,     0,     0,     0
           0,          0,           0,          0,          0,
0,          0,          0,          0,          0,          0,
0,          0,          0,          0,          0,          0,
0,    0,    0,    0, q10*t,     0,     0
           0,          0,           0,          0,          0,
0,          0,          0,          0,          0,          0,
0,          0,          0,          0,          0,          0,
0,    0,    0,    0,     0, q11*t,     0
           0,          0,           0,          0,          0,
0,          0,          0,          0,          0,          0,
0,          0,          0,          0,          0,          0,
0,    0,    0,    0,     0,     0, q12*t
 ];

Rn=sPsub(nPsub1+1:nPsub,nPsub1+1:nPsub);

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
for i=0:Lsigma

    if i==0

        X(:,i+1)=Xe;
        wm(i+1)=lambada/(Lsigma+lambada);
        wc(i+1)=wm(i+1)+1-alfa^2+beta;

    else
        wm(i+1)=0.5/(Lsigma+lambada);
        wm(i+1+Lsigma)=wm(i+1);
        wc(i+1)=0.5/(Lsigma+lambada);
        wc(i+1+Lsigma)=wc(i+1);

        gamma_P(1:nXe,1)=P(1:nXe,i);;

        gamma_P=sqrt_gamma*gamma_P;
        X(:,i+1)=Xe(:,1)+gamma_P;
        X(:,i+1+Lsigma)=Xe(:,1)-gamma_P;
    end
end

Xkk1=zeros(nXe,Lsigma*2+1);

%time update
for i=1 :Lsigma*2+1
% runge kutta 4th order method
h=t;
   k1=F*X(:,i);
   k2=F*(X(:,i)+0.5*k1*h);
   k3=F*(X(:,i)+0.5*k2*h);
```

```
   k4=F*(X(:,i)+k3*h);
   Xkk1(:,i)=X(:,i)+h/6*(k1+2*k2+2*k3+k4);
end

X_k=wm(1)*Xkk1(:,1);
for i=2 : Lsigma*2+1
I1_=wm(i)*Xkk1(:,i);
X_k=X_k+I1_;
end

X_kex=zeros(nXe,2*Lsigma);
for i=2:2*Lsigma+1
       X_kex(:,i-1)=X_k;
end
[Sxxx,P_xk]=qr([sqrt(wc(2))*(Xkk1(:,2:end)-X_kex) sqrt(Rv)]',0);

P_xk=chol(P_xk'*P_xk+sign(wc(1))*(abs(wc(1)))*(Xkk1(:,1)-
X_k)*(Xkk1(:,1)-X_k)');

P_xk=P_xk';

Ykk1=zeros(nRn,Lsigma*2 + 1 );

for i=1 : Lsigma*2 + 1
invR=[                                cos(Xkk1(12,i))*cos(Xkk1(11,i)),
cos(Xkk1(11,i))*sin(Xkk1(12,i)),          -sin(Xkk1(11,i))
 cos(Xkk1(12,i))*sin(Xkk1(10,i))*sin(Xkk1(11,i)) -
cos(Xkk1(10,i))*sin(Xkk1(12,i)), cos(Xkk1(12,i))*cos(Xkk1(10,i)) +
sin(Xkk1(12,i))*sin(Xkk1(10,i))*sin(Xkk1(11,i)),
cos(Xkk1(11,i))*sin(Xkk1(10,i))
 sin(Xkk1(12,i))*sin(Xkk1(10,i)) +
cos(Xkk1(12,i))*cos(Xkk1(10,i))*sin(Xkk1(11,i)),
cos(Xkk1(10,i))*sin(Xkk1(12,i))*sin(Xkk1(11,i)) -
cos(Xkk1(12,i))*sin(Xkk1(10,i)), cos(Xkk1(10,i))*cos(Xkk1(11,i))];

R=[ cos(Xkk1(12,i))*cos(Xkk1(11,i)),
cos(Xkk1(12,i))*sin(Xkk1(10,i))*sin(Xkk1(11,i)) -
cos(Xkk1(10,i))*sin(Xkk1(12,i)), sin(Xkk1(12,i))*sin(Xkk1(10,i)) +
cos(Xkk1(12,i))*cos(Xkk1(10,i))*sin(Xkk1(11,i))
 cos(Xkk1(11,i))*sin(Xkk1(12,i)), cos(Xkk1(12,i))*cos(Xkk1(10,i)) +
sin(Xkk1(12,i))*sin(Xkk1(10,i))*sin(Xkk1(11,i)),
cos(Xkk1(10,i))*sin(Xkk1(12,i))*sin(Xkk1(11,i)) -
cos(Xkk1(12,i))*sin(Xkk1(10,i))
         -sin(Xkk1(11,i)),
cos(Xkk1(11,i))*sin(Xkk1(10,i)),
cos(Xkk1(10,i))*cos(Xkk1(11,i))];

dR=[0,
cos(Xkk1(10,i))*(Xkk1(13,i))*Xkk1(11,i)+sin(Xkk1(10,i))*(Xkk1(14,i))+s
in(Xkk1(10,i))*(Xkk1(13,i))*Xkk1(12,i)-cos(Xkk1(10,i))*(Xkk1(15,i)),
```

```
(Xkk1(15,i))*sin(Xkk1(10,i))+Xkk1(12,i)*cos(Xkk1(10,i))*(Xkk1(13,i))-
sin(Xkk1(10,i))*(Xkk1(13,i))*Xkk1(11,i)+cos(Xkk1(10,i))*(Xkk1(14,i))
Xkk1(15,i), -
sin(Xkk1(10,i))*(Xkk1(13,i))+(Xkk1(15,i))*sin(Xkk1(10,i))*Xkk1(11,i)+X
kk1(12,i)*cos(Xkk1(10,i))*(Xkk1(13,i))*Xkk1(11,i)+Xkk1(12,i)*sin(Xkk1(
10,i))*(Xkk1(14,i)), -
sin(Xkk1(10,i))*(Xkk1(13,i))*Xkk1(12,i)*Xkk1(11,i)+cos(Xkk1(10,i))*(Xk
k1(15,i))*Xkk1(11,i)+cos(Xkk1(10,i))*Xkk1(12,i)*(Xkk1(14,i))-
cos(Xkk1(10,i))*(Xkk1(13,i))
-(Xkk1(14,i)), cos(Xkk1(10,i))*(Xkk1(13,i)), -
sin(Xkk1(10,i))*(Xkk1(13,i))];
ddR=[0, -
sin(Xkk1(10,i))*(Xkk1(13,i))^2*Xkk1(11,i)+cos(Xkk1(10,i))*(Xkk1(16,i))
*Xkk1(11,i)+2*cos(Xkk1(10,i))*(Xkk1(13,i))*(Xkk1(14,i))+sin(Xkk1(10,i)
)*(Xkk1(17,i))+cos(Xkk1(10,i))*(Xkk1(13,i))^2*Xkk1(12,i)+sin(Xkk1(10,i
))*(Xkk1(16,i))*Xkk1(12,i)+2*sin(Xkk1(10,i))*(Xkk1(13,i))*(Xkk1(15,i))
-cos(Xkk1(10,i))*(Xkk1(18,i)),
(Xkk1(18,i))*sin(Xkk1(10,i))+2*(Xkk1(15,i))*cos(Xkk1(10,i))*(Xkk1(13,i
))-
Xkk1(12,i)*sin(Xkk1(10,i))*(Xkk1(13,i))^2+Xkk1(12,i)*cos(Xkk1(10,i))*(
Xkk1(16,i))-cos(Xkk1(10,i))*(Xkk1(13,i))^2*Xkk1(11,i)-
sin(Xkk1(10,i))*(Xkk1(16,i))*Xkk1(11,i)-
2*sin(Xkk1(10,i))*(Xkk1(13,i))*(Xkk1(14,i))+cos(Xkk1(10,i))*(Xkk1(17,i
))
    Xkk1(18,i), -cos(Xkk1(10,i))*(Xkk1(13,i))^2-
sin(Xkk1(10,i))*(Xkk1(16,i))+(Xkk1(18,i))*sin(Xkk1(10,i))*Xkk1(11,i)+2
*(Xkk1(15,i))*cos(Xkk1(10,i))*(Xkk1(13,i))*Xkk1(11,i)+2*(Xkk1(15,i))*s
in(Xkk1(10,i))*(Xkk1(14,i))-
Xkk1(12,i)*sin(Xkk1(10,i))*(Xkk1(13,i))^2*Xkk1(11,i)+Xkk1(12,i)*cos(Xk
k1(10,i))*(Xkk1(16,i))*Xkk1(11,i)+2*Xkk1(12,i)*cos(Xkk1(10,i))*(Xkk1(1
3,i))*(Xkk1(14,i))+Xkk1(12,i)*sin(Xkk1(10,i))*(Xkk1(17,i)), -
cos(Xkk1(10,i))*(Xkk1(13,i))^2*Xkk1(11,i)*Xkk1(12,i)-
sin(Xkk1(10,i))*(Xkk1(16,i))*Xkk1(11,i)*Xkk1(12,i)-
2*sin(Xkk1(10,i))*(Xkk1(13,i))*Xkk1(11,i)*(Xkk1(15,i))-
2*sin(Xkk1(10,i))*(Xkk1(13,i))*(Xkk1(14,i))*Xkk1(12,i)+cos(Xkk1(10,i))
*Xkk1(11,i)*(Xkk1(18,i))+2*cos(Xkk1(10,i))*(Xkk1(14,i))*(Xkk1(15,i))+c
os(Xkk1(10,i))*(Xkk1(17,i))*Xkk1(12,i)+sin(Xkk1(10,i))*(Xkk1(13,i))^2-
cos(Xkk1(10,i))*(Xkk1(16,i))
    -(Xkk1(17,i)), -
sin(Xkk1(10,i))*(Xkk1(13,i))^2+cos(Xkk1(10,i))*(Xkk1(16,i)), -
cos(Xkk1(10,i))*(Xkk1(13,i))^2-sin(Xkk1(10,i))*(Xkk1(16,i))];

if nGPS>=1
for j=1:nGPS
Ykk1(1+6*(j-1):3+6*(j-1),i)=(Xkk1(1:3,i)+R*[GPS(j).Pgps_1.x-
s.P0_1.x;GPS(j).Pgps_1.y-s.P0_1.y;GPS(j).Pgps_1.z-s.P0_1.z]);

Ykk1(4+6*(j-1):6+6*(j-1),i)=(Xkk1(4:6,i)+dR*[GPS(j).Pgps_1.x-
s.P0_1.x;GPS(j).Pgps_1.y-s.P0_1.y;GPS(j).Pgps_1.z-s.P0_1.z]);
end
end
```

```
if nIMU>=1
for j=1:nIMU
Ykk1(1+nGPS*6+9*(j-1):3+nGPS*6+9*(j-1),i)=(-
invR*g+invR*Xkk1(7:9,i)+invR*ddR*[IMU(j).Pmru_1.x-
s.P0_1.x;IMU(j).Pmru_1.y-s.P0_1.y;IMU(j).Pmru_1.z-
s.P0_1.z])+Xkk1(19:21,i)+IMUbias_in(j).ab;

Ykk1(4+nGPS*6+9*(j-1):6+nGPS*6+9*(j-1),i)=Xkk1(10:12,i);

Ykk1(7+nGPS*6+9*(j-1):9+nGPS*6+9*(j-
1),i)=Xkk1(13:15,i)+Xkk1(22:24,i)+IMUbias_in(j).wb;
end
end

if nRadar>=1
for j=1:nRadar
Ykk1(1+nGPS*6+nIMU*9+3*(j-1):3+nGPS*6+nIMU*9+3*(j-
1),i)=(Xkk1(1:3,i)+R*[Radar(j).Pgps_1.x-s.P0_1.x;Radar(j).Pgps_1.y-
s.P0_1.y;Radar(j).Pgps_1.z-s.P0_1.z]);

end
end

end


Y_k=wm(1)*Ykk1(:,1);
for i=2 : Lsigma*2+1

I1_=wm(i)*Ykk1(:,i);
Y_k=Y_k+I1_;

end

Y_kex=zeros(nRn,2*Lsigma);
for i=2:2*Lsigma+1
       Y_kex(:,i-1)=Y_k;
end
[Syyy,P_yk]=qr([sqrt(wc(2))*(Ykk1(:,2:end)-Y_kex)  sqrt(Rn)]',0);
[P_yk,p]=chol(P_yk'*P_yk+wc(1)*(Ykk1(:,1)-Y_k)*(Ykk1(:,1)-Y_k)');

P_yk=P_yk';
XI=Xkk1(:,1)-X_k;
YI=Ykk1(:,1)-Y_k;;
Pxy=wc(1)*XI*YI';

for i=1+1 : Lsigma*2+1

I1_=Xkk1(:,i)-X_k;
```

```
XI=I1_;

I1_=Ykk1(:,i)-Y_k;

YI=I1_;

I1=wc(i)*XI*YI';
Pxy=Pxy+I1;
end

KK=(Pxy/P_yk')/P_yk;

Xe=X_k+KK*f*(Ym'-Y_k);

U=KK*P_yk;
[Ptest,p]=chol(P_xk*P_xk'-U*U','lower');
if p==0 %in case P is not positive definite
    P=chol(P_xk*P_xk'-U*U','lower');
else
   P=P2;
end

P2=P;


IMUbias=IMU;
invR=[ cos(Xe(12,1))*cos(Xe(11,1)),
cos(Xe(12,1))*sin(Xe(10,1))*sin(Xe(11,1)) -
cos(Xe(10,1))*sin(Xe(12,1)), sin(Xe(12,1))*sin(Xe(10,1)) +
cos(Xe(12,1))*cos(Xe(10,1))*sin(Xe(11,1))
 cos(Xe(11,1))*sin(Xe(12,1)), cos(Xe(12,1))*cos(Xe(10,1)) +
sin(Xe(12,1))*sin(Xe(10,1))*sin(Xe(11,1)),
cos(Xe(10,1))*sin(Xe(12,1))*sin(Xe(11,1))  -  cos(Xe(12,1))*sin(Xe(10,1))
         -sin(Xe(11,1)),
cos(Xe(11,1))*sin(Xe(10,1)),
cos(Xe(10,1))*cos(Xe(11,1))]';
 if nIMU>=1
for i=1:nIMU
   IMUbias(i).ab=Ym(1+nGPS*6+9*(i-1):3+nGPS*6+9*(i-1))'+invR*g-
(invR*Xe(7:9,1)+invR*ddR*[IMU(i).Pmru_1.x-s.P0_1.x;IMU(i).Pmru_1.y-
s.P0_1.y;IMU(i).Pmru_1.z-s.P0_1.z]);
   IMUbias(i).wb=Ym(7+nGPS*6+9*(i-1):9+nGPS*6+9*(i-1))'-Xe(13:15,1);
end
 end
```

9.2.3 KF Algorithm for Observer

```
% % % % % % % % % % % % % KF algorithm% % % % % % % % % % % % %
function y  = fcn(s,t,w,tau,lost,cwb,cwxi,cR,alpha)

persistent P;
persistent Xe;
persistent Xe1;
persistent flag;
persistent wb;

if isempty(flag)
    flag=1;
end;
n=15;
if isempty(Xe)
    Xe=zeros(n,1);
end;
if isempty(Xe1)
    Xe1=zeros(n,1);
end;
if isempty(P)
    P=eye(n,n);
end;

if flag==1
    Xe=eye(n,1)
    P=eye(n,n);
    flag=0;
    z=zeros(12,1);
end
z=zeros(12,1);
z(1:9)=s;
z(10:12)=Xe(4:6,1);
R=zeros(12,12);
R(1,1)=cR(1);
R(2,2)=cR(2);
R(3,3)=cR(3);
R(4,4)=cR(4);
R(5,5)=cR(5);
R(6,6)=cR(6);
R(7,7)=cR(7);
R(8,8)=cR(8);
R(9,9)=cR(9);

R(10,10)=1e8;
R(11,11)=1e8;
R(12,12)=1e2;
if lost(1) ==0

else
```

```
    R(1,1)=cR(1)*1e4;
    R(2,2)=cR(2)*1e4;
    R(4,4)=cR(4)*1e4;
    R(5,5)=cR(5)*1e4;
    z(1:2)=zeros(2,1);
    z(4:5)=zeros(2,1);
end

if lost(2) ==0

else
    R(3,3)=cR(3)*1e4;

    R(6,6)=cR(6)*1e4;

    z(3)=0;%
    z(6)=0;
end

M=[     7.010149032153999e+06                          0            0
    0     8.519007042379107e+06      4.718726399134355e+05
    0    -2.595508500000000e+06      3.797290756932775e+09] ; %correct
value that is found in vessel.mat and vessleABC.mat

D=[2.648609825197792e+05 0 0
    0 8.816423000000000e+05 -1e7
    0 -1e7 337743760]; %correct value that is found in vessel.mat and
vessleABC.mat in Bv

omega=diag(w);
zeta=diag([0.01,0.01,0.01])*1;%damping

wb=diag(cwb);
wxi=diag(cwxi);
T=diag([600,600,600]);
phi=Xe(3);
Rz=[cos(phi) -sin(phi) 0
    sin(phi) cos(phi) 0
    0 0 1];
F22=-(M)\D;
F23=(M)\Rz';
F33=diag([0,0,0]);
F54=-omega.^2;
F55=-2*zeta*omega;
B21=inv(M);

F=[zeros(3) Rz zeros(3) zeros(3) zeros(3)
    zeros(3) F22 F23 zeros(3) zeros(3)
    zeros(3) zeros(3) F33 zeros(3) zeros(3)
    zeros(3) zeros(3) zeros(3) zeros(3) eye(3)
```

```
    zeros(3) zeros(3) zeros(3) F54 F55];%process transition matrix for
continious time

A=[ eye(3),     (F22*Rz*t^2)/2 + Rz*t,
(F23*Rz*t^2)/2,                       zeros(3),
zeros(3)
    zeros(3), (F22^2*t^2)/2 + F22*t + eye(3), F23*t + (F22*F23*t^2)/2
+ (F23*F33*t^2)/2,                          zeros(3),
zeros(3)
    zeros(3),                           zeros(3),
(F33^2*t^2)/2 + F33*t + eye(3),                         zeros(3),
zeros(3)
    zeros(3),                           zeros(3),
zeros(3),         (F54*t^2)/2 + eye(3),
(F55*t^2)/2 + eye(3)*t
    zeros(3),                           zeros(3),
zeros(3), (F54*F55*t^2)/2 + F54*t, (F55^2*t^2)/2 + F55*t + (F54*t^2)/2
+ eye(3)
    ];%process transition matrix

Q=[                              (F23^2*Rz^2*t^5*wb)/20,
(F23^2*Rz*t^4*wb*(2*F22*t + 2*F33*t + 5*eye(3)))/40,
(F23*Rz*t^3*wb*(6*F33^2*t^2 + 15*F33*t + 20*eye(3)))/120,
zeros(3),
zeros(3)
    (F23^2*Rz*t^4*wb*(2*F22*t + 2*F33*t + 5*eye(3)))/40,
(F23^2*t^3*wb*(3*F22^2*t^2 + 6*F22*F33*t^2 + 15*F22*t + 3*F33^2*t^2 +
15*F33*t + 20*eye(3)))/60, ((F23*wb*F33^3)/20 +
(F22*F23*wb*F33^2)/20)*t^5 + ((F23*wb*F33^2)/4 +
(F22*F23*wb*F33)/8)*t^4 + ((F22*F23*wb)/6 + (F23*F33*wb)/2)*t^3 +
(F23*wb*t^2)/2,
zeros(3),
zeros(3)
    (F23*Rz*t^3*wb*(6*F33^2*t^2 + 15*F33*t + 20*eye(3)))/120,
((F23*wb*F33^3)/20 + (F22*F23*wb*F33^2)/20)*t^5 + ((F23*wb*F33^2)/4 +
(F22*F23*wb*F33)/8)*t^4 + ((F22*F23*wb)/6 + (F23*F33*wb)/2)*t^3 +
(F23*wb*t^2)/2,
(wb*F33^4*t^5)/20 + (wb*F33^3*t^4)/4 + (2*wb*F33^2*t^3)/3 + wb*F33*t^2
+ wb*t,
zeros(3),
zeros(3)
    zeros(3),
zeros(3),
zeros(3),
(t^3*wxi*(3*F55^2*t^2 + 15*F55*t + 20*eye(3)))/60,
((wxi*F55^3)/20 + (F54*wxi*F55)/20)*t^5 + ((wxi*F55^2)/4 +
(F54*wxi)/8)*t^4 + (F55*wxi*t^3)/2 + (wxi*t^2)/2
    zeros(3),
zeros(3),
zeros(3), ((wxi*F55^3)/20 + (F54*wxi*F55)/20)*t^5 + ((wxi*F55^2)/4 +
(F54*wxi)/8)*t^4 + (F55*wxi*t^3)/2 + (wxi*t^2)/2, ((wxi*F54^2)/20 +
(wxi*F54*F55^2)/10 + (wxi*F55^4)/20)*t^5 + ((wxi*F55^3)/4 +
```

```
(F54*wxi*F55)/4)*t^4 + ((2*wxi*F55^2)/3 + (F54*wxi)/3)*t^3 +
F55*wxi*t^2 + wxi*t
    ];%process noise covariance matrix


B0=[zeros(3);B21;zeros(3);zeros(3);zeros(3)];

u=tau;

H=[eye(3) zeros(3) zeros(3) eye(3) zeros(3);
    zeros(3) Rz zeros(3) zeros(3) eye(3) ;
    zeros(3) Rz*F22 Rz*F23 Rz*Rz'*F54 Rz*Rz'*F55
    zeros(3) eye(3) zeros(3) zeros(3) zeros(3) ;
    ];


% % % % % % % % % KF

P = alpha^2*A * P * A' + Q;
K=P*H'/(H*P*H'+R);

% % % % % % % % % % runge kutta 4th order% % % % % % % % % %
h=t;
k1=F*Xe + B0*u;
k2=F*(Xe+0.5*k1*h)+B0*u;
k3=F*(Xe+0.5*k2*h)+B0*u;
k4=F*(Xe+k3*h)+B0*u;
Xe=Xe+h/6*(k1+2*k2+2*k3+k4);

f=eye(12);
if lost(1)~=0
    f(1:2,1:2)=f(1:2,1:2)*0;
    f(4:5,4:5)=f(4:5,4:5)*0;

end

if lost(2)~=0
    f(3,3)=f(3,3)*0;
    f(6,6)=f(6,6)*0;
    f(7:9,7:9)=f(7:9,7:9)*0;
end
% Correction based on observation:
Xe = Xe +  K*f*(z-H*Xe-[zeros(6,1); Rz*B21*u;zeros(3,1)]);
P = P -  K*H*P;
P=0.5*(P+P');
y=Xe;
```