NTNU – Trondheim
Norwegian University of
Science and Technology

Localization and Tracking of Ships and Objects Using a UAV Mounted Thermal Imaging Camera.

Kenan Trnka

Master of Science in Cybernetics and Robotics
Submission date: Januar 2015
Supervisor: Thor Inge Fossen, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Faculty of Information Technology,
Mathematics and Electrical Engineering
Department of Engineering Cybernetics

MSC THESIS DESCRIPTION SHEET

Name:	Kenan Trnka
Department:	Engineering Cybernetics
Thesis title (Norwegian):	Lokalisering og følging av skip og objekter ved hjelp av infrarødt kamera montert på UAV
Thesis title (English):	Localization and Tracking of Ships and Objects using a UAV mounted Thermal Imaging Camera

Thesis Description: The purpose of the thesis is to design, implement and compare different observers capable of precisely localizing objects (icebergs). This includes finding their position and speed based on infrared video and GPS measurements from an UAV. The observers should be implemented and tested on the UAV's onboard computer, as well as interfaced with the UAV's infrared video feed. The task will deal with challenges such as observer design, sensor fusion and simulation on real world data.

The following items should be considered:

1. Define the scope of the thesis and clarify what your contributions are.
2. Design and integration of the X8 payload system for data logging.
3. Synchronization of GPS measurements and video stream
4. Overview of real-time detection of objects.
5. Discuss possibilities for implementation of a human-in-the-loop system such that it is possible to monitor and administrate the list of detected objects on the ground.
6. Develop a position and velocity observer in North-East-Down coordinates based on GPS/IMU measurements and image-based object positions.
7. Experimental testing and generation of data using the X8 fixed-wing UAV for data collection and synchronization of GPS and video stream.
8. Conclude your results

Start date:	2014-09-01
Due date:	2015-01-25

Thesis performed at:	Department of Engineering Cybernetics, NTNU
Supervisor:	Professor Thor I. Fossen, Dept. of Eng. Cybernetics, NTNU
Co-Supervisor:	Frederik S. Leira, Dept. of Eng. Cybernetics, NTNU

Summary

This thesis presents the creation of a payload system for a small unmanned aircraft, which is used for data collection and testing of detection and tracking theory. Theory for position estimation of on-ground or on-sea objects from a camera from height is presented, along with methods for reducing error associated with camera lens distortion errors, and synchronization errors between GPS/INS and camera measurements. An overview of methods for detecting and classifying objects is presented, along with a method for determining a ship's orientation in the water, allowing for direction (yaw) feedback to a nonlinear observer. It is shown that such a measurement greatly increases the accuracy of the observer. Theory for observing and tracking objects with known and unknown models is presented, using both linear and nonlinear Kalman filters. The observers and other theory were tested in simulations, and will be tested with real-world data after flight tests. The results are concluded in a paper submitted to ICUAS'15 by Leira and myself.

Sammendrag

Denne oppgaven presenterer konstruksjon av en nyttelast for et lite ubemannet fly, som skal brukes til datainnsamling og testing av deteksjons- og sporingsteori. Teori for estimering av posisjonen til objekter på bakken eller på sjøen fra et kamera i høyde er presentert, sammen med metoder for minimering av feil assosiert med kameralinseeffekter, og tidssynkroniseringsfeil mellom GPS/INS og kameramålinger. En oversikt over metoder for deteksjon og klassifisering av objekter er presentert, sammen med en metode for å bestemme et skips orientering i vannet, som gir mulighet for tilbakekobling av skipets retning inn i en ulineær tilstandsestimator. Det er vist at en slik måling i stor grad øker nøyaktigheten til tilstandsestimator. Teori for tilstandsestimering og sporing av objekter med kjente og ukjente modeller er presentert, ved bruk av både linære og ulineære Kalmanfiltre. Observerne og annen teori ble testet i simuleringer, og skal testes med reelle data samlet inn fra flygetester. Resultatene av dette er konkludert i en artikkel innsendt til ICUAS'15 i samarbeid med Leira.

Acknowledgements

I would like to thank my supervisor Thor I. Fossen for helpful feedback along the way, and for helping me shape this thesis into something I could be proud of. I would also like to thank my co-supervisor Frederik S. Leira for his help with the computer vision aspect of the thesis, and his co-operation during the construction of the payload. I would also like to thank Jostein Furseth, Espen Skjong and Stian Nundal for discussing and reading through this thesis. Finally, I would like to thank my parents, and my loving girlfriend Silje, who has supported me throughout, with both dinners, clean clothes and proof-reading.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

1.1 Background and motivation

The detection and tracking of objects in water, such as icebergs, has been the focus of many studies. Tracking icebergs helps avoid collisions with ships and submersible vehicles. Tracking other objects, such as ships, small boats, people and debris has many benefits. This is usually done with the aid of ships, larger manned aircraft, or satellite images. Small unmanned aircraft can be better suited for such tasks, which are easily deployable, low cost and most importantly, able to fail without directly effecting loss of life, or large scale damage to objects in its vicinity. The sea environment is ideal for detecting objects by thermal imaging photogrammetry, as the sea surface in many cases will have a different thermal signature than objects floating in it. The sea environment will therefore be the environment of focus in this thesis. The main goals are to simplify the procedures of acquiring data, and detecting and tracking objects, and automatically dealing with issues such as synchronization errors.

1.2 Literature survey

The topic of tracking objects from height using a camera has been researched before. Xiang and Tian [3] have developed a method for automatic georeferencing of images from UAVs, where georeferencing accuracy was tested, for the purpose of image mosaicing[1]. Hemerly [15] presents a method for automatically determining lens correction coefficients and interior camera parameters for use in automatic georeferencing in UAVs. The same method is implemented in OpenCV[2], which will be discussed later in Chapter 6.6. Methods for tracking known and uknown systems using linear and nonlinear observers, Kalman observers etc. are discussed by Fossen [1], Kang, Krener, Xiao and Xu [18] and Leira [13]. Classification of ships and boats has been discussed by Teutch and Krüger [19] with good results, and a classifier written by Leira [13] gives a 93.3% accurate classification of objects of interest at sea. Many topics have been discussed separately, so our goal is to combine theories for detection, tracking and classification into a single system. Little information is available on video and GPS/INS synchronization, therefore methods will have to be researched.

1.3 Contribution

A relatively low-cost, simple framework for thermal object detection and tracking, built upon a low-cost airframe will be presented. The primary goal of this thesis is to create a fully working system for detecting and tracking objects on the sea surface, using a lightweight UAV. The payload for the X8 discussed later in Chapter 6, has been designed with systems and sensors necessary to verify the object detection and classification algorithms written by Leira [13], and the

[1] Mosaicing is the process of stiching multiple overlapping snapshot images of an image or document together into one continuous composite.

[2] Open Source Computer Vision Library.

georeferencing mathematics, observer models and video synchronization algorithm written in this thesis. Methods for compensating for different errors in an on-ground object position estimate are presented and analyzed. Synchronization of on-board measurements is discussed, and a method for determining synchronization errors is developed. Object tracking using both linear and nonlinear model-based Kalman filters is implemented and tested.

The X8, along with its payload, will be used as a testbed for a paper which is submitted to ICUAS'15 by Leira and myself [14], titled

- *A Light Weight Camera Based Payload with Georeferencing Capabilities for Small Fixed Wing UAVs.*

The paper is based largely on work done by Leira for his doctoral thesis, including the video synchronization and observer theory developed in this thesis.[3]

[3]International Conference on Unmanned Aircraft Systems.

1.4 Organization

This thesis is organized into nine chapters, including this introductory chapter.

- Chapter 2 - Theory of reference frames, camera theory, georeferencing and error sources are discussed.
- Chapter 3 - An overview of theory for detecting objects in thermal images using computer vision is presented.
- Chapter 4 - Introduces ship and aircraft models used later in simulations, along with linear and nonlinear state observers.
- Chapter 5 - The observers and camera geo-referencing are tested in simulation.
- Chapter 6 - A test platform created for this thesis and Leira's doctoral thesis is presented.
- Chapter 7 - A summary of flight tests of the X8 is presented.
- Chapter 8 - Discussion.
- Chapter 9 - Conclusion and discussion of improvements which could be done in the future.

Chapter 2

Theory

2.1 Reference frames

To be able to estimate the position of an object seen through a camera, we need to clarify the use of different reference frames. The position of an object on the Earth can be done by expressing its position in the Earth-centered Earth Fixed (ECEF) frame. The position is given by [1, p. 19]:

$$\boldsymbol{p}_{b/e}^{e} = \begin{bmatrix} X_e \\ Y_e \\ Z_e \end{bmatrix} \tag{2.1}$$

The x-axis points to 0° latitude and 0° longitude, and deviations from this is given by [1, p. 19]:

$$\boldsymbol{\Theta}_{en} = \begin{bmatrix} l \\ \mu \end{bmatrix} \tag{2.2}$$

where l and μ are latitude and longitude. We will use the North East Down (NED) frame to describe the positions of detected objects on the ground. Transforming from ECEF to NED is given by the

rotation below, dependent on the latitude and longitude [1, p. 34]:

$$\boldsymbol{R}_e^n(\boldsymbol{\Theta}_{en}) = \begin{bmatrix} -\sin(\mu)\cos(l) & -\sin(l) & -\cos(\mu)\cos(l) \\ -\sin(\mu)\sin(l) & \cos(l) & -\cos(\mu)\sin(l) \\ \cos(\mu) & 0 & -\sin(\mu) \end{bmatrix} \tag{2.3}$$

The position in NED coordinates is given by [1, p. 19]:

$$\boldsymbol{p}_{b/n}^n = \begin{bmatrix} N \\ E \\ D \end{bmatrix} \tag{2.4}$$

The aircraft itself has a frame which is fixed to it, known as the BODY frame. The rotation between the NED and BODY frame is shown in Figure 2.2, and is expressed by the attitude, which is given by [1, p. 19]:

$$\boldsymbol{\Theta}_{nb} = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \tag{2.5}$$

where ϕ, θ and ψ are euler angles roll, pitch and yaw, respectively. Figure 2.1 illustrates these rotations.

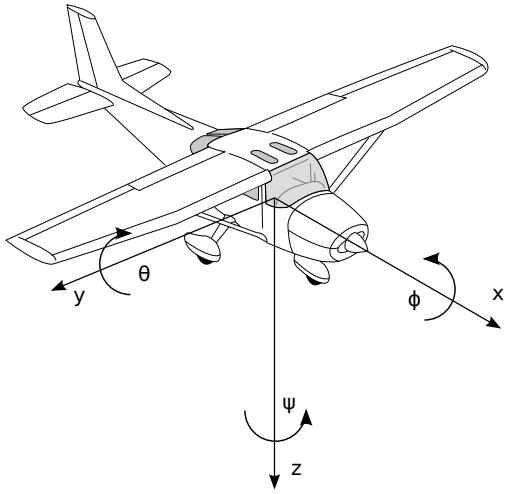


Figure 2.1: Body attitude.

The rotation between BODY and NED is given by [1, p. 22]:

$$\boldsymbol{R}_b^n(\boldsymbol{\Theta}_{nb}) = \boldsymbol{R}_{z,\psi}\boldsymbol{R}_{y,\theta}\boldsymbol{R}_{x,\phi} \tag{2.6}$$

$$= \begin{bmatrix} \mathrm{c}\psi \cdot \mathrm{c}\phi & -\mathrm{s}\psi \cdot \mathrm{c}\phi + \mathrm{c}\psi \cdot \mathrm{s}\theta \cdot \mathrm{s}\phi & \mathrm{s}\psi \cdot \mathrm{s}\phi + \mathrm{c}\psi \cdot \mathrm{c}\phi \cdot \mathrm{s}\theta \\ \mathrm{s}\psi \cdot \mathrm{c}\theta & \mathrm{c}\psi \cdot \mathrm{c}\phi + \mathrm{s}\phi \cdot \mathrm{s}\theta \cdot \mathrm{s}\psi & -\mathrm{c}\psi \cdot \mathrm{s}\phi + \mathrm{s}\theta \cdot \mathrm{s}\psi \cdot \mathrm{c}\phi \\ -\mathrm{s}\theta & \mathrm{c}\theta \cdot \mathrm{s}\phi & \mathrm{c}\theta \cdot \mathrm{c}\phi \end{bmatrix} \tag{2.7}$$

Since rotation matrices are orthogonal, the inverse rotation from NED to BODY follows as the transpose of Equation (2.6):

$$\boldsymbol{R}_n^b(\boldsymbol{\Theta}_{nb}) = (\boldsymbol{R}_b^n)^{-1} = (\boldsymbol{R}_b^n)^\top \tag{2.8}$$

$$= \begin{bmatrix} \mathrm{c}\psi \cdot \mathrm{c}\phi & \mathrm{s}\psi \cdot \mathrm{c}\theta & -\mathrm{s}\theta \\ -\mathrm{s}\psi \cdot \mathrm{c}\phi + \mathrm{c}\psi \cdot \mathrm{s}\theta \cdot \mathrm{s}\phi & \mathrm{c}\psi \cdot \mathrm{c}\phi + \mathrm{s}\phi \cdot \mathrm{s}\theta \cdot \mathrm{s}\psi & \mathrm{c}\theta \cdot \mathrm{s}\phi \\ \mathrm{s}\psi \cdot \mathrm{s}\phi + \mathrm{c}\psi \cdot \mathrm{c}\phi \cdot \mathrm{s}\theta & -\mathrm{c}\psi \cdot \mathrm{s}\phi + \mathrm{s}\theta \cdot \mathrm{s}\psi \cdot \mathrm{c}\phi & \mathrm{c}\theta \cdot \mathrm{c}\phi \end{bmatrix} \tag{2.9}$$

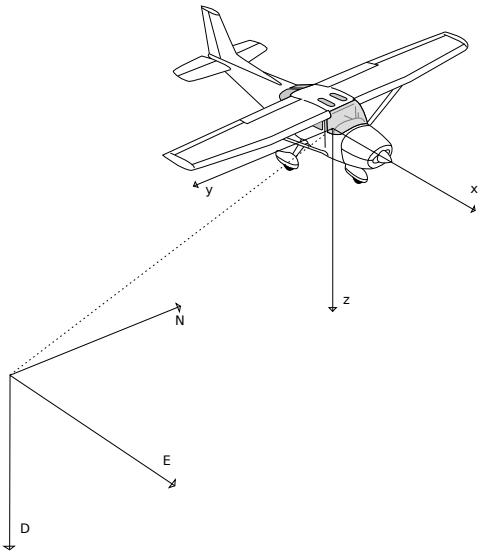


Figure 2.2: Aircraft body frame in relation to NED frame

Simplified equations of motion for an aircraft, necessary for simulation, will be derived from the equations of motion of a ship later in Chapter 4.1.5.

2.2 Camera reference frame

In the case of the aircraft used for the test flight portion of this thesis, the camera is mounted in the belly of the aircraft, and is oriented so that the z-axis points down through the camera. However, it has a fixed rotation of 90° degrees about the aircraft's z-axis, in addition to the attitude of the gimbal. This first fixed rotation will be called the mounting offset, which is given by:

$$\boldsymbol{\Theta}_{mount} = \begin{bmatrix} \phi_m \\ \theta_m \\ \psi_m \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \frac{\pi}{2} \end{bmatrix} \tag{2.10}$$

In addition, the attitude of the gimbal is given by:

$$\boldsymbol{\Theta}_{cam} = \begin{bmatrix} 0 \\ \theta_c \\ \psi_c \end{bmatrix} \tag{2.11}$$

As we can see, the gimbal is only actuated in two degrees. Transforming between BODY and CAM is then given by these two rotations:

$$\boldsymbol{R}_b^c = \boldsymbol{R}_m^c(\boldsymbol{\Theta}_{cam})\boldsymbol{R}_b^m(\boldsymbol{\Theta}_{mount}) \tag{2.12}$$

$$= \boldsymbol{R}_{z,\psi}\boldsymbol{R}_{y,\theta}\boldsymbol{R}_{z,\frac{\pi}{2}} \tag{2.13}$$

$$= \begin{bmatrix} \cos(\psi_c)\cos(\theta_c) & -\sin(\psi_c) & \cos(\psi_c)\sin(\theta_c) \\ \cos(\theta_c)\sin(\psi_c) & \cos(\psi_c) & \sin(\psi_c)\sin(\theta_c) \\ -\sin(\theta_c) & 0 & \cos(\theta_c) \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.14}$$

$$= \begin{bmatrix} -\sin(\psi_c) & -\cos(\psi_c)\cos(\theta_c) & \cos(\psi_c)\sin(\theta_c) \\ \cos(\psi_c) & -\cos(\theta_c)\sin(\psi_c) & \sin(\psi_c)\sin(\theta_c) \\ 0 & \sin(\theta_c) & \cos(\theta_c) \end{bmatrix} \tag{2.15}$$

The camera frame is show in Figure 2.3. We then find the rotation from NED to CAM frame:

$$\boldsymbol{R}_n^c = \boldsymbol{R}_b^c \boldsymbol{R}_n^b \tag{2.16}$$

$$= \begin{bmatrix} -\mathrm{s}\psi_c & -\mathrm{c}\psi_c \cdot \mathrm{c}\theta_c & \mathrm{c}\psi_c \cdot \mathrm{s}\theta_c \\ \mathrm{c}\psi_c & -\mathrm{c}\theta_c \cdot \mathrm{s}\psi_c & \mathrm{s}\psi_c \cdot \mathrm{s}\theta_c \\ 0 & \mathrm{s}\theta_c & \mathrm{c}\theta_c \end{bmatrix} \tag{2.17}$$

$$\cdot \begin{bmatrix} \mathrm{c}\psi \cdot \mathrm{c}\phi & -\mathrm{s}\psi \cdot \mathrm{c}\phi + \mathrm{c}\psi \cdot \mathrm{s}\theta \cdot \mathrm{s}\phi & \mathrm{s}\psi \cdot \mathrm{s}\phi + \mathrm{c}\psi \cdot \mathrm{c}\phi \cdot \mathrm{s}\theta \\ \mathrm{s}\psi \cdot \mathrm{c}\theta & \mathrm{c}\psi \cdot \mathrm{c}\phi + \mathrm{s}\phi \cdot \mathrm{s}\theta \cdot \mathrm{s}\psi & -\mathrm{c}\psi \cdot \mathrm{s}\phi + \mathrm{s}\theta \cdot \mathrm{s}\psi \cdot \mathrm{c}\phi \\ -\mathrm{s}\theta & \mathrm{c}\theta \cdot \mathrm{s}\phi & \mathrm{c}\theta \cdot \mathrm{c}\phi \end{bmatrix} \tag{2.18}$$

$$= \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{2.19}$$

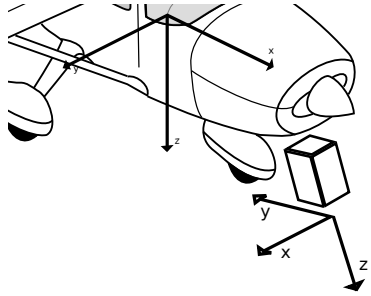


Figure 2.3: Rotated camera frame.

This relation will be used in the next chapter on georeferencing.

2.3 Georeferencing

We are interested in determining the position of objects seen by the thermal imaging camera. In order to do so, we need to transform the pixel coordinates of the objects into coordinates in the NED or ECEF frames. Figure 2.4 shows the relation between the camera and NED frames. We assume that the input from the object detection algorithm will be a specific point in the image plane which represents the image coordinates (placement of this point will be determined by the detection algorithm, usually near the assumed center of mass of the object). By denoting a point in the image $\boldsymbol{p} = [p_u, p_v]^\top$, we can find the equivalent mapping point $\boldsymbol{P} = [X_m, Y_m, Z_m]^\top$ with the following equation [3]:

$$s \begin{bmatrix} \boldsymbol{p} \\ 1 \end{bmatrix} = \boldsymbol{A} \begin{bmatrix} \boldsymbol{R}_n^c & \boldsymbol{R}_n^c \boldsymbol{C} \end{bmatrix} \begin{bmatrix} \boldsymbol{P} \\ 1 \end{bmatrix} \tag{2.20}$$

where s is an arbitrary scaling factor, used to keep the resulting vector homogeneous. Further, we have the translation $\boldsymbol{C} = [C_x, C_y, C_z]^\top$, which corresponds to the position of the camera in the NED frame. The 3x3 $\boldsymbol{A}$ matrix contains the camera intrinsic parameters:

$$\boldsymbol{A} = \begin{bmatrix} f_{px,u} & 0 & u_0 \\ 0 & f_{px,v} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.21}$$

where u_0, v_0 are the camera's principal point, which is the center point of the image projected onto the camera and $f_{px,u}, f_{px,v}$ are the camera pixel focal lengths. The center point does not necessarily coincide with the center of the camera coordinate system, however, a fair assumption is to set $u_0 = 0.5k_u$ and $v_0 = 0.5k_v$.

For the FLIR Tau2 thermal camera, we have the following basic table of parameters:

Table 2.1: Camera parameters for the FLIR Tau2 thermal camera

Pixel size	
k_u	704 (upscaled)
k_v	480 (upscaled)
Principal point	
u_0	352 (upscaled)
v_0	240 (upscaled)
Focal length	
f	9 [mm]
Pixel pitch	17 [μm]

The camera has an actual resolution of 336 by 256 pixels, and is the resulting image is scaled up to 704 by 480 pixels. This means that when calculating the focal length in pixels, we have to recalculate the pixel pitch to match the upscaled image. First we can calculate the sensor size:

$$w_s = 17 \cdot 10^{-6} \cdot 336 = 5.712 \text{ [mm]} \tag{2.22}$$

$$h_s = 17 \cdot 10^{-6} \cdot 256 = 4.352 \text{ [mm]} \tag{2.23}$$

Where w_s and h_s are sensor width and height, respectively. From this we can calculate the new pixel pitch for the upscaled image:

$$pitch_u = \frac{w_s}{k_u} = 8.11 \cdot 10^{-3} \text{ [}\mu\text{m]} \tag{2.24}$$

$$pitch_v = \frac{h_s}{k_v} = 9.07 \cdot 10^{-3} \text{ [}\mu\text{m]} \tag{2.25}$$

The focal lengths are then given as:

$$f_{px,u} = \frac{f}{pitch_u} \approx 1118\text{[px]} \tag{2.26}$$

$$f_{px,v} = \frac{f}{pitch_v} \approx 1000\text{[px]} \tag{2.27}$$

We can now calculate our intrinsic camera matrix $\boldsymbol{A}$:

$$\boldsymbol{A} = \begin{bmatrix} 1118 & 0 & 352 \\ 0 & 1000 & 240 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.28}$$

We will make the assumption that all objects lie in the $Z = 0$ plane, which allows us to reduce Equation (2.20) to:

$$s \begin{bmatrix} p_u \\ p_v \\ 1 \end{bmatrix} = \boldsymbol{A} \begin{bmatrix} \boldsymbol{c}_1 & \boldsymbol{c}_2 & \boldsymbol{R}_n^c \boldsymbol{C} \end{bmatrix} \begin{bmatrix} X_m \\ Y_m \\ 1 \end{bmatrix} \tag{2.29}$$

where $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ denote the first and second columns of $\boldsymbol{R}_n^c$. The camera exterior orientation parameters has full rank and is invertible, and we are interested in obtaining the ground position from the image coordinates, so we can find an expression for $\boldsymbol{P}$ by inverting the matrix:

$$\underbrace{s \begin{bmatrix} X_m \\ Y_m \\ 1 \end{bmatrix}}_{\boldsymbol{P}_h} = \begin{bmatrix} \boldsymbol{c}_1 & \boldsymbol{c}_2 & \boldsymbol{R}_n^c \boldsymbol{C} \end{bmatrix}^{-1} \boldsymbol{A}^{-1} \begin{bmatrix} p_u \\ p_v \\ 1 \end{bmatrix} \tag{2.30}$$

where $\boldsymbol{P}_h = \begin{bmatrix} P^\top & 1 \end{bmatrix}^\top$ is the homogeneous version of $\boldsymbol{P}$. In addition, if we have detailed height maps of a particular area, the $Z = 0$ assumption can be disregarded.

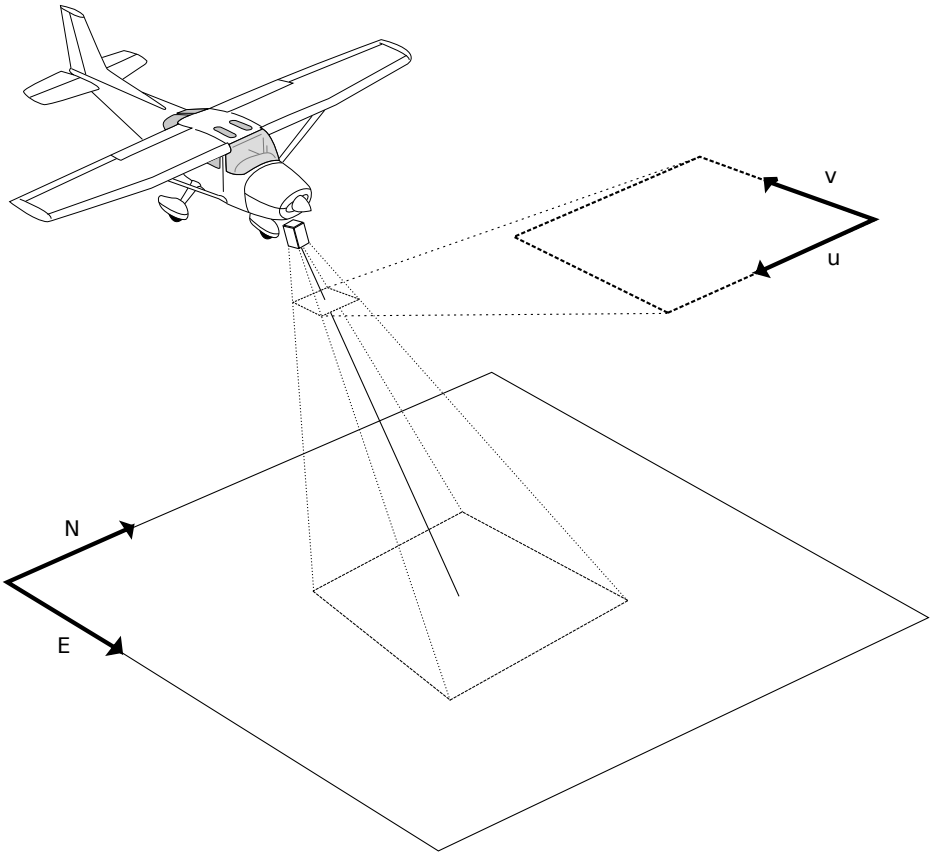


Figure 2.4: Camera plane shown projected on the ground.

2.4 Error sources and disturbances

To obtain an accurate estimate of an object's position, knowledge and removal of known error sources is needed. We will look into the main error and disturbance sources here.

2.4.1 Data synchronization

Synchronizing the timestamps of the aircraft and camera position estimation is important, as the estimated on-ground position will move if the two sources are not synchronized. If we assume that the aircraft is traveling with velocity $\boldsymbol{v} = \boldsymbol{v}_0$, its position will be changing by $\boldsymbol{p}_a = \boldsymbol{v}t$. In this example, we assume that the position of the camera is the same as the aircraft, $\boldsymbol{p}_c = \boldsymbol{p}_a$. If we assume that the position estimate of the aircraft is received at $t = 0$, the delayed image will arrive at $t = \Delta t$. As before, $\boldsymbol{p}_{h,\Delta t}$ are the homogenous pixel coordinates, received at $t = \Delta t$. The difference in position will then be:

$$\tilde{\boldsymbol{p}}_c = \boldsymbol{p}_{c,\Delta t} - \boldsymbol{p}_{c,0} = \boldsymbol{v}\Delta t \tag{2.31}$$

We can find an expression for the error in estimated position by inserting into Equation (2.30):

$$\tilde{\boldsymbol{P}}_h = \boldsymbol{P}_{h,\Delta t} - \boldsymbol{P}_{h,0} \tag{2.32}$$

$$\begin{aligned}\tilde{\boldsymbol{P}}_h =& \left(\begin{bmatrix} \boldsymbol{c}_1 & \boldsymbol{c}_2 & \boldsymbol{R}_n^c \boldsymbol{p}_{c,\Delta t} \end{bmatrix}^{-1} \boldsymbol{A}^{-1} \boldsymbol{p}_{h,\Delta t} \right) \\ & - \left(\begin{bmatrix} \boldsymbol{c}_1 & \boldsymbol{c}_2 & \boldsymbol{R}_n^c \boldsymbol{p}_{c,0} \end{bmatrix}^{-1} \boldsymbol{A}^{-1} \boldsymbol{p}_{h,\Delta t} \right) \end{aligned} \tag{2.33}$$

$$= \left(\begin{bmatrix} \boldsymbol{c}_1 & \boldsymbol{c}_2 & \boldsymbol{R}_n^c \boldsymbol{p}_{c,\Delta t} \end{bmatrix}^{-1} - \begin{bmatrix} \boldsymbol{c}_1 & \boldsymbol{c}_2 & \boldsymbol{R}_n^c \boldsymbol{p}_{c,0} \end{bmatrix}^{-1} \right) \boldsymbol{A}^{-1} \boldsymbol{p}_{h,\Delta t} \tag{2.34}$$

The elements $\boldsymbol{c}_1$, $\boldsymbol{c}_2$ and $\boldsymbol{R}_n^c$ are also dependent on the time (Δt or 0), but subscripts have been omitted to de-clutter the equations.
We can test the error for a specific scenario, in which the aircraft is travelling due north, at 400 ft above ground level (AGL), with a

ground speed of 20 m/s. The camera parameters are chosen so as to fit the FLIR Tau2 camera, and the object is assumed to be seen in the middle of the image. We can see in Table 2.2 how a time delay affects positional accuracy.

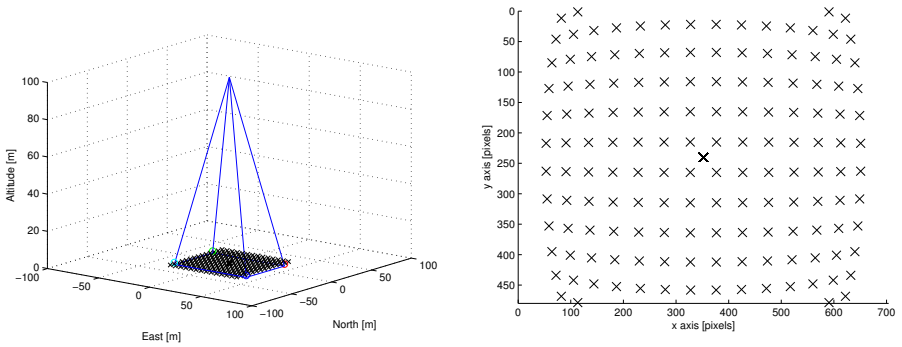
Table 2.2: Camera image time delay positioning error.

Time delay [ms]	Position error north [m]
0	0
10	-0.2
100	-2
1000	-20

In this case, since there is no rotation, Equation (2.32) reduces to (2.31), and we get a linear relationship between the time delay of the image, and the estimated position error. It shows that determining the unknown delay Δt is important for proper position estimation.

2.4.2 Lens distortion

Lens distortion is an effect which is caused by parallel light traveling through a curved lens, hitting a flat image sensor. Because of this, the perceived image will have *radial distortion*, which must be corrected to get the best estimate of ground position. Radial distortion is shown in Figure 2.5. Typically, lens distortion is more prominent near the edges of the image.



(a) Visualisation of aircraft above square grid of objects on the ground.

(b) View of grid through camera, showing radial distortion.

Figure 2.5: Visualization of radial distortion. The distortion is exaggerated to clearly show its effect.

Most models try to fit a high order polynomial equation for the radial distortion, which is given by:

$$r_d = r + \delta_r \tag{2.35}$$

where r_d is the distorted radial distance, and δ_r is the radial distortion. Further, the polynomial used to model the distortion is on the form:

$$r_d = rf(r) = r(1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + \cdots) \tag{2.36}$$

where $k_1, k_2, k_3, ...$ are distortion coefficients, and $r = \sqrt{u_d^2 + v_d^2}$. This model is presented in [4], before presenting a method which is claimed

to be superior. However, we will use the simplified model

$$\begin{aligned} u_c &= u_d(1 + k_1 r^2 + k_2 r^4 + k_5 r^6) \\ v_c &= v_d(1 + k_1 r^2 + k_2 r^4 + k_5 r^6) \end{aligned} \tag{2.37}$$

for the radial effects, reducing the number of coefficients, and allowing us to directly use the OpenCV calibration algorithm. This same method is used in [3]. Here, u_c, v_c are the corrected (undistorted) normalized positions, u_d, v_d are the distorted, normalized positions, k_1, k_2, k_5 are distortion coefficients. The distorted, normalized positions can be found as [15]:

$$s \begin{bmatrix} u_d \\ v_d \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{c}_1 & \boldsymbol{c}_2 & \boldsymbol{R}_n^c \boldsymbol{C} \end{bmatrix} \begin{bmatrix} X_m \\ Y_m \\ 1 \end{bmatrix} \tag{2.38}$$

From this we can find the undistorted frame coordinates:

$$s \begin{bmatrix} p_u \\ p_v \\ 1 \end{bmatrix} = \boldsymbol{A} \begin{bmatrix} u_c + d_1 \\ v_c + d_2 \\ 1 \end{bmatrix} \tag{2.39}$$

Where again, s is an arbitrary scalar used to keep the vectors homogeneous. The tangential errors d_1, d_2 are the offset of the principal point of the camera, or *principal point shift*. It is given by:

$$\begin{aligned} d_1 &= 2k_3 \cdot u_d \cdot v_d + k_4(r^2 + 2u_d^2) \\ d_2 &= k_3(r^2 + 2v_d^2) + 2k_4 \cdot u_d \cdot v_d \end{aligned} \tag{2.40}$$

Determining the coefficients is described in Chapter 6.6.

We are however interested in determining the position of an object on the ground from a distorted image. We therefore need the inverse

operation:

$$s \begin{bmatrix} u_c + d_1 \\ v_c + d_2 \\ 1 \end{bmatrix} = \boldsymbol{A}^{-1} \begin{bmatrix} p_u \\ p_v \\ 1 \end{bmatrix} \tag{2.41}$$

$$s \begin{bmatrix} u_c \\ v_c \\ 1 \end{bmatrix} = \boldsymbol{A}^{-1} \begin{bmatrix} p_u \\ p_v \\ 1 \end{bmatrix} - \begin{bmatrix} d_1 \\ d_2 \\ 0 \end{bmatrix} \tag{2.42}$$

Finally, since we have that

$$u_d = \frac{u_c}{1 + k_1 r^2 + k_2 r^4 + k_5 r^6} \tag{2.43}$$

$$v_d = \frac{u_c}{1 + k_1 r^2 + k_2 r^4 + k_5 r^6} \tag{2.44}$$

we find the ground position:

$$s \begin{bmatrix} X_m \\ Y_m \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{c}_1 & \boldsymbol{c}_2 & \boldsymbol{R}_n^c \boldsymbol{C} \end{bmatrix}^{-1} \begin{bmatrix} u_d \\ v_d \\ 1 \end{bmatrix} \tag{2.45}$$

2.5 Video/GPS synchronization using Optical Flow

Optical flow methods have been a subject of research for a long time [16][17]. The optical flow between two images, or two frames of a video, describes the motion of objects (pixels) from one image to the next, and has been used extensively for video compression, motion detection schemes and object segmentation. For our purposes it is useful to determine the *synchronization error* between Global Positioning System (GPS) and Inertial Navigation Systen (INS) measurements, and camera measurements.

2.5.1 Correlation between flow and GPS/INS measurements

Methods, such as the Lucas Kanade (LK) method [16] can give the linear transformation from one frame to the next, which allows us to calculate the rotational motion of the camera. As the camera is firmly attached to the aircraft, its rotational motion must correspond to the rotational motion of the aircraft (unless the camera gimbal is actuated, however such rotations are known and can be filtered out). Processing the thermal image data takes a certain amount of time, and this causes a delay between data from the video and the data from the GPS/INS-measurements. This delay is denoted Δt. As is mention in Chapter 2.4.1, a synchronization error leads to position estimation errors on the ground, as shown in Table 2.2. A method for determining Δt from INS rotation measurements compared with Optical Flow rotation estimates is presented in the next section.

2.5.2 Estimation method

Figure 2.6 shows a generated example of similar measurements taken from different sources, with a time delay of 20 ms introduced. Additionally the amplitude of the measurements from the other source

is smaller, to make the example more realistic with regards to errors from flow rotation estimation.

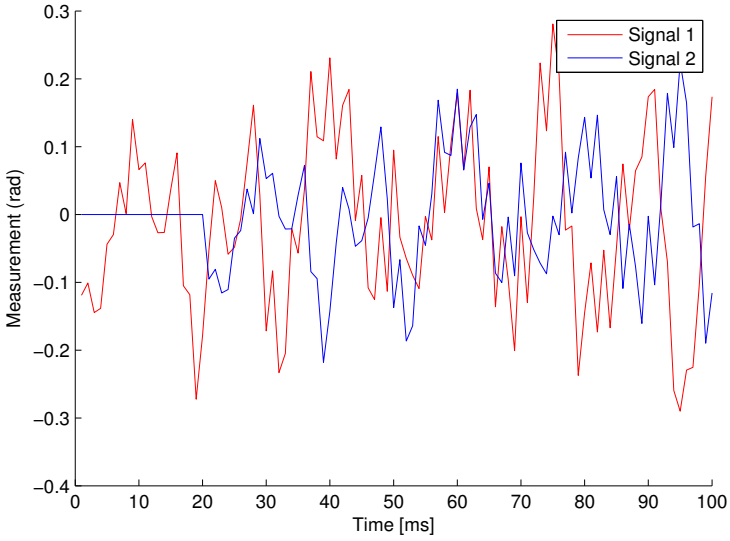


Figure 2.6: Similar data, captured by two sources at different times.

The discrete signals are defined as $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$, of size $(1 \times n)$. The first step, in order to reduce the possibility of estimating Δt wrong due to bias errors in the signals, is to filter the signals with the one-dimensional Prewitt (1×3) operator, given as:

$$\boldsymbol{P} := \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \tag{2.46}$$

The Prewitt allows us to find an approximation of the signals first derivative. Filtering the signal is done by discrete *convolution*, which is defined as:

$$(f * g)(n) = \sum_{m=-M}^{M} f(n-m)g(m) \tag{2.47}$$

Substituting for our case, we get that:

$$d\boldsymbol{s}(n) \approx \sum_{m=-1}^{1} \boldsymbol{s}(n-m)\boldsymbol{P}(m) \tag{2.48}$$

The differentiated signals are shown in Figure 2.7.

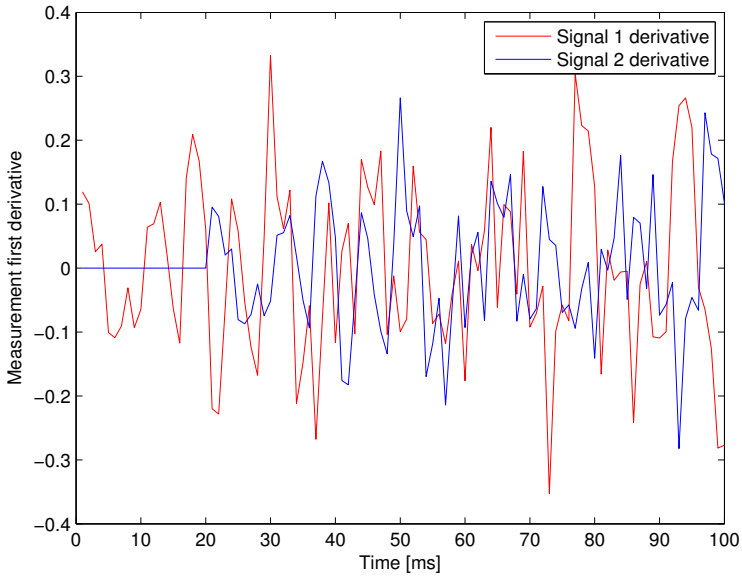


Figure 2.7: Signals after Prewitt differential operator.

Now we find the Δt which gives the best match between the signals. The error $\boldsymbol{q}(\Delta t)$ is given as:

$$\boldsymbol{q}(\Delta t) = \sum_{t} [d\boldsymbol{s}_1(t) - d\boldsymbol{s}_2(t + \Delta t)]^2 \tag{2.49}$$

We are interested in finding the Δt which minimizes the error function:

$$\Delta t_{min} = \arg\min_{\Delta t} \boldsymbol{q}(\Delta t) \tag{2.50}$$

The error is shown in Figure 2.8 for different Δt, and clearly shows a large dip around 20 milliseconds, which is the delay we purposefully introduced.

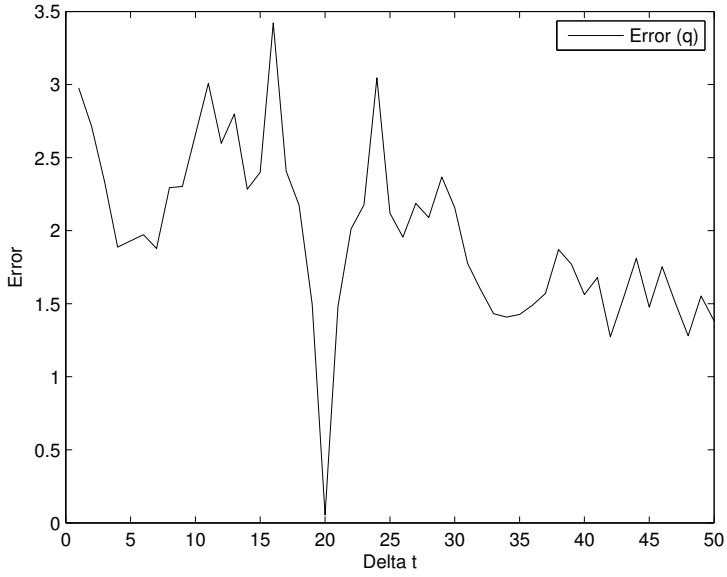


Figure 2.8: Errors for different values of Δt.

The synchronized signals are shown in Figure 2.9.

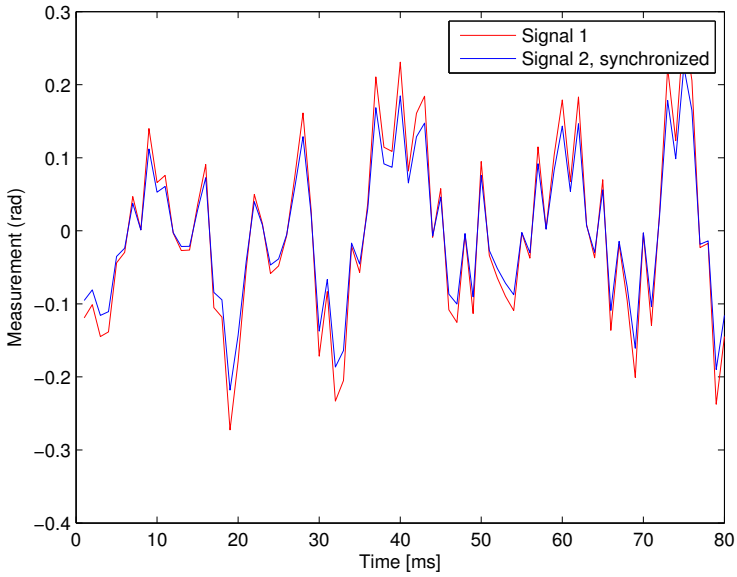


Figure 2.9: Signals after successful synchronization.

2.6 Summary

In this chapter, reference frame theory has been presented, along with methods for geo-referencing objects on the ground from a UAV mounted camera. Errors sources, such as lens distortion and data synchronization are presented, along with methods for handling these issues. A simple method for synchronization of camera data with GPS/INS data was presented.

Chapter 3

Detection and classification

The following chapter will give a quick overview of object detection techniques within the field of Computer Vision (CV), which were used by Leira [13] to detect objects floating in water, ships, people and other. Additionally, a method for determining yaw angle of an elongated object is presented. As it is necessary to be able to distinguish objects from one another, and from frame to frame, a quick overview of object classification will be given. All images in this chapter, with the exception of Figure 3.4, are courtesy of Leira.

An overview of data flow in the system detection and classification system developed by Leira is shown in Figure 3.1. For the payload created for the test platform discussed in Chapter 6, the *Object Detection* and *Classification* modules are provided by Leira, while the *Object Tracking* module is provided by this thesis, based on theory developed in Chapter 4.

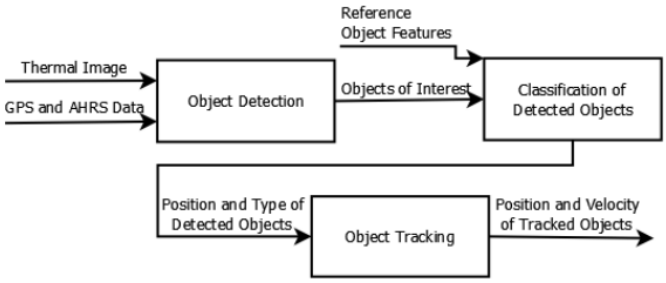


Figure 3.1: Data flow diagram showing detection, classification and tracking system.

3.1 Segmentation

Image segmentation is very important when doing analysis of image data. The main goal is to divide the image into parts which have a strong correlation with objects presented in the image. In our case, we are interested in detecting and tracking ships and icebergs in ocean environments. An example picture acquired from the thermal camera is shown in Figure 3.2. In the figure, the objects of interest are the ship and two small objects.

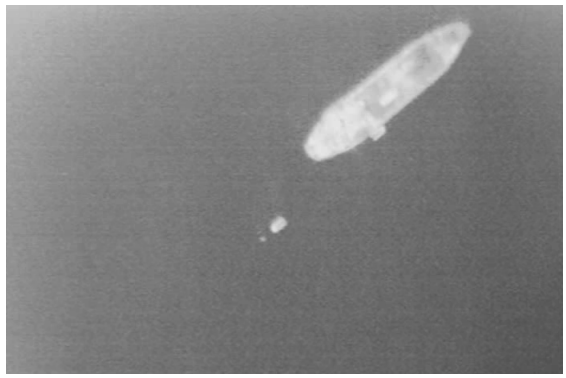


Figure 3.2: Thermal image taken from UAV of a ship and other objects.

3.1.1 Image smoothing

Image smoothing methods are methods for suppressing image noise, by using redundancies in the image data. [9, p. 124] The new values are based on mixing the brightness values in a neighborhood $\mathcal{O}$ of the pixel. Local image smoothing provides noise suppression, and can also remove other degradations which could prevent a good result after further processing. The image is smoothed by convolving the image with a Gaussian kernel $\boldsymbol{g}$, where $\boldsymbol{g}$ is an $n \times n$ matrix, also known as a convolution kernel, approximating Gaussian distribution. An example of a 3x3 Gaussian kernel is given as:

$$\boldsymbol{g} = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{3.1}$$

The convolution operator for the image is defined as [13, p. 3]:

$$\boldsymbol{I}_s(x,y) = (\boldsymbol{I} * \boldsymbol{g})(x,y) \tag{3.2}$$

$$= \sum_{k=0.5(n-1)}^{m=0.5(n-1)} \sum_{m=0.5(n-1)} \boldsymbol{I}(x-m, y-k)\boldsymbol{g}(m,k) \tag{3.3}$$

Where $\boldsymbol{I}$ is a matrix containing the image which will be processed, and $\boldsymbol{I}_s$ is the smoothed image matrix. The effects of this operation can be seen in Figure 3.3.

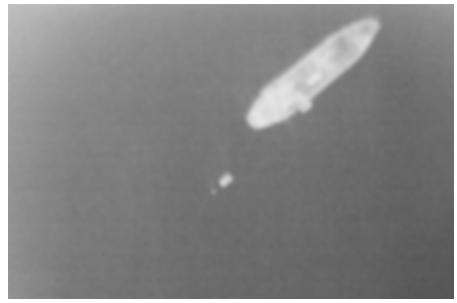


Figure 3.3: Figure 3.2 after smoothing.

3.1.2 Thresholding

Different methods of thresholding exist. The simplest is gray-level thresholding, which segments objects from the background by light intensity. A threshold value, T, represents a light intensity, above which the image is classified as part of the object, and below as part of the background. The algorithm is described as [9, p. 176]:

1. Search all the pixels $f(i, j)$ of the image f. An image element $g(i, j)$ of the segmented image is an object pixel if $f(i, j) \geq T$, and is a background pixel otherwise.

The transformation is given as:

$$g(i,j) = \begin{cases} 1, & \text{for} f(i,j) \geq T \\ 0, & \text{for} f(i,j) < T. \end{cases} \tag{3.4}$$

Methods of optimally determining the threshold value exist. Figure 3.4 shows thresholding with three different values for T. As we can see, this helps separate the objects from the background, but also shows that the system is not robust; other methods must be developed to properly separate objects from the background.

(a) Low T value. (b) Medium T value. (c) High T value.

Figure 3.4: Sample of binary thresholded images.

3.1.3 Thresholded image after Prewitt operator

The Prewitt gradient operator determines an approximation to the first derivative of the image. It is given as [9, p. 136]:

$$\boldsymbol{P}_x := \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}, \boldsymbol{P}_y := \boldsymbol{P}_x^\top \tag{3.5}$$

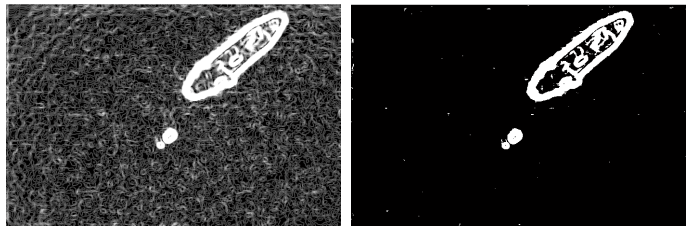
The horizontal and vertical Prewitt images are given as:

$$\boldsymbol{I}_{P,x} = \boldsymbol{I} * \boldsymbol{P}_x \tag{3.6}$$

$$\boldsymbol{I}_{P,y} = \boldsymbol{I} * \boldsymbol{P}_y \tag{3.7}$$

To find horizontal and vertical edges, the two images $\boldsymbol{I}_{P,x}$ and $\boldsymbol{I}_{P,y}$ can be combined into the Prewitt image:

$$\boldsymbol{I}_p(x,y) = \sqrt{\boldsymbol{I}_{P,x}(x,y)^2 + \boldsymbol{I}_{P,y}(x,y)^2} \tag{3.8}$$



(a) Figure 3.2 after Prewitt operator.

(b) Thresholding after Prewitt operator.

Figure 3.5: Image with thresholding of Prewitt image.

As we can see, the thresholding works much better after processing the image with the Prewitt operator, however, there are small disturbances in the image which may give false positives. According to Leira [13], these can be removed by removing items of unexpectedly small size [13, p. 3], resulting in the image shown in Figure 3.6.

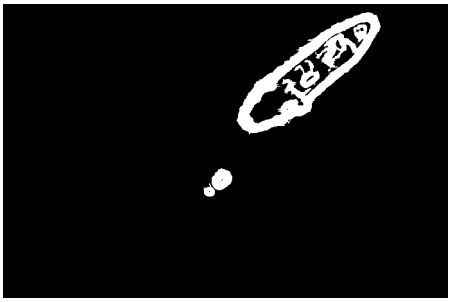
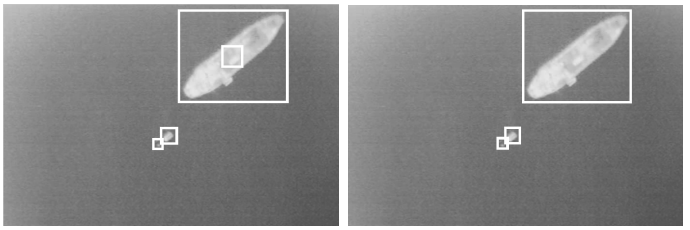


Figure 3.6: The image is free of blobs, with only objects of interest remaining.

3.1.4 Bounding of detected objects

It is now possible to surround detected continuously connected components with bounding boxes. Bounding boxes wholly contained within other boxes are disregarded [13, p. 4]. The result of this is shown in Figure 3.7.



(a) Bounding boxes.

(b) Bounding boxes with subboxes removed.

Figure 3.7: Image with objects detected.

3.2 Yaw estimation

In most cases, a ship or a boat is longer in the direction that they travel, than their width. This is evident in Figure 3.2, where we can see that the ship is either headed down to the left, or up to the right. The angles, relative to the image frame, are shown in Figure 3.8. As we will see in Chapter 4.4, the observability of our nonlinear observer attains full rank when the ship's yaw is measured directly.

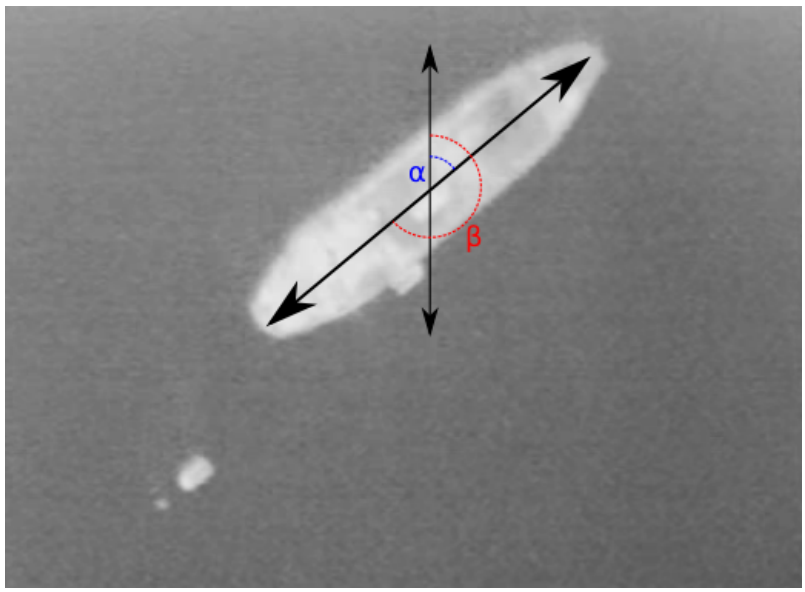


Figure 3.8: The image shows the two angles α an β, which are the two possible ship yaw estimates.

A ship object will appear as an elongated blob. The property of *elongatedness* is defined as the ratio between the length and the width of the region bounding rectangle [9, p. 355]. It can be determined by eroding[1] the object until it disappears. If the number of erosion

[1]Erosion is a fundamental property of morphological image processing, which, for a binary image, removes the outer "layer" of pixels, progressively making the object smaller.

cycles is denoted d, the elongatedness e is given as [9]

$$e = \frac{A}{(2d)^2} \tag{3.9}$$

where A is the area of the object. When an objects elongatedness is determined, it can be determined if the object is likely to be a ship. In such a case, the direction (yaw) of the ship can be calculated.

3.2.1 Image region moments

Image moments are a useful way to describe objects after segmentation, and in our case, helps us determine a ship's yaw direction.

For a scalar image $\boldsymbol{I}(x, y)$, we have that the moment of dimension $(p \times q)$ is given as [9, p. 357]:

$$m_{pq} = \sum_{x=-\infty}^{\infty} \sum_{y=-\infty}^{\infty} x^p y^q \boldsymbol{I}(x, y) \tag{3.10}$$

where x and y are the pixel coordinates of the image. The central moments of the region are defined as:

$$\mu_{pq} = \sum_{x=-\infty}^{\infty} \sum_{y=-\infty}^{\infty} (x - x_c)^p (y - y_c)^q \boldsymbol{I}(x, y) \tag{3.11}$$

where $x_c = \frac{m_{10}}{m_{00}}$ and $y_c = \frac{m_{01}}{m_{00}}$ are the centroid components. The central moments are useful for calculating the direction of a region, which is shown in the next section.

3.2.2 Direction of the ship

Direction is defined as the direction of the longer side of an elongated regions minimum bounding rectangle. When the shape moments are known, the direction θ can be calculated as:

$$\theta = \frac{1}{2} \arctan \frac{2\mu_{11}}{\mu_{20} - \mu_{02}} \tag{3.12}$$

This direction corresponds to the angles α or β shown in Figure 3.8. Which direction is correct can later be determined from the ship's estimated motion.

3.3 Object classification

Classification of objects detected in the thermal image stream is important, as it is necessary to distinguish objects from one another in order to be able to track each one individually. Leira [13] presents a method for classifying objects using moments and central moments. The classifier is trained using images containing known objects, which later allows it to classify new unknown objects, and distinguish them from one another. The classifier created by Leira, which will be used in test flights of our combined system, shows that it classifies 93.3% of detected objects of interest correctly.

3.4 Summary

In this chapter, an overview of theory used for detecting and classifying objects on the ocean surface has been presented, along with a method for determining the ship's yaw angle from image moments.

Chapter 4

Observers

Determining the current track, velocity and position for unidentified objects in the water depends very much on what kind of object is being tracked. A small boat will be a lot more maneuverable than a large ship, and an observer which does not take into account these differences will not work well in either case. Determining the model best suited for each case is a challenge which will not be the focus of this thesis, as its workload warrants a thesis of its own. However, two basic models will be looked into, a simple positional model tracking icebergs and other floating objects, and another for ship tracking. These models will be used as a foundation for a Kalman filter.

4.1 Ship model

4.1.1 Translational dynamics

An object floating on the surface of water can be modelled by a simple translational motion model, based upon the Newton-Euler equations of motion [1, p. 48]

$$m[\dot{\boldsymbol{\nu}}^b_{b/n} + \boldsymbol{S}(\boldsymbol{\omega}^b_{b/n})\boldsymbol{\nu}^b_{b/n}] = \boldsymbol{f}^b_b \tag{4.1}$$

where

$$\boldsymbol{\nu}_{b/n}^{b} = \begin{bmatrix} u & v & w \end{bmatrix}^{\top} \tag{4.2}$$

is linear velocity of origin relative $\{n\}$, expressed in $\{b\}$,

$$\boldsymbol{\omega}_{b/n}^{b} = \begin{bmatrix} p & q & r \end{bmatrix}^{\top} \tag{4.3}$$

is angular velocity of $\{b\}$ relative to $\{n\}$, expressed in $\{b\}$. Finally, the forces acting on the system are given by

$$\boldsymbol{f}_{b}^{b} = \begin{bmatrix} X & Y & Z \end{bmatrix}^{\top} \tag{4.4}$$

acting on the origin expressed in $\{b\}$, m is the mass of the object, and $\boldsymbol{S}$ is the cross product operator, defined as [1, p. 20]:

$$\boldsymbol{S}(\boldsymbol{\omega}_{b/n}^{b}) = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix}, \qquad \boldsymbol{S} = -\boldsymbol{S}^{\top} \tag{4.5}$$

4.1.2 Rotational dynamics

The rotational dynamics are given as [1, p. 48]:

$$\boldsymbol{I}_g \dot{\boldsymbol{\omega}}_{b/n}^{b} - \boldsymbol{S}(\boldsymbol{I}_g \boldsymbol{\omega}_{b/n}^{b}) \boldsymbol{\omega}_{b/n}^{b} = \boldsymbol{m}_g \tag{4.6}$$

where $\boldsymbol{I}_g$ is the inertia matrix about the center of gravity, defined as

$$\boldsymbol{I}_g = \begin{bmatrix} I_x & -I_{xy} & -I_{xz} \\ -I_{yx} & I_y & -I_{yz} \\ -I_{zx} & -I_{zy} & I_z \end{bmatrix}, \qquad \boldsymbol{I_g} = \boldsymbol{I_g}^{\top} > 0 \tag{4.7}$$

and $\boldsymbol{m}_g$ is a vector of moments. A fair assumption for the inertia matrix is that is has homogenous weight distribution, and xz-plane symmetry [1, p. 134], which gives

$$I_{xy} = I_{xz} = 0 \tag{4.8}$$

4.1.3 Newton-Euler equations of motion about CG

The total equations of motion in matrix form is given as [1, p. 49]:

$$\begin{bmatrix} m\boldsymbol{I}_{3x3} & \boldsymbol{0}_{3x3} \\ \boldsymbol{0}_{3x3} & \boldsymbol{I}_g \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{\nu}}^b_{g/n} \\ \dot{\boldsymbol{\omega}}^b_{b/n} \end{bmatrix} + \begin{bmatrix} m\boldsymbol{S}(\boldsymbol{\omega}^b_{b/n}) & \boldsymbol{0}_{3x3} \\ \boldsymbol{0}_{3x3} & -\boldsymbol{S}(\boldsymbol{I}_g\boldsymbol{\omega}^b_{b/n}) \end{bmatrix} \begin{bmatrix} \boldsymbol{\nu}^b_{g/n} \\ \boldsymbol{\omega}^b_{b/n} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}^b_g \\ \boldsymbol{m}^b_g \end{bmatrix} \tag{4.9}$$

The position of the object in NED coordinates is given as

$$\dot{\boldsymbol{p}}^n_{b/n} = \boldsymbol{R}^n_b(\boldsymbol{\Theta}_{nb})\boldsymbol{\nu}^n_{b/n} \tag{4.10}$$

where $\boldsymbol{\Theta}_{nb} = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^\top$ are the objects generalized angles, related with the angular velocities below:

$$\dot{\boldsymbol{\Theta}}_{nb} = \boldsymbol{T}_\Theta(\boldsymbol{\Theta}_{nb})\boldsymbol{\omega}^b_{b/n} \tag{4.11}$$

where the transformation $\boldsymbol{T}_\Theta$ is as shown below [1, p. 25]:

$$\boldsymbol{T}_\Theta(\boldsymbol{\Theta}_{nb}) = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi/\cos\theta & \cos\phi/\cos\theta \end{bmatrix} \tag{4.12}$$

4.1.4 3-DOF maneuvering model

A ship can be fairly accurately modeled in the horizontal plane using 6 variables. Surge, sway, yaw, and their respective rates are sufficient to accurately model a ship's motion from positional measurements. In Figure 4.1 we can see the ship motion in 3 degrees of fredom.

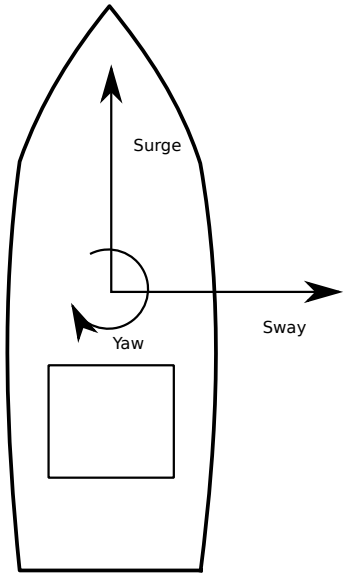


Figure 4.1: Ship 3-DOF motion.

In this case, the pitch θ and roll ϕ are often close to zero (unless the wave disturbance is very prominent), and can be assumed in our case to be zero. The heaving motion can also be disregarded. This then gives our three degrees of freedom (3-DOF) state vectors:

$$\boldsymbol{\nu} = \begin{bmatrix} u \\ v \\ r \end{bmatrix}, \quad \boldsymbol{\eta} = \begin{bmatrix} N \\ E \\ \psi \end{bmatrix} \tag{4.13}$$

When we assume that $\phi = \theta = 0$, we see that $\boldsymbol{T}_{\boldsymbol{\Theta}}(\boldsymbol{\Theta}_{nb}) = \boldsymbol{I}$, simplifying the relation between the angular velocity r and the yaw rate ψ,

to simply:

$$\dot{\psi} = r \tag{4.14}$$

The relation between $\boldsymbol{\eta}$ and $\boldsymbol{\nu}$ thus becomes rather simple:

$$\dot{\boldsymbol{\eta}} = \boldsymbol{R}(\psi)\boldsymbol{\nu} \tag{4.15}$$

Surge dynamics - Damping

The surge dynamics of a ship are rather complex [1, p. 122-128], and for our purposes unnecessary to model exactly, as the observers do not have direct knowledge of the ship's dynamics anyway. The motion of the ship will not be drastically different, and is still suited for the purposes of testing the observers. Therefore, we will model it as a simple linear damping model, given as

$$\tau_{D_u} = -\frac{1}{m} D_u u \tag{4.16}$$

$$\tau_{D_v} = -\frac{1}{m} D_v v \tag{4.17}$$

where D is a constant, and m is the mass of the ship. This is a rather unrealistic model, but it gives us a simple representation of a ship's dynamics, allowing us to create paths which can be tracked by the observers developed in the following sections. The ship's forward motion is controlled by the thrust input u_1.

Rudder dynamics

Simplified rudder dynamics can be given as

$$\dot{r} = u \sin u_2 - Cr \tag{4.18}$$

where u is the ship's forward speed, u_2 is the rudder angle input, r is the yaw rate and C is a rudder restoring force constant.

Full model in state form

We can now rewrite our ship dynamics model on the form $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$, with $\boldsymbol{x} = [\boldsymbol{\eta}, \boldsymbol{\nu}]^\top$:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} x_4 \cos x_3 - x_5 \sin x_3 \\ x_4 \sin x_3 + x_5 \cos x_3 \\ x_6 \\ m^{-1} u_1 - m^{-1} D x_4 \\ -m^{-1} D x_5 \\ x_4 \sin u_2 - C x_6 \end{bmatrix} \tag{4.19}$$

The ship's rudder and thrust are controlled by a simple PID controller.

4.1.5 Simplified model for Aircraft

To be able to simulate camera motion, a simplified model of Aircraft motion is needed. Aircraft motion is rather similar to ship motion, with important distinctions. The heaving motion of a ship along the z-axis denotes the altitude of the aircraft. In addition, unlike ship's, aircraft turn by initiating a roll.

A very simplified model for an Aircraft in the NED coordinate system is given by [2]:

$$\boldsymbol{\nu} = \begin{bmatrix} \boldsymbol{\nu}^n_{b/n} \\ \boldsymbol{\omega}^n_{b/n} \end{bmatrix} = \begin{bmatrix} U \\ V \\ W \\ P \\ Q \\ R \end{bmatrix}, \quad \boldsymbol{\eta} = \begin{bmatrix} \boldsymbol{p}^n_{b/n} \\ \boldsymbol{\Theta}_{nb} \end{bmatrix} = \begin{bmatrix} N \\ E \\ D \\ \Phi \\ \Theta \\ \Psi \end{bmatrix} \tag{4.20}$$

As we can see, the variables are capitalized, simply to differentiate them from ship variables.

Coordinate systems

For aircraft it is common to define three different body-fixed coordinate systems,

- Body axes,
- Stability axes,
- Wind axes.

The systems are shown in Figure 4.2. The *angle of attack* α, and *sideslip angle* β, are defined as:

$$\tan \alpha = \frac{W}{U} \tag{4.21}$$

$$\sin \beta = \frac{V}{V_T} \tag{4.22}$$

where

$$V_T = \sqrt{U^2 + V^2 + W^2} \tag{4.23}$$

is the total speed of the aircraft.

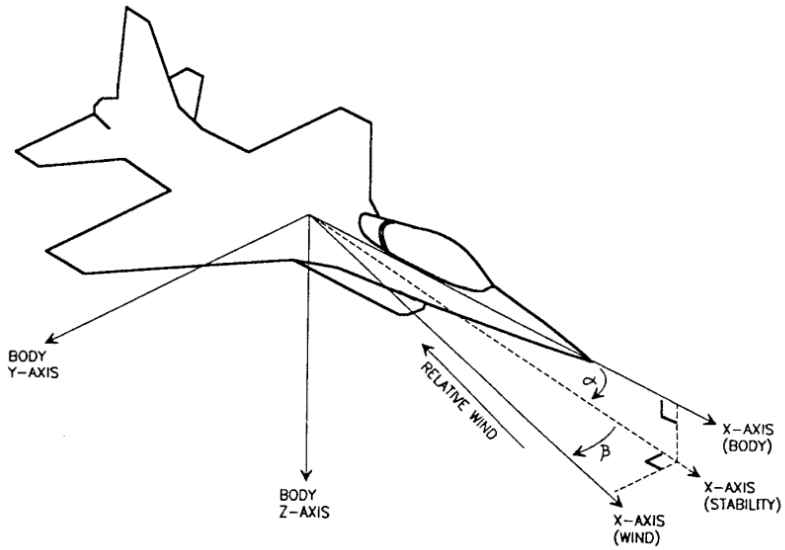


Figure 4.2: Definition of stability and wind axes for an aircraft [10].

Equations of motion

We are only interested in the motion of the camera, and not the particulars of the motion of the aircraft. To test our georeferencing and object tracking, we are only interested in allowing the aircraft to travel at a certain altitude, with no pitching or side-slip. The pitching motion is analogous to the rolling motion where the camera is concerned, and will not add any new information about the robustness of our system. We can therefore simplify further, by assuming that $\alpha = \beta = 0$, and in turn $V_T = U$.
The kinematic equations for translation are given just as for ship motion:

$$\dot{\boldsymbol{p}}^n_{b/n} = \boldsymbol{R}^n_b(\boldsymbol{\Theta}_{nb})\boldsymbol{\nu}^n_{b/n} \tag{4.24}$$

In addition, the kinematic equations for attitude are given below, just as for ships:

$$\dot{\boldsymbol{\Theta}}_{nb} = \boldsymbol{T}_\Theta(\boldsymbol{\Theta}_{nb})\boldsymbol{\omega}^b_{b/n} \tag{4.25}$$

Where $\boldsymbol{T}_\Theta(\boldsymbol{\Theta}_{nb})$ is defined just as in Equation (4.12), with $[\Phi, \Theta, \Psi]^\top$ replacing $[\phi, \theta, \psi]^\top$.

The total equations of motion in matrix form is given as for the ship before:

$$\begin{bmatrix} m\boldsymbol{I}_{3x3} & \boldsymbol{0}_{3x3} \\ \boldsymbol{0}_{3x3} & \boldsymbol{I}_g \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{\nu}}^b_{g/n} \\ \dot{\boldsymbol{\omega}}^b_{b/n} \end{bmatrix} + \begin{bmatrix} m\boldsymbol{S}(\boldsymbol{\omega}^b_{b/n}) & \boldsymbol{0}_{3x3} \\ \boldsymbol{0}_{3x3} & -\boldsymbol{S}(\boldsymbol{I}_g\boldsymbol{\omega}^b_{b/n}) \end{bmatrix} \begin{bmatrix} \boldsymbol{\nu}^b_{g/n} \\ \boldsymbol{\omega}^b_{b/n} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}^b_g \\ \boldsymbol{m}^b_g \end{bmatrix} \tag{4.26}$$

where m is the mass of the aircraft, $\boldsymbol{I}_g$ is the inertia tensor as defined in Equation (4.7). As we are interested in only a very specific subset of behavior from our model, we can disregard both gravity and air resistance. This may sound odd, but the motion of a fully modeled aircraft and our model will be rather similar, and more than sufficient for our purposes as simply a "camera carrier".

Bank-to-turn dynamics

When an aircraft banks, or rolls along its x-axis, it generates a moment about its yaw axis, or the z-axis. For a coordinated turn, where $\dot{\beta} = 0$, $\beta = 0$, $\Theta = 0$, $V_T = U_0 = constant$, we have that the yaw moment is given by [2, p. 24]:

$$R = \frac{W_0}{U_0} P + \frac{g}{U_0} \sin \Phi \tag{4.27}$$

Where g is the gravity constant. We assumed at the start that $\alpha = 0$, then $W_0 = 0$, therefore the moment reduces to:

$$R = \frac{g}{U_0} \sin \Phi \tag{4.28}$$

We can now rewrite our aircraft dynamics model on the form $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$, with $\boldsymbol{x} = [N, E, D, \Psi, \Phi, U]^\top$:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} x_6 \cos x_4 \\ x_6 \sin x_4 \\ 0 \\ x_6^{-1} g \sin x_5 \\ 0 \\ 0 \end{bmatrix} \tag{4.29}$$

4.2 Simple linear tracking with a discrete Kalman filter

The Kalman filter (KF) is a recursive filter, which can estimate the states of both linear and non-linear systems, regardless of noisy measurements. It was first published by Rudolf E. Kalman in 1960 [5]. When the process and measurement noise is white and Gaussian, for an observable linear system, the Kalman filter will provide an unbiased and minimum variance estimate, along with being the optimal state estimator for the system. Without any knowledge of the underlying model, we can use a simple discrete linear position tracking model, which tracks the N and E positions, and velocities.

4.2.1 Overview of Kalman filter variables

For a linear system with n states and m measurements, the various Kalman variables are given as:

- $\boldsymbol{Q}$ - (n x n) covariance matrix for process noise w_k.
- $\boldsymbol{R}$ - (m x m) covariance matrix for measurement noise v_k.
- $\bar{\boldsymbol{P}}_0$ - (n x n) initial error covariance matrix.
- $\bar{\boldsymbol{x}}_0$ - (n x 1) initial prior state conditions.
- $\boldsymbol{w_k}$ - (n x 1) process noise vector.
- $\boldsymbol{v_k}$ - (m x 1) measurement noise vector.
- $\boldsymbol{y_k}$ - (m x 1) measurement vector.
- $\boldsymbol{H}$ - (m x n) matrix connecting measurements to states.
- $\boldsymbol{K_k}$ - (n x m) Kalman gain matrix.
- $\hat{\boldsymbol{P}}$ - (n x n) estimated error covariance.
- $\hat{\boldsymbol{x}}_k$ - (n x 1) state estimate.

4.2.2 Kalman state model

A model for the position N can be given as:

$$N = \int v_N \, dt + N_0 \tag{4.30}$$

where v_N denotes the speed along the N axis, and N_0 is the initial position. The same model is used for E:

$$E = \int v_E \, dt + E_0 \tag{4.31}$$

This gives us the time derivative:

$$\dot{N} = v_N \tag{4.32}$$
$$\dot{E} = v_E \tag{4.33}$$

A linear state space model is given by the equation:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{E}\boldsymbol{w} \tag{4.34}$$
$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{v} \tag{4.35}$$

Choosing the state space vector as $\boldsymbol{x} = [N, E, v_N, v_E]^\top$, we get the transition matrix $\boldsymbol{A}$ and the input vector $\boldsymbol{B}$:

$$\boldsymbol{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.36}$$

The $\boldsymbol{E}$ matrix is chosen as $\boldsymbol{E} = \boldsymbol{I}$.
Now we can use this model with a discrete-time Kalman filter. The filter is given by:

$$\boldsymbol{x}(k+1) = \boldsymbol{\Phi}\boldsymbol{x}(k) + \boldsymbol{\Delta}\boldsymbol{u}(k) + \boldsymbol{\Gamma}\boldsymbol{w}_k \tag{4.37}$$
$$\boldsymbol{y}(k) = \boldsymbol{H}(k) + \boldsymbol{v}(k) \tag{4.38}$$

The transition matrix is given by [7]:

$$\boldsymbol{\Phi}(t) = e^{\boldsymbol{A}t} = \boldsymbol{I} + \boldsymbol{A}t + \frac{1}{2!}\boldsymbol{A}^2t^2 + \cdots + \frac{1}{i!}\boldsymbol{A}^it^i + \cdots \tag{4.39}$$

In our case, $\boldsymbol{A}^2 = \boldsymbol{0}_{4x4}$, so our model will be exact. Choosing a step size h, we can discretize our model using Euler integration:

$$\boldsymbol{\Phi}(h) = \boldsymbol{I} + \boldsymbol{A}h \tag{4.40}$$

$$= \begin{bmatrix} 1 & 0 & h & 0 \\ 0 & 1 & 0 & h \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.41}$$

There are no inputs as $\boldsymbol{B} = \boldsymbol{0}$, so we have that $\boldsymbol{\Delta} = \boldsymbol{0}$. Further, we have that [1, p. 296]:

$$\boldsymbol{\Gamma} = \left(\int_{\tau=0}^{h} \boldsymbol{\Phi}(\tau)\, d\tau \right) \boldsymbol{E} \tag{4.42}$$

$$= \boldsymbol{A}^{-1}(\boldsymbol{\Phi} - \boldsymbol{I})\boldsymbol{E} \tag{4.43}$$

However, in our case, $\boldsymbol{A}$ is a singular matrix, so $\boldsymbol{A}^{-1}$ is undefined. We can calculate $\boldsymbol{\Gamma}$ by integrating the $i = 1$ Euler integrated approximation of $\boldsymbol{\Phi}$:

$$\boldsymbol{\Gamma} = \left(\int_{\tau=0}^{h} (\boldsymbol{I} + \boldsymbol{A}\tau)\, d\tau \right) \boldsymbol{E} \tag{4.44}$$

$$= \left(\boldsymbol{I}h + \frac{1}{2}\boldsymbol{A}h^2 \right) \boldsymbol{E} \tag{4.45}$$

$$= \begin{bmatrix} h & 0 & \frac{h^2}{2} & 0 \\ 0 & h & 0 & \frac{h^2}{2} \\ 0 & 0 & h & 0 \\ 0 & 0 & 0 & h \end{bmatrix} \tag{4.46}$$

We have measurements of the first two states of the system, so the output matrix $\boldsymbol{H}$ is given as:

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{4.47}$$

4.2.3 Kalman filter operation

The filter process

The Kalman filter uses initial prior estimates of the system state $\bar{\boldsymbol{x}}_0$, as well as the initial error covariance to calculate the Kalman gain $\bar{\boldsymbol{P}}_0$ [6, p. 219]:

$$\boldsymbol{K}(k) = \bar{\boldsymbol{P}}(k)\boldsymbol{H}^\top(k)\left[\boldsymbol{H}(k)\bar{\boldsymbol{P}}(k)\boldsymbol{H}^\top(k) + \boldsymbol{R}(k)\right]^{-1} \tag{4.48}$$

where $\boldsymbol{R}(k)$ is the covariance matrix for the measurement noise $\boldsymbol{v}$. The internal estimate $\hat{\boldsymbol{x}}(k)$ can now be updated with the measurement $\boldsymbol{y}$(k):

$$\hat{\boldsymbol{x}}(k) = \bar{\boldsymbol{x}}(k) + \boldsymbol{K}(k)\left[\boldsymbol{y}(k) - \boldsymbol{H}(k)\bar{\boldsymbol{x}}(k)\right] \tag{4.49}$$

The updated error covariance matrix can now be calculated:

$$\hat{\boldsymbol{P}}(k) = \left[\boldsymbol{I} - \boldsymbol{K}(k)\boldsymbol{H}(k)\right]\bar{\boldsymbol{P}}(k)\left[\boldsymbol{I} - \boldsymbol{K}(k)\boldsymbol{H}(k)\right]^\top + \boldsymbol{K}(k)\boldsymbol{R}(k)\boldsymbol{K}^\top(k) \tag{4.50}$$

The state estimate can now finally be propagated, using the new estimates:

$$\bar{\boldsymbol{x}}(k+1) = \boldsymbol{\Phi}(k)\hat{\boldsymbol{x}}(k) + \boldsymbol{\Delta}(k)\boldsymbol{u}(k) \tag{4.51}$$

And the error covariance matrix:

$$\bar{\boldsymbol{P}}(k+1) = \boldsymbol{\Phi}(k)\hat{\boldsymbol{P}}(k)\boldsymbol{\Phi}^\top(k) + \boldsymbol{\Gamma}(k)\boldsymbol{Q}(k)\boldsymbol{\Gamma}^\top(k) \tag{4.52}$$

where $\boldsymbol{Q}(k)$ is the covariance matrix for the process noise. The two matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ are the covariance matrices for the noise. When the exact properties of the noise are not known, the filter can be tuned by changing their values. Testing of the filter is done in the simulation chapter.

4.3 Nonlinear model with Extended Kalman Filter

For non-linear systems, the Extended Kalman filter (EKF) can be used. In an EKF, the system equations are linearized about the current best estimate, and has proven itself as a good estimator in real-world applications. The EKF, however, is not necessarily optimal, and may diverge in certain situations where the linearized estimate is too far from the true state. The EKF can be applied to nonlinear systems on the form:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{E}\boldsymbol{w} \tag{4.53}$$
$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{v} \tag{4.54}$$

We choose the Kalman state vector to estimate both $\boldsymbol{\eta}$ and $\boldsymbol{\nu}$:

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} N \\ E \\ \psi \\ u \\ v \\ r \end{bmatrix} \tag{4.55}$$

The Kalman model is given as:

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} x_4 \cos x_3 - x_5 \sin x_3 \\ x_4 \sin x_3 + x_5 \cos x_3 \\ x_6 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.56}$$

The system *Jacobian* matrix which is determined as:

$$\frac{\delta \boldsymbol{f}}{\delta \boldsymbol{x}} = \begin{bmatrix} 0 & 0 & -x_4 \sin x_3 + x_5 \cos x_3 & \cos x_3 & -\sin x_3 & 0 \\ 0 & 0 & x_4 \cos x_3 - x_5 \sin x_3 & \sin x_3 & \cos x_3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.57}$$

We do not have knowledge of the inputs to the system so $\boldsymbol{B} = \boldsymbol{0}$, and we have to rely on measurements to estimate the states. $\boldsymbol{E}$ is chosen as $\boldsymbol{E} = \boldsymbol{I}$. We have access to the N and E measurements from the geo-referencing system, so our 2x6 observation matrix $\boldsymbol{H}$ will be:

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.58}$$

The Kalman system matrix is determined by:

$$\boldsymbol{\Phi}(k) \approx \boldsymbol{I} + h \frac{\delta \boldsymbol{f}}{\delta \boldsymbol{x}}(k) \tag{4.59}$$

Finally, $\boldsymbol{\Gamma}(k)$ is determined by:

$$\boldsymbol{\Gamma}(k) = h\boldsymbol{E} \tag{4.60}$$

4.3.1 Extended Kalman filter operation

The EKF works rather similarly to the linear Kalman filter. Initial prior estimate of system state $\bar{\boldsymbol{x}}_0$ and initial error covariance $\bar{\boldsymbol{P}}$ are used to calculate the Kalman gain, as done in (4.48). The state estimate is updated in the same manner as in (4.49). The error covariance update is done as in (4.50). The state estimate propagation is however different. The state estimate is propagated along the linearized version of the nonlinear system:

$$\bar{\boldsymbol{x}}(k+1) = \hat{\boldsymbol{x}}(k) + h\left[\boldsymbol{f}(\hat{\boldsymbol{x}}(k)) + \boldsymbol{B}\boldsymbol{u}(k)\right] \tag{4.61}$$

The error covariance is propagated in the same manner as in 4.52, however with a different method of calculating $\boldsymbol{\Phi}(k)$.

4.4 Observability

In both the linear and nonlinear case, the system states need to be observable in order for the Kalman filter to be able to estimate them. For the linear case, the observability can be determined by calculating the observability matrix $\mathcal{O}$. For a linear, time-invariant system with n states,

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \tag{4.62}$$

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} \tag{4.63}$$

$$\mathcal{O} = \begin{bmatrix} \boldsymbol{H}^\top & \boldsymbol{A}^\top\boldsymbol{H}^\top & \cdots & (\boldsymbol{A}^\top)^{n-1}\boldsymbol{H}^\top \end{bmatrix} \tag{4.64}$$

must be of full column rank, such that at least a left inverse exists [1, p. 292].

4.4.1 Linear filter observability

We can verify this for the linear filter,

$$\mathcal{O} = \begin{bmatrix} \boldsymbol{I}_{4x4} & \boldsymbol{0}_{4x4} \end{bmatrix} \tag{4.65}$$

which has full column rank$(\mathcal{O}) = 4$.

4.4.2 Nonlinear filter observability

For the nonlinear observer, the case is more complex. Observability of certain states may depend on whether other states are stationary, such as is the case for our ship model. An estimate of the ship yaw angle ψ cannot be determined when only measuring position, if the ship is not moving. Theory for observability and weak observability of nonlinear systems has been developed by Hermann and Krener [11], while an overview is given by [12]. We can test for local observability for a system on the form

$$\begin{aligned} \dot{\boldsymbol{x}} &= f(\boldsymbol{x}, \boldsymbol{u}) \\ \boldsymbol{z} &= h(\boldsymbol{x}) \end{aligned} \tag{4.66}$$

where $\boldsymbol{z}$ is a (m x 1) vector, and $\boldsymbol{x}$ is the (n x 1) state vector. The $\mathcal{G}(\boldsymbol{x}, \boldsymbol{u}^\star)$ matrix, which is the set of all finite linear combinations of the Lie derivatives of $\boldsymbol{h}(\boldsymbol{x})$ is given by:

$$\mathcal{G}(\boldsymbol{x}, \boldsymbol{u}^\star) = \begin{bmatrix} L_f^0 h_1(\boldsymbol{x}) \\ \vdots \\ L_f^0 h_m(\boldsymbol{x}) \\ \vdots \\ L_f^{n-1} h_1(\boldsymbol{x}) \\ \vdots \\ L_f^{n-1} h_m(\boldsymbol{x}) \end{bmatrix} \tag{4.67}$$

Where L is the Lie derivate operator, defined as

$$\begin{aligned} L_f \boldsymbol{h}(\boldsymbol{x}) &= \frac{\delta \boldsymbol{h}}{\delta \boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x}) \\ L_f^2 \boldsymbol{h}(\boldsymbol{x}) &= \frac{\delta (L_f \boldsymbol{h})}{\delta \boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x}) \\ L_f^k \boldsymbol{h}(\boldsymbol{x}) &= \frac{\delta (L_f^{k-1} \boldsymbol{h})}{\delta \boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x}) \end{aligned} \tag{4.68}$$

The observability matrix $\mathcal{O}(\boldsymbol{x}, \boldsymbol{u}^\star)$ is then given as:

$$\mathcal{O}(\boldsymbol{x}, \boldsymbol{u}^\star) = \frac{\delta \mathcal{G}}{\delta \boldsymbol{x}} \tag{4.69}$$

The observability matrix must have rank($\mathcal{O}$) = n for the system to be observable, as for the linear system.

Observability using two inputs with nonlinear observer

In the case of two measurements (N and E), the measurement vector is

$$\boldsymbol{h}(\boldsymbol{x}) = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{4.70}$$

The Lie derivative matrix then becomes

$$\mathcal{G}(\boldsymbol{x}, \boldsymbol{u}^\star) = \begin{bmatrix} x_1 \\ x_2 \\ x_4 \cos x_3 - x_5 \sin x_3 \\ x_5 \cos x_3 + x_4 \sin x_3 \\ -x_6(x_5 \cos x_3 + x_4 \sin x_3) \\ x_6(x_4 \cos x_3 - x_5 \sin x_3) \\ -x_6^2(x_4 \cos x_3 + x_5 \sin x_3) \\ -x_6^2(x_5 \cos x_3 - x_4 \sin x_3) \\ x_6^3(x_5 \cos x_3 + x_4 \sin x_3) \\ -x_6^3(x_4 \cos x_3 - x_5 \sin x_3) \\ x_6^4(x_4 \cos x_3 + x_5 \sin x_3) \\ x_6^4(x_5 \cos x_3 - x_4 \sin x_3) \end{bmatrix} \tag{4.71}$$

Further, we see that the observability matrix is

$$\mathcal{O}(\boldsymbol{x}, \boldsymbol{u}^\star) = \frac{\delta \mathcal{G}}{\delta \boldsymbol{x}} = \begin{bmatrix} r_{11} & \cdots & r_{16} \\ \vdots & \ddots & \vdots \\ r_{61} & \cdots & r_{66} \end{bmatrix} \tag{4.72}$$

which has $\text{rank}(\mathcal{O}) = 5 \neq n$. All system states are therefore not observable by the two-input observer. This will be evident in simulations done in Chapter 5.

Observability using three inputs with nonlinear observer

Adding yaw-feedback to the observer allows the observer to estimate both yaw and yaw rate, gives us

$$\boldsymbol{h}(\boldsymbol{x}) = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{4.73}$$

Carrying out the same calculations as before, we can see that the new 18x6 observability matrix has $\text{rank}(O) = 6 = n$, making the system fully observable.

4.5 Summary

In this chapter, theory of ship motion and simplified aircraft motion has been developed. Linear and non-linear observers, based on Kalman filters, have been developed for ships and icebergs, and full rank observability has been shown for the linear observer and the three-input nonlinear observer.

Chapter 5

Simulation

In the previous chapters, theory for ship and object tracking, along with geo-referencing using a camera have been derived. Before the tracking system was tested on real flight data, simulated flights over simulated ships have been done, using the models described in the previous chapters. All code used to generate the plots contained in this chapter can be obtained by request, or through DAIM.

5.1 Overview

Several scenarios will be simulated. The observers will be tested on a ship model with and without noise, and the observer will be tested in a simulated aircraft with camera attached. The following scenarios are considered:

- A ship, following a preset pattern, is tracked with the observers at differing sampling rates, without noise.
- The nonlinear observer is tested with yaw estimate feedback.
- The previous scenarios are tested with measurement noise.
- Effects of tuning will later be discussed for the nonlinear yaw-feedback observer.

The scenarios are chosen to verify the performance of the observers, along with establishing whether there is a need for more complex observers, or more input measurements. In the following sections, we have defined the total speed of the craft as:

$$v_{tot} = \sqrt{u^2 + v^2} \tag{5.1}$$

This is necessary in the linear observer model, as the speed of the object is not tracked directly by the model, only the decomposed speeds $x_3 = \dot{N}$ and $x_4 = \dot{E}$. For the linear observer, the total speed is therefore given as:

$$v_{tot} = \sqrt{x_3^2 + x_4^2} \tag{5.2}$$

5.1.1 Simulated ship path

Our ship model will follow a simple path, defined by the following inputs:

1. At time $t = 0$s, accelerate to 4 m/s.

2. At time $t = 10$s, turn right to 90° degree heading.

3. At time $t = 25$s, turn left to 270° degree heading, and increase forward speed to 5 m/s.

4. The simulation continues until $t = 60s$.

The path taken by the simulated ship is shown in Figure 5.1.

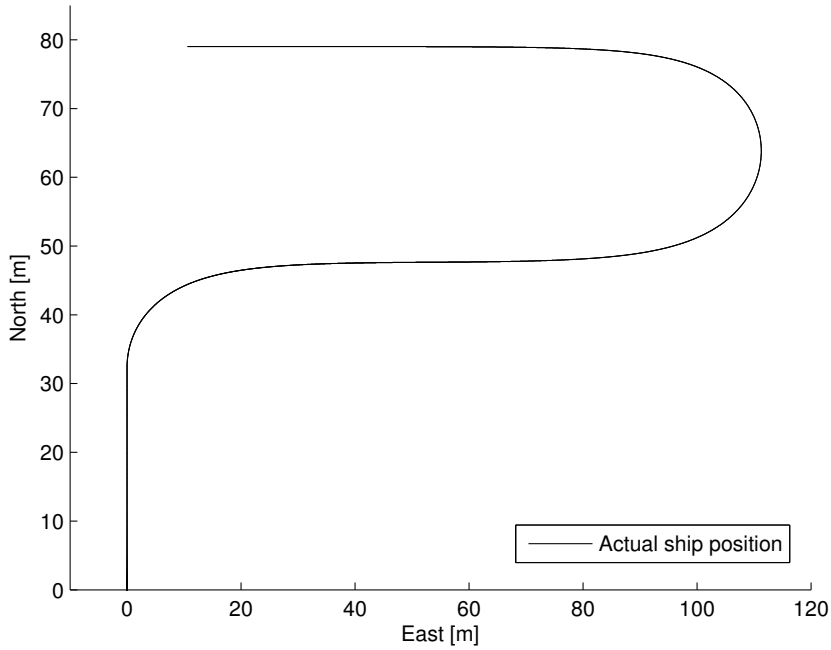


Figure 5.1: Predefined path of simulated ship model.

5.1.2 Sampling rate

The system is simulated with a step size of $h = 0.01$ seconds. The measurements however are based on the frame-rate of the thermal imaging camera, which is 8 fps. This means that it can supply a frame every $t = \frac{1}{8}s$. We assume that our object detection algorithms will be able to run quickly enough to process these frames and supply measurements every $t = 0.125s$. Lower sampling rates will be considered later.

5.2 Linear observer

In the following section, the linear observer described in Chapter 4.2 is tested on the model following the path previously mentioned.

5.2.1 Simulation without noise

In this case, the covariance matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ were chosen by trial and error. The measurement covariance matrix $\boldsymbol{R}$ was chosen to reflect the knowledge of the noise on the measurement:

$$R = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix} \tag{5.3}$$

The variance of the measurement with no noise is 0, however setting the $\boldsymbol{R}$ matrix as $\boldsymbol{R} = \mathbf{0}$ introduces high frequency oscillations into the estimates. This is most likely due to numerical approximation problems, and will be discussed later in the discussion chapter.

$$Q = \begin{bmatrix} 0.5 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0.5 \end{bmatrix} \tag{5.4}$$

The $\boldsymbol{Q}$ matrix was chosen to yield the best results, by trial and error. The prior error covariance matrix was chosen as

$$\bar{\boldsymbol{P}} = \mathbf{0} \tag{5.5}$$

As we can see from Figures 5.2 and 5.3, the position errors are rather small, on the order of a few meters. However, every time the ship turns, the filter tends to overshoot, and compensate later. Additionally, near the end of the simulation, we can see that the error in E position does not converge to zero, giving a constant deviation of around 0.3 meters.

As we can see from Figure 5.4, the error in forward speed is substantial at first, but the filter tends to converge to the correct speed value, with some small estimation errors where the ship turns.

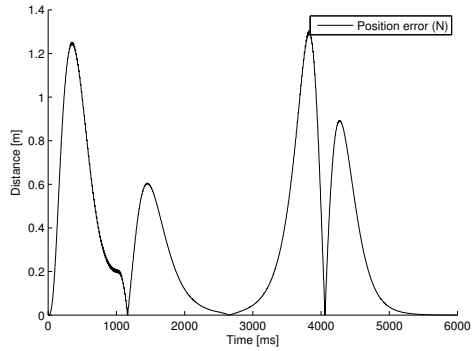


Figure 5.2: Error in N position for linear observer without noise.

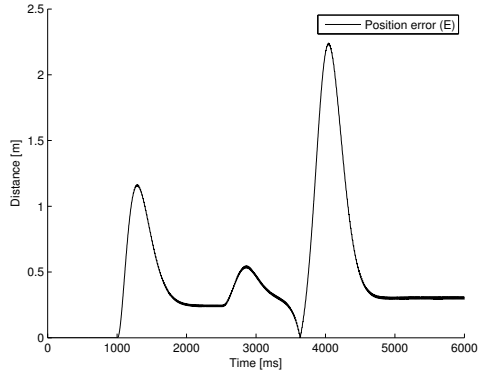


Figure 5.3: Error in E position for linear observer without noise.

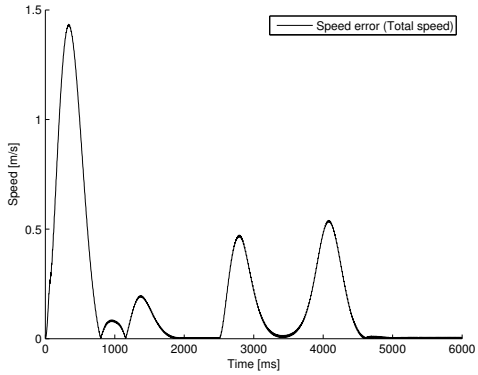


Figure 5.4: Error in total speed for linear observer without noise.

5.2.2 Simulation with noise

Simulating the system without noise is rather optimistic, we must assume that the measurement will be noisy. A zero-mean white noise v with $\mathrm{Var}(v) \approx 2$ was introduced to the measurement. The covariance matrices were chosen as:

$$\boldsymbol{R} = \begin{bmatrix} \mathrm{Var}(v) & 0 \\ 0 & \mathrm{Var}(v) \end{bmatrix} \approx \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \tag{5.6}$$

The $\boldsymbol{Q}$ matrix was chosen by trial and error as:

$$Q = 0.1\boldsymbol{I}_{6x6} \tag{5.7}$$

The prior error covariance estimate was chosen as:

$$\bar{\boldsymbol{P}} = \boldsymbol{I}_{6x6} \tag{5.8}$$

As we can see from Figure 5.5, the filter estimate (red line) does an adequate job of following the position of the ship, however the overshooting persists. The error magnitude has also become much greater than without noise. This is especially evident in Figure 5.7, which shows a large deviation in the E position of up to 6 meters. The north position however stays below 2.5 meters of error, as seen in Figure 5.6.

The total speed error, as shown in Figure 5.8, is unfortunately rather large throughout, being at certain points over half the size of the actual speed, showing the limitations of this type of linear observer for a rather complex ship system. This model however would be better suited for the slow and non-directional motion of an iceberg, which does not have any input forces directing its motion, other than wave and wind disturbances, which can be modeled as noise. However, we can see that with process noise, the estimate seems to converge to the correct position once the ship follows a straight path.

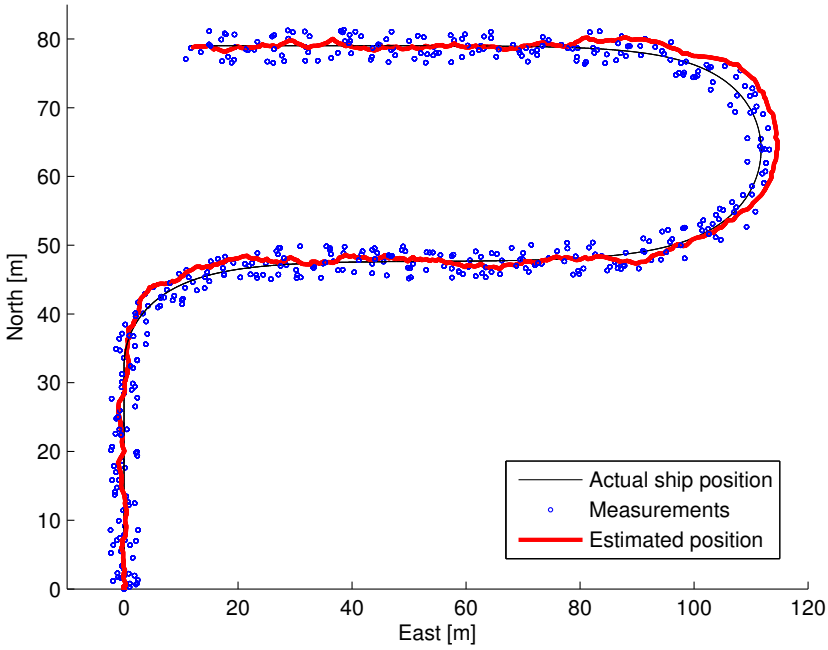


Figure 5.5: Linear observer tracking with measurement noise.

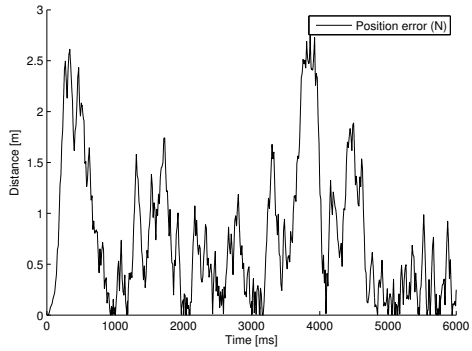


Figure 5.6: Error in N position for linear observer with noise.

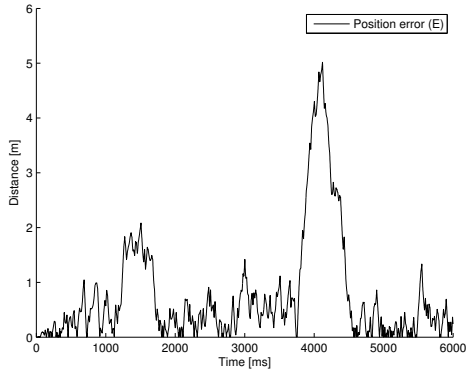


Figure 5.7: Error in E position for linear observer with noise.

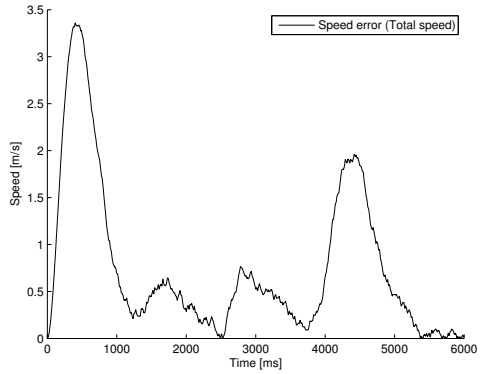


Figure 5.8: Error in total speed for linear observer with noise.

5.3 Nonlinear observer

The nonlinear Kalman observer described in Chapter 4.3 is tested on the same predefined pattern as the linear observer.

5.3.1 Simulation without noise

In the first simulation, we only have feedback on the N and E positions of the ship being tracked. All other states are determined from the model. The covariance matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ are chosen to reflect the knowledge of noise in the measurement:

$$\boldsymbol{R} = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix} \tag{5.9}$$

Due to the same issue encountered with simulating the linear observer, the covariance matrix $\boldsymbol{R}$ can not be set as $\boldsymbol{R} = \boldsymbol{0}$ due to the high frequency oscillations which would arise. This will be discussed later. The $\boldsymbol{Q}$ matrix is chosen by trial and error to be a diagonal matrix with identical terms:

$$\boldsymbol{Q} = 0.1\boldsymbol{I}_{6x6} \tag{5.10}$$

As before, prior error estimate is $\bar{\boldsymbol{P}} = \boldsymbol{0}_{6x6}$. As we can see, the system tracks the position rather well, within about a meter, with the exception that the East position shown in Figure 5.10 shows a constant deviation. The North position however is tracked rather well, see Figure 5.9. In addition, as we can see in Figure 5.13, the lateral speed error shows a constant deviation, where in reality the lateral speed is $v = 0$. The filter fails to determine this due to the fact that it does not have any yaw feedback, and as mention in Chapter 4.4, the system is not fully observable, thus estimating the ship to be moving with a slight yaw angle offset, as is evidenced by Figure 5.11, causing the filter to estimate a side-slipping motion. The forward speed is estimated rather well, as can be seen from Figure

5.12, however the lateral speed v shows a constant deviation, Figure 5.13.

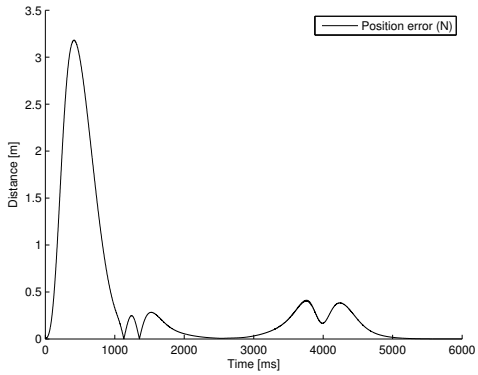


Figure 5.9: N error for nonlinear observer without noise.

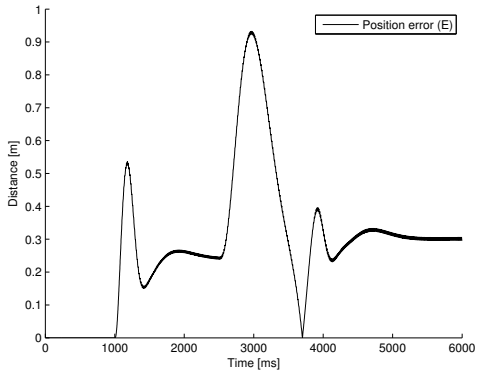


Figure 5.10: E error for nonlinear observer without noise.

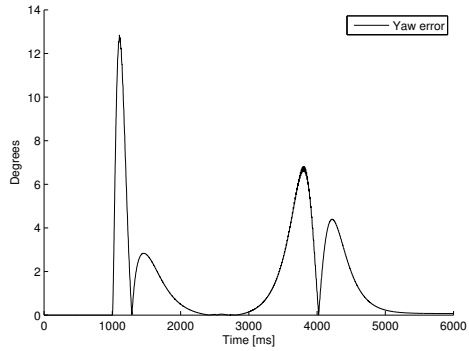


Figure 5.11: Yaw error for nonlinear observer without noise.

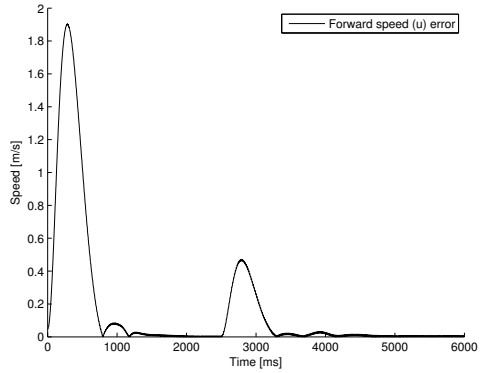


Figure 5.12: u error for nonlinear observer without noise.

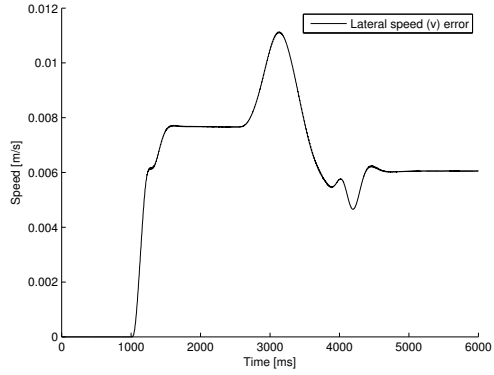


Figure 5.13: v error for nonlinear observer without noise.

5.3.2 Simulation without noise, with yaw feedback

The nonlinear model tested in the previous section seems to perform quite well, but does tend to have some inaccuracies due to the limited state measurement. Introducing measurement of yaw, using the method which was discussed in Chapter 3.2, will allow the filter to more easily estimate yaw, and remove constant errors from the speed estimates. The covariance matrices were chosen as

$$\boldsymbol{R} = \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.1 \end{bmatrix} \tag{5.11}$$

where $\boldsymbol{R} \neq \boldsymbol{0}$ due to the high frequency issues mention earlier, and the $\boldsymbol{Q}$ matrix was chosen by trial and error as:

$$\boldsymbol{Q} = 0.1\boldsymbol{I}_{6x6} \tag{5.12}$$

As we can see from comparing the yaw estimates without feedback in Figure 5.11 and with feedback in 5.18, we see that the yaw estimate is somewhat better than without feedback. The reason for the high discrepancies is most likely due to the sampling rate and values chosen for the $\boldsymbol{Q}$ matrix. However, we can now see that the constant v deviation is no longer present, see Figure 5.17. The observer performs rather well, the N position is tracked rather well with some small discrepancies shown in Figure 5.14, however the constant E deviation still persists.

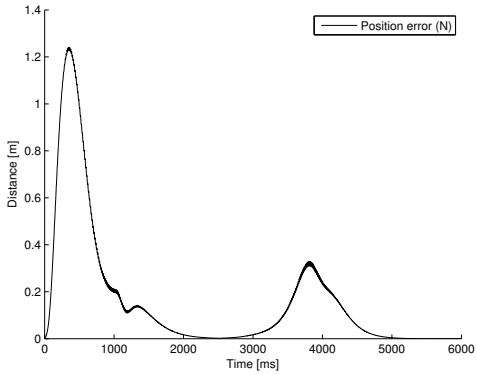


Figure 5.14: N position error for nonlinear observer without noise with yaw feedback.

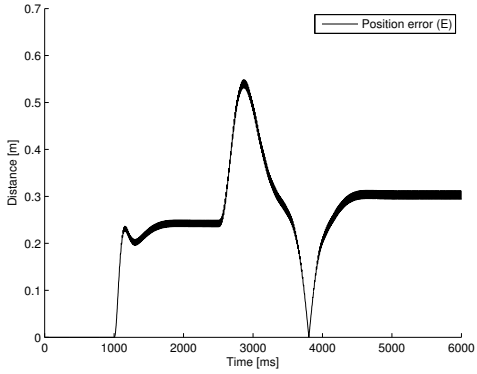


Figure 5.15: E position error for nonlinear observer without noise with yaw feedback.

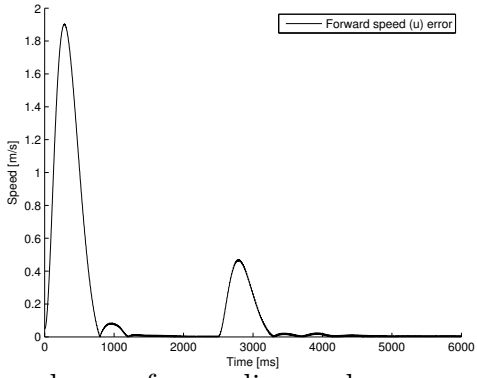


Figure 5.16: u speed error for nonlinear observer without noise with yaw feedback.

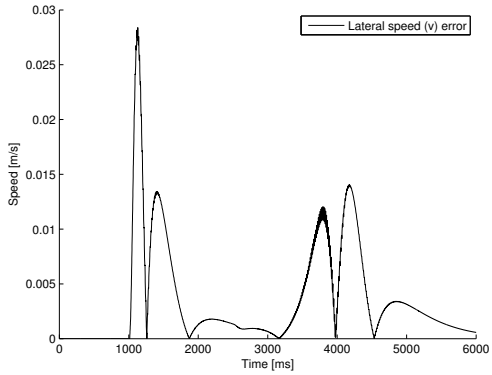


Figure 5.17: v speed error for nonlinear observer without noise with yaw feedback.

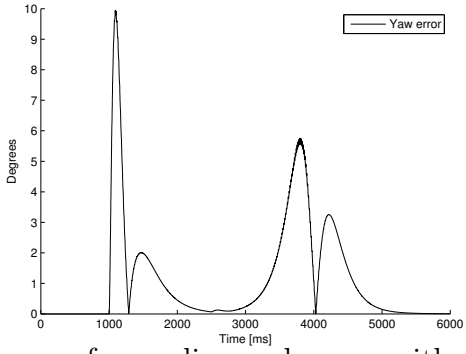


Figure 5.18: Yaw error for nonlinear observer without noise and with yaw feedback.

5.3.3 Simulation with noise, with no yaw feedback

Introducing measurement noise better shows the difference in performance between the linear and nonlinear observers. The observer does a much better job at tracking the actual position of the ship than the linear observer with noise, as can be seen when comparing Figure 5.19 and the linear observer Figure 5.5.

In this case, the measurement was added zero-mean white noise v with variance $\text{Var}(v) \approx 2$. The $\boldsymbol{R}$ matrix was chosen to reflect the variance of the noise:

$$\boldsymbol{R} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \tag{5.13}$$

The $\boldsymbol{Q}$ matrix was chosen by trial-and-error as

$$\boldsymbol{Q} = 0.01\boldsymbol{I}_{6x6}, \tag{5.14}$$

and the prior error covariance estimate was chosen as $\bar{\boldsymbol{P}}_0 = \boldsymbol{0}_{6x6}$. As we can see from Figure 5.20, the N estimate seems to converge to zero. The large discrepancy in the beginning is due to filter tuning, as we can see from Figure 5.23, the forward speed takes some time to converge, which in turn affects the N position. The v speed is shown in Figure 5.24, and it is still visible that determining the speed properly is difficult without yaw feedback. The total speed, shown in Figure 5.25 shows the discrepancies better. If the u and v speeds were tuned more responsively, the speed would converge more quickly, however the estimate would be noisier, which in turn would give a noisier position estimate. Proper tuning of filters will be discussed later in the discussion chapter. In general, we can see that the filter does alright for the positional estimates, and the speed errors are acceptable. The yaw estimate however, shows large discrepancies, as shown in Figure 5.22. This is consistent with the results of Chapter 4.4, which shows that this observer does not have full rank, and therefore cannot correctly estimate the yaw angle.

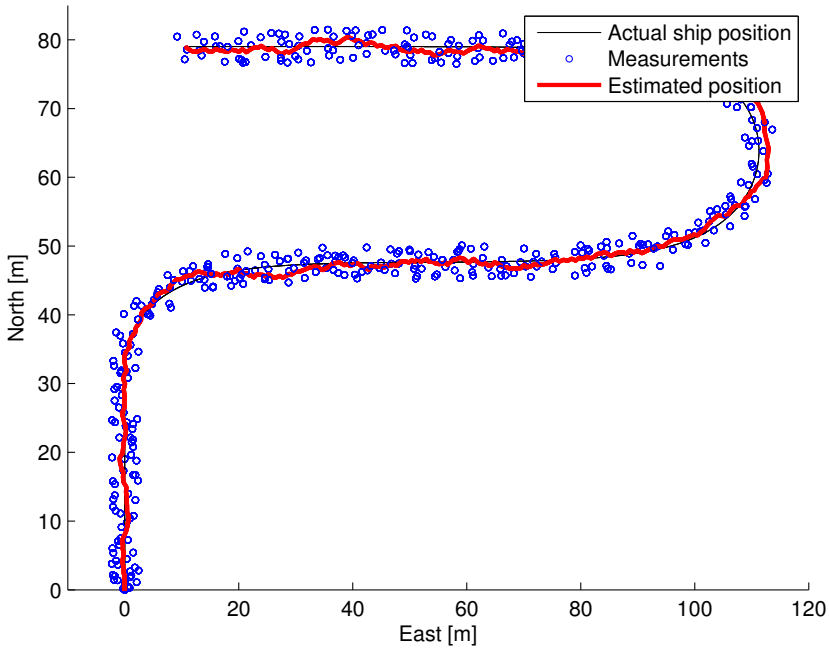


Figure 5.19: Nonlinear observer tracking with measurement noise.

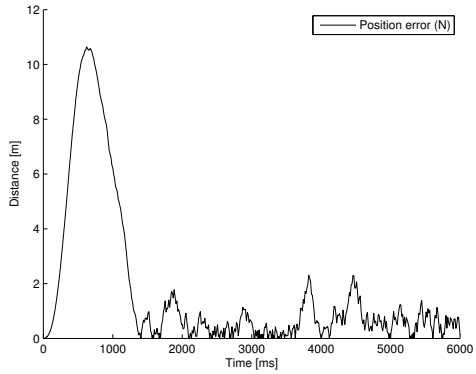


Figure 5.20: Error in N position of nonlinear observer with noise.

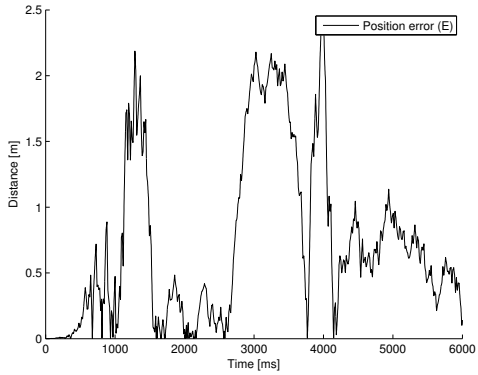


Figure 5.21: Error in E position of nonlinear observer with noise.

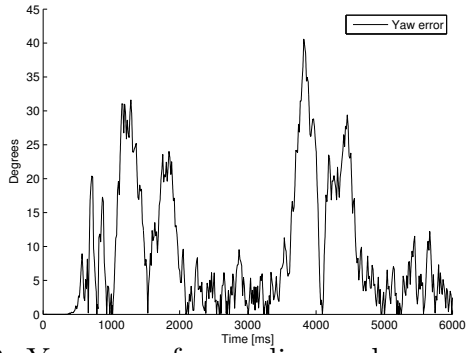


Figure 5.22: Yaw error for nonlinear observer with noise.

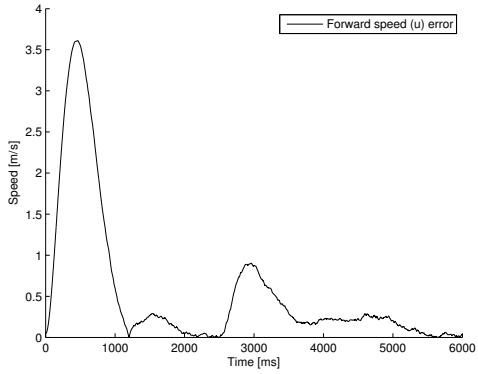


Figure 5.23: u error for nonlinear observer with noise.

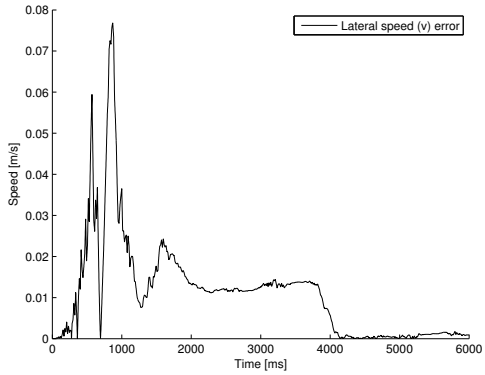


Figure 5.24: v error for nonlinear observer with noise.

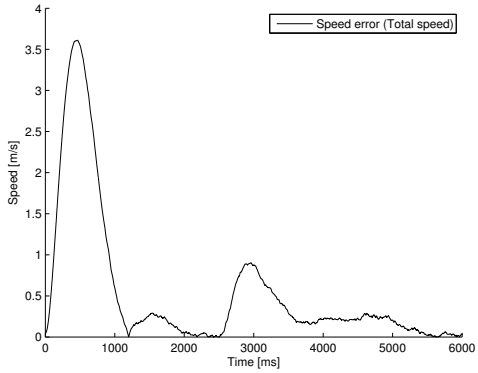


Figure 5.25: Error in total speed for nonlinear observer with noise.

5.3.4 Simulation with noise, with yaw feedback

As we can see from the previous simulation, the observer estimates yaw rather poorly. Including a measurement of the yaw, as explained in Chapter 3.2, allows the filter to more properly estimate both yaw, position and speeds. The position measurement has a noise variance $\text{Var}(v_p) \approx 2$, and the yaw measurement has a variance $\text{Var}(v_y) \approx 8 \cdot 10^{-4}$. The measurement covariance matrix $\boldsymbol{R}$ was chosen to reflect the noise:

$$\boldsymbol{R} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 8 \cdot 10^{-4} \end{bmatrix} \tag{5.15}$$

The $\boldsymbol{Q}$ matrix is chosen as

$$\boldsymbol{Q} = 0.1\boldsymbol{I}_{6x6}, \tag{5.16}$$

while the initial prior error covariance estimate is chosen as $\boldsymbol{P}_0 = \boldsymbol{0}$.

As we can see, the yaw error is now substantially smaller when we compare Figure 5.31 with 5.22. After the initial discrepancy, the N position estimate performs better than without yaw feedback, as seen by comparing Figures 5.20 and 5.27. The effects of yaw feedback on the speed estimate is visible in the total speed estimation shown in Figure 5.32. The v speed now doesn't show large discrepancies or constant biases, as seen in Figure 5.17. As we can see, with minimal tweaking, the filter performs rather well at tracking the ship and estimating its states. Tuning the filter for better performance will be discussed later.

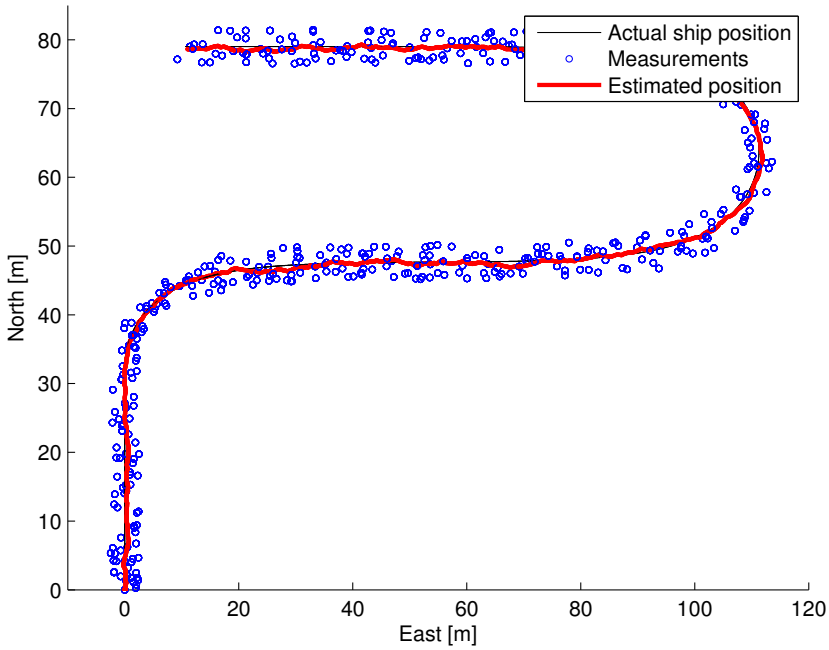


Figure 5.26: Nonlinear observer tracking with measurement noise, with yaw feedback.

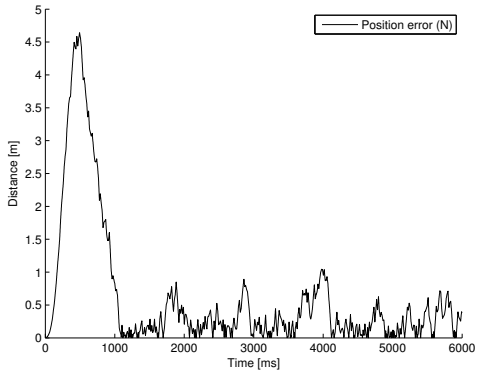


Figure 5.27: N position error for nonlinear observer with noise, with yaw feedback.

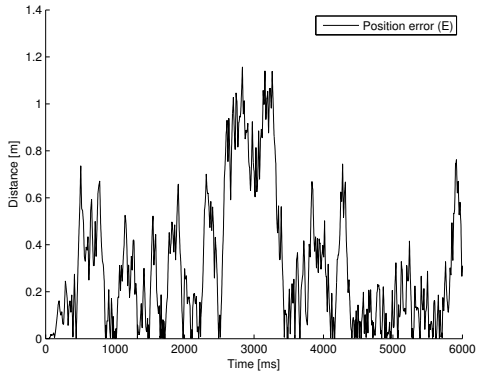


Figure 5.28: E position error for nonlinear observer with noise, with yaw feedback.

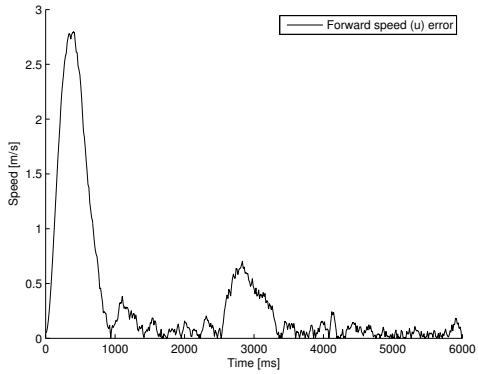


Figure 5.29: u speed error for nonlinear observer with noise, with yaw feedback.

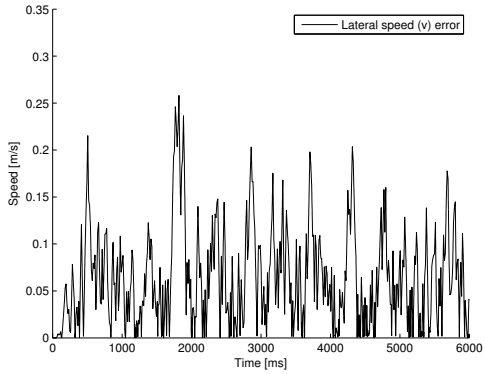


Figure 5.30: v speed error for nonlinear observer with noise, with yaw feedback.

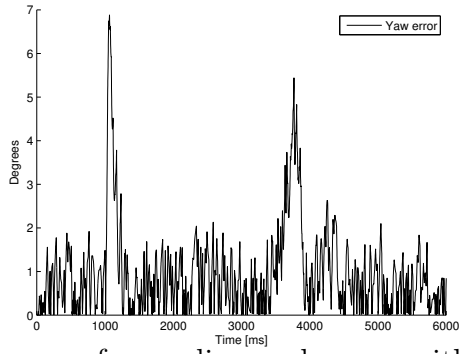


Figure 5.31: Yaw error for nonlinear observer with noise, with yaw feedback.

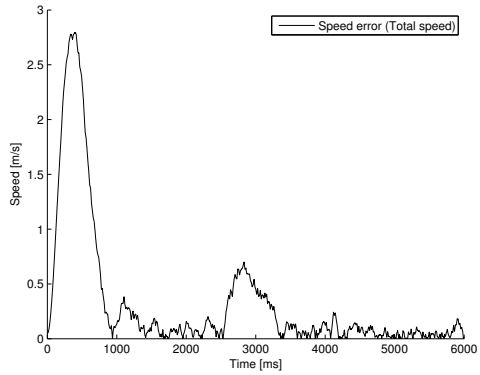


Figure 5.32: Error in total speed for nonlinear observer with yaw feedback.

5.4 Summary

In this chapter, the linear and nonlinear observers were tested on different scenarios, with and without noise. It shows that the linear observer works rather well, but is surpassed by the nonlinear observer, which more properly estimates both the speeds and position of the ship, along with estimating the ship's yaw.

Chapter 6

Test Platform

6.1 Skywalker X8 flying wing

The Skywalker X8 flying wing[1] is a simple, lightweight UAS/UAV airframe, which will be used for preliminary testing, and data acquisition. The airframe itself is 2120 mm wide, with a MTOW of 3.5 kg. It is shown in Figure 6.1. A shot of the internals is shown in Figure 6.2. The airframe will, in addition to necessary avionics and propulsion systems, carry a payload consisting of a gimballed FLIR thermal imaging camera. The main mission is to use the platform to log data for further analysis, so that a system for real time object detection and processing may be developed for the UAV Factory Penguin B[2].

[1] http://www.hobbyking.com/HOBBYKING/store/__27132__Skywalker_X_8_FPV_UAV_Flying_Wing_2120mm.html

[2] http://www.uavfactory.com/product/46

Figure 6.1: X8 frame, with wings mounted.

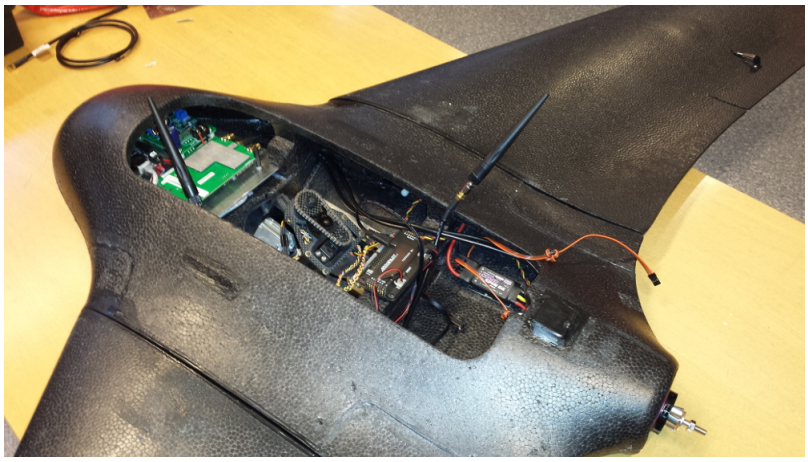


Figure 6.2: X8 frame with payload and gimbal mounted.

6.2 X8 Payload

The X8 payload consists of the following equipment:

- Beaglebone Black[3]
- AXIS M7001 A/D video converter[4]
- FLIR Tau 2 Uncooled LWIR camera[5]
- Retractable gimbal mount for FLIR camera BTC-88[6]
- Ubiquity Rocket M5 (5 GHz)[7]
- 3-port in-house made switch (with serial to ethernet-interface)
- 48V step-up converter
- 5V step-down converter

The avionics are supplied by the ArduPilot, which will be connected to the in-house switch with a USB-interface, which allows the software running on the BeagleBone access to aircraft attitude and positional information.

[3]http://beagleboard.org/BLACK
[4]http://www.axis.com/en/products/cam_m7001/index.htm
[5]http://www.flir.com/cvs/cores/view/?id=54717
[6]http://www.microuav.com/btc88
[7]http://dl.ubnt.com/rocketM5_DS.pdf

6.3 Payload communications

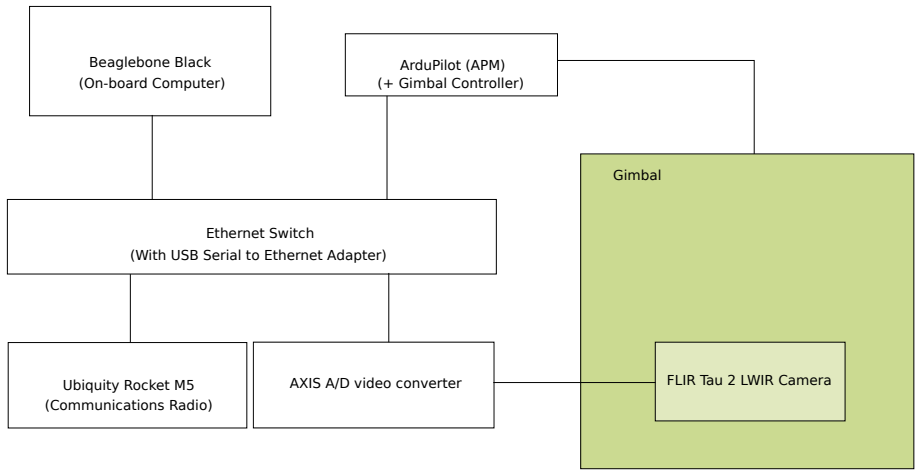
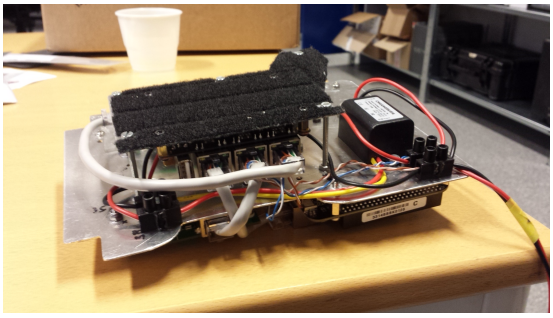


Figure 6.3: The payload communications diagram

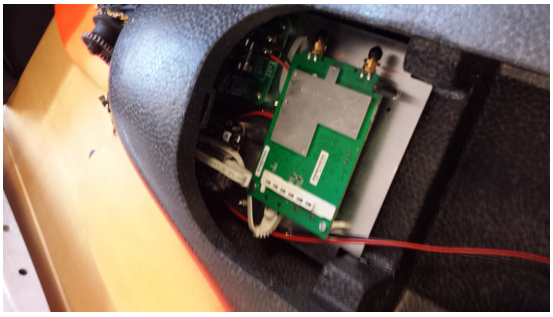
6.4 Construction

The X8 payload went through an iterative design, where design elements were changed during production due to limitations in weight, and changing the switch type from off the shelf to in-house made. The original design contained a 5-port Ethernet switch, which was changed for a 3-port switch capable of communicating with the ArduPilot directly. This allows us to read ArduPilot data through a Serial-to-Ethernet interface.

A 2-mm aluminium plate was cut to fit in the front compartment of the X8 airframe, measuring 15x20cm. The floor of the front compartment is covered in velcro, so a soft velcro locking frame was created and fitted to the plate, see Figure 6.4. The components were fitted to the board. Initially the cables were chosen as available, but due to weight constraints, bespoke, proper length cables and POE injectors were created, decluttering the payload. The top of the payload is shown in Figure 6.5.



(a) Velcro lined, raised bottom of payload assembly.



(b) Payload mounted in nose of aircraft.

Figure 6.4: Payload mounting.

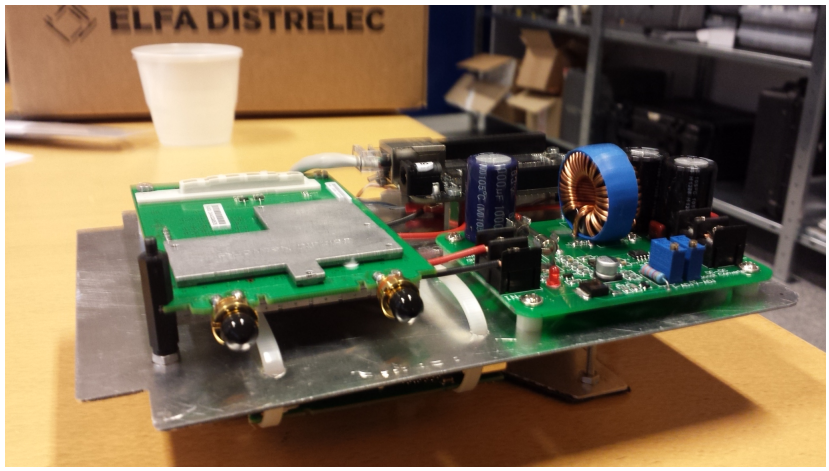


Figure 6.5: Top view of payload assembly, showing 48V step-up, BeagleBone and Ubiquity radio.

6.5 Gimbal mount

The FLIR thermal camera will be mounted in a BTC-88 gimbal mounted on a retractor mechanism. The retractor mechanism allows the gimbal to deploy underneath and retract into the airframe. A hole was drilled in the foam body of the X8 to allow the gimbal and retractor to fit, as seen in Figure 6.6.

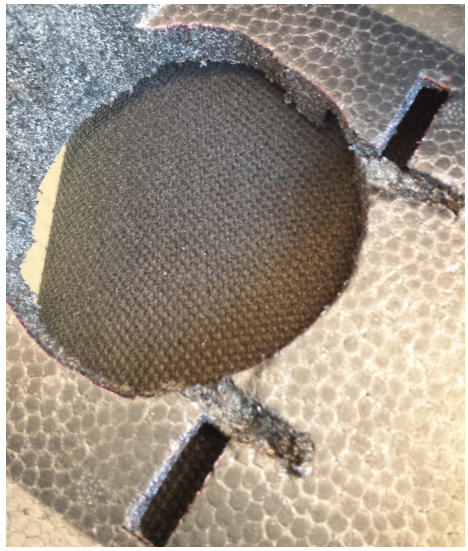
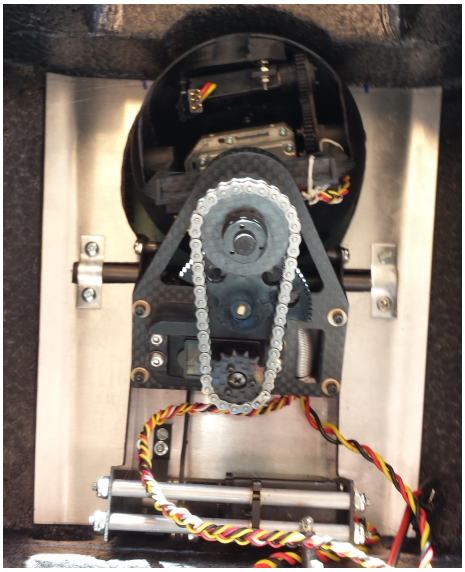
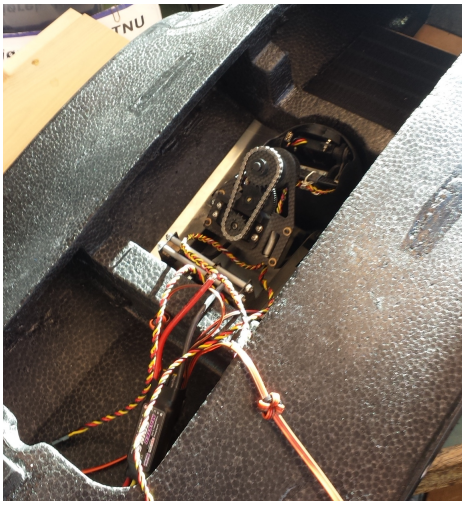


Figure 6.6: Hole and pivot rod holes bored into the X8 frame.

The entire retractor mechanism with the gimbal attached was mounted on a 1mm aluminium plate, which was cut and bent into position to fit in the X8's center bay, and to reinforced the structure due to the parts of the airframe which were cut away. The completed mounted system is show in Figure 6.7.



(a) Mounting of gimbal and retractor.

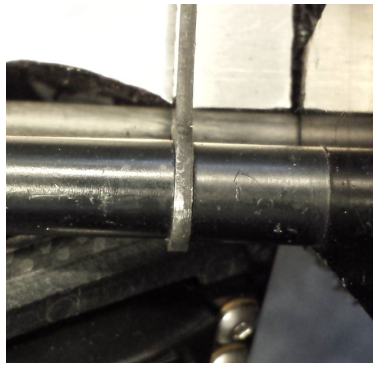


(b) Wider shot of payload bay.

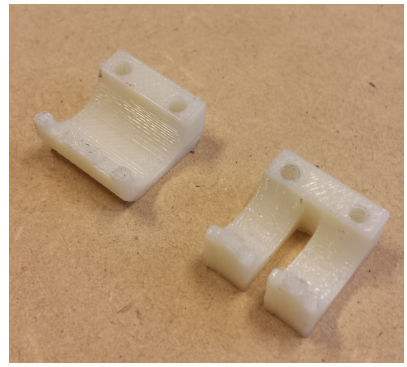
Figure 6.7: Gimbal and retractor mounted into X8 frame.

Unfortunately, the retractor mechanism suffered from wobble due to excessive play in the retractor arm. However, since the aluminium

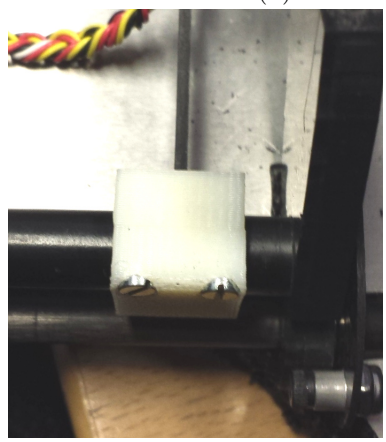
plate was already glued to the airframe, the offending arm could not be replaced, so an adaptor was created in Solidworks[8]. The part was 3d-printed, which was deemed sufficiently strong to hold it in place, which replaced an aluminium part milled by Furseth. The part drawings can be obtained by request. The 3d printed fix is shown in Figure 6.8.



(a) The offending retractor arm.



(b) The 3d-printed part.



(c) The part is mounted.

Figure 6.8: With the part mounted, gimbal wobble is greatly reduced.

[8]CAD modeling software.

6.6 Camera calibration

When calibrating a regular daylight camera, most algorithms require the use of either a chessboard pattern or a square circles grid-pattern printed on a sheet of paper, with known spacing. The algorithm can then detect the corners of the chessboard or the circles, and use the information to estimate the radial distortion parameters mentioned in Chapter 2.4.2. This is however difficult with a thermal camera, as any patterns printed on paper will not be visible in the thermal spectrum. Therefore, a plate was cut out of 5 millimetre thick MDF[9] plate, into which a pattern was drilled, consisting of a 12-by-12 grid of circles. This plate is shown in Figure 6.9. The circles were spaced 20 millimetres apart. This plate in turn was placed in front of a heat source, creating a grid pattern visible in the thermal spectrum. After setting the camera output to use a black-hot colour palette, the OpenCV camera calibration algorithm was able to detect the grid pattern and estimate radial distortion coefficients. The estimated coefficients, along with the camera matrix from Equation (2.28), are shown in Table 6.1, and Equation (6.1) respectively.

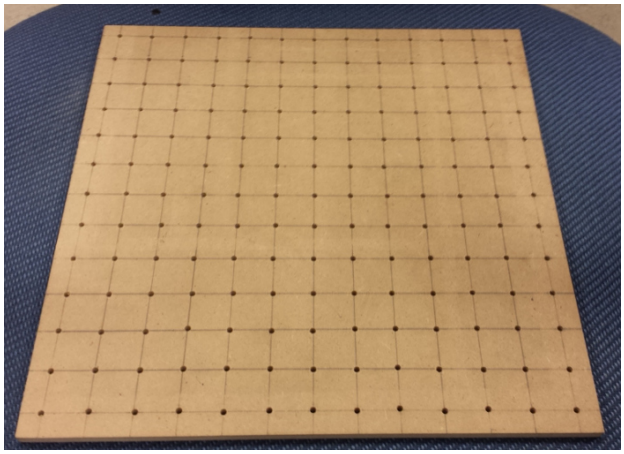


Figure 6.9: Calibration plate used for calibrating a thermal lens.

[9]Medium Density Fibreboard.

Table 6.1: Coefficients calculated by OpenCV camera calibration algorithm

Coefficients	**Value**
k_1	-8.25
k_2	$-5.16 \cdot 10^2$
k_3	$-1.27 \cdot 10^{-2}$
k_4	$-5.50 \cdot 10^{-2}$
k_5	-8.72

$$\boldsymbol{A} = \begin{bmatrix} 4734 & 0 & 369 \\ 0 & 4734 & 261 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.1}$$

As we can see, the estimates for the camera intrinsic parameters differs from the pre-calculated matrix in Equation (2.28) by a rather large factor. This discrepancy will be discussed in the discussion chapter.

6.7 Summary

In this chapter, the construction of the payload created for future flight tests is documented, along with a method for calibrating camera matrix parameters for our thermal camera.

Chapter 7

Flight tests

A flight test of the Skywalker X8, along with the payload made in collaboration with Leira, was originally scheduled for mid-to-late January, well within the time scope of this thesis. Due to circumstances beyond our control, the flight tests have been rescheduled past the due date of this thesis, therefore the results of the flight, as well as performance of the real system can not be summarized here. The results of flight testing will be summarized later in the ICUAS'15 paper.

Chapter 8

Discussion

8.1 Simulation results

8.1.1 Performance of observers

As we can see from the simulations, all three observers (linear, nonlinear, nonlinear with yaw feedback) manage to track the movement of the ship. It is however visible, especially in the noiseless cases, that the state estimator lags the actual states with some degree. It is especially evident in the initial speed estimates, where a large error spike is seen. This is due to slow convergence of the error covariance estimate, which in turn is due to poor tuning of the filter. This will be discussed in the next section.

8.1.2 Tuning the Kalman filter for better performance

The Kalman filter design matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ allow the filter to be tuned for better performance. If the diagonal values for $\boldsymbol{R}$ are set high, the filter interprets this as the measurements being unreliable, and will therefore give less weight to these measurements. If they are set to be low, the filter will give more weight to the measurements.

The $\boldsymbol{Q}$ matrix on the other hand, can be thought of as assigning more or less "merit" to the internally estimated states of the system. It is obvious here that in most cases, direct knowledge of the type and variance of noise encountered will not be known, and some form of tuning of these matrices will be necessary. Different methods of tuning the filter have been proposed throughout the years [20][21], and using some of these methods would most likely yield much better results that choosing values for $\boldsymbol{Q}$ and $\boldsymbol{R}$ by trial-and-error.

8.2 Performance of observer with measurement data loss

The nonlinear observer with yaw feedback does a rather good job of tracking the ship motion, however, it is interesting to see how the observer behaves when the measurements are lost for certain times, such as is the case when the aircraft loses the object from sight, or fails to classify it. A simple simulation was done where the measurements disappear for certain times:

- From $t = 10$s to $t = 12$s.
- From $t = 18$s to $t = 25$s.
- From $t = 38$s to $t = 41$s.

This scenario is only tested for the observer which performed best, which was the nonlinear observer with yaw feedback. The result is shown in Figure 8.1. As we can see, the observer performs rather well despite large gaps in the measurement data. In the first gap, we can see that the system has yet to observe any change in yaw rate, and therefore when measurements disappear, the tracking continues on the wrong course. However, during the second dropout, we can see that the filter overestimates the yaw change, since it does not receive input that the ship has stopped changing path. In the last dropout, we can see that the filter does a rather good job of tracking

the curvature of the path, and little correction is needed when the measurements restart. It is obvious that given a long enough outage, the filters estimate of the ship's position will become worse. If for instance the measurements did not reappear at time $t = 41$s, the filter would estimate that the ship is spinning in circles. This can be countered by adding a restoring force on the estimated yaw rate r,

$$\dot{r} = -cr \tag{8.1}$$

where c is a constant which can be tuned. The estimated position would then assume that if the measurement is lost, the ship is more likely to have stopped turning in the time where the measurement was unavailable. However, both assumptions are equally valid, the ship could both have stopped turning and continued in circles, therefore such a modification should only be done when more information about the system model being tracked is known.

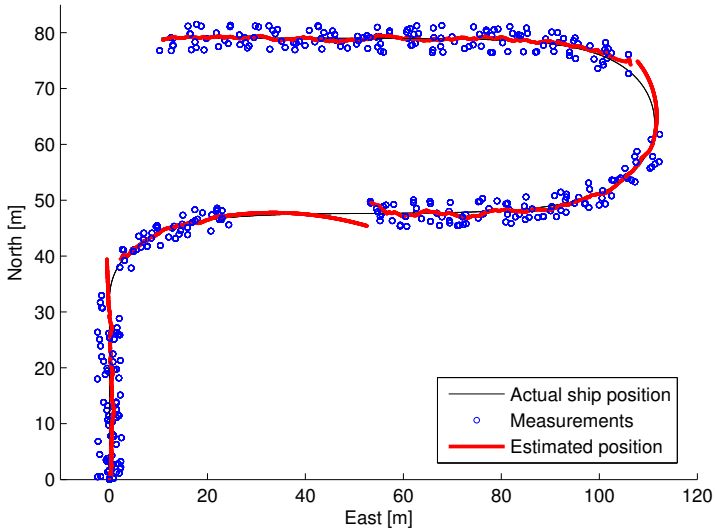


Figure 8.1: Simulation of nonlinear observer with yaw feedback, experiencing measurement outages.

8.2.1 High frequency noise in Kalman filter estimate

While simulating the observers without noise, it was noted in Chapter 5.2.1 that a high-frequency oscillation occurred in the Kalman filter estimates. This issue likely stems from the numerical nature of the filter implementation, introducing roundoff errors [6, p. 260], such as lack of symmetrization of the error covariance matrix $\boldsymbol{P}$. Several different ways of solving the issue have been discussed, however, in our case an interesting point is made about U-D factorization [6, p. 368]. U-D factorization starts with factorizing the $\boldsymbol{P}$ matrix into an upper triangular and diagonal part:

$$\boldsymbol{P} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^\top \tag{8.2}$$

The $\boldsymbol{U}$ and $\boldsymbol{D}$ matrices are then propagated by themselves, and later recombined into the $\boldsymbol{P}$ matrix. This helps reduce many of the issues encountered with numerical roundoff [6, p. 370], and would likely solve the issue which was encountered.

8.2.2 Linear observer performance and usage

The performance of the simple linear observer was surprisingly accurate in estimating the position of the ship, but did however have trouble estimating the correct speed. Such a linear observer is better suited for tracking icebergs, where a linear motion model makes more sense, as the iceberg can not have any self-induced motion, and its direction of travel is not dependent on its yaw orientation, which is mostly governed by difficult to model water and wind-forces. The linear model also makes sense if there is no knowledge of the underlying system generating the motion.

8.3 Classification

One of the main contributions of the classification system is the ability to recognize an object after it has left the camera's field of view. If there are many different objects, and each object has an observer estimating its position, it is important that the next time the object in question returns into view, its measurements are input to the correct observer. Otherwise, the observer will start using the new measurements as belonging to the wrong object. Methods for increasing the correct classification percentage is discussed as an item under future work.

Chapter 9

Conclusion and future work

As we can see, the simulations show that the observer theory developed seems to work well for tracking ships and other objects. Unfortunately, it is hard to draw a solid conclusion at the current time. However, the rescheduled flight tests will be summarized in the paper submitted to ICUAS'15 [14], and allow us to better test the performance of the observers, synchronization method, object detection and classification on real-life data.

9.1 Future work

The system presented in this thesis consists of several different components which need to function together in order to provide accurate tracking of objects of interest. The detection and classification subsystems by Leira [13] provide necessary data for the observers and synchronized georeferencing system developed in this thesis to properly estimate position of on-ground objects. The largest challenge ahead lies in making sure all modules are synchronized in order for the system to work properly, and the on-board computer needs have enough processing power to be able to run all the different algorithms. Hardware-in-Loop (HIL) testing of the system is crucial before flight tests are performed. It should also be tested whether the

synchronization procedure developed in Chapter 2.5.1 could work in real-time. Additionally, creating a simple human-in-the-loop system for administration of detection object should be looked into.

9.1.1 Observer robustness

Implementation of the U-D factorization technique mentioned in Chapter 8.2.1 should be done for future work, to make the nonlinear observer more robust against divergence and oscillations.

9.1.2 Daylight camera

An interesting addition to the classification system would be data received from a daylight (visible light) camera. Many detectable features are lost when viewing an object in the thermal range, which could be useful for cases where the classifier has difficulty determining the type, and the object instance.

9.1.3 FLIR Camera interface

The FLIR Tau 2 camera has several different modes of operation, and these settings may be set by communicating with the camera over an RS232 serial connection. This would allow us to change the different parameters of the camera, which could aid the object detection and classification process. An important future development would be to directly communicate with the camera, allowing for much more control of the detection process.

9.1.4 Camera calibration

In Chapter 6.6, it was noted that the estimated camera interior parameter matrix was far off from the pre-calculated matrix. This indicates that there were either issues with the calibration procedure itself, or the calibration tool was improperly initialized. This is looked into in the ICUAS'15 paper [14].

Bibliography

[1] Thor I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control*. Norwegian University of Science and Technology, Trondheim, Norway, 2011.

[2] Thor I. Fossen, *Mathematical Models for Control of Aircraft And Satellites*. Center for Autonomous Marine Operations and Systems, Department of Engineering Cybernetics, NTNU. 3rd edition, April 2013.

[3] Haitao Xiang, Lei Tian, *Method for automatic georeferencing aerial remote sensing (RS) images from an unmanned aerial vehicle (UAV) platform*. Monsanto Company, USA. Department of Agricultural and Biological Engineering, University of Illinois at Urbana-Champaign, USA. 22 December 2010.

[4] Lili Ma, YangQuan Chen, Kevin L. Moore, *Rational Radial Distortion Models of Camera Lenses with Analytical Solution for Distortion Correction*, Utah State University, 10 April, 2004.

[5] R. E. Kalman, *A New Approach to Linear Filtering And Prediction Problems*. ASME Journal of Basic Engineering, 1960.

[6] Robert Grover Brown, Patrick Y.C. Hwang, *Introduc-*

tion To Random Signals and Applied Kalman Filtering. Wiley, New York NY, Third Edition, 1997.

[7] Jens G. Balchen, Trond Andresen, Bjarne A. Foss, *Reguleringsteknikk*, Institutt for teknisk kybernetikk, NTNU, 5th edition, januar 2003.

[8] Richard O. Duda, Peter E. Hart, David G. Stork, *Pattern Classification*. Wiley, New York NY, Second Edition, 2001.

[9] Sonka, Milan and Hlavac, Vaclav and Boyle, Roger, *Image processing, analysis, and machine vision*. Cengage Learning, Third Edition, 2008.

[10] Stevens, B. L. and F. L. Lewis, *Aircraft Control and Simulation*. John Wiley & Sons Ltd., 2003.

[11] Robert Hermann and Arthur J. Krener *Nonlinear Controllability and Observability*. IEEE, 1977.

[12] J. K. Hedrick and A. Girard, *Control of Nonlinear Dynamic Systems: Theory and Applications*. 2005.

[13] Frederik S. Leira, Tor Arne Johansen, Thor I. Fossen, *Automatic Detection, Classification and Tracking of Objects in the Ocean Surface from UAVs Using a Thermal Camera*. Centre for Autonomous Marine Operations and Systems, Department of Engineering Cybernetics, Norwegian University of Science and Technology. 2015.

[14] Frederik S. Leira, Kenan Trnka, Thor I. Fossen, Tor Arne Johansen, *A Light Weight Camera Based Payload with Georeferencing Capabilities for Small Fixed Wing UAVs*, Centre for Autonomous Marine Operations and

Systems, Department of Engineering Cybernetics, Norwegian University of Science and Technology. 2015.

[15] Elder M. Hemerly, *Automatic Georeferencing of Images Acquired by UAV's*. Systems and Control Department, Technological Institute of Aeronautics, S ao Jos e dos Campos-S ao Paulo, Brazil. August 2014.

[16] Bruce D. Lucas, Takeo Kanade, *An Iterative Image Registration Technique with an Application to Stereo Vision*. Computer Science Department, Carnegie-Mellon University, Pittsburgh, Pennsylvania. 1981.

[17] Berthold K. Horn and Brian G. Schunck, *Determining Optical Flow*. Proc. SPIE 0281, Techniques and Applications of Image Understanding, 319 (November 12, 1981).

[18] Wei Kang, Arthur J. Krener, Mingqing Xiao, Liang Xu, *A Survey of Observers for Nonlinear Dynamical Systems*. NRL, AFOSR. 2013.

[19] Michael Teutsch and Wolfgang Krüger, *Classification of small boats in infrared images for maritime surveillance*. Proceedings of 2nd NURC International Water-Side Security Conference, Marina di Carrara, Italy, November 2010.

[20] Bernt M. Åkesson, John Bagterp Jørgensen, Niels Kjølstad Poulsen, Sten Bay Jørgensen, *A generalized autocovariance least-squares method for Kalman filter tuning*. Technical University of Denmark, 2007.

[21] Silverio Bolognani, Luca Tubiana, and Mauro Zigliotto, *Extended Kalman Filter Tuning in Sensorless PMSM Drives*. IEEE Transactions on Industry Applications, vol. 39, no. 6. November 2003.