Stabilization of a thermal camera at sea

Martin Stanic

LUND UNIVERSITY

Department of Automatic Control

MSc Thesis
ISSN 0280-5316
TFRT--6013

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Abstract

During, for example, search-and-rescue operations at sea, technical equipment like spotlights and thermal cameras are important aids. However, heave and sway affect the ship and make it harder for finding a person in distress.

This thesis presents a way of stabilizing a thermal camera by controlling a stepper motor connected to it. Moreover, the thermal camera can only be turned upwards and downwards.

The whole process was investigated to see what can and needs to be measured for stabilization at sea to work. With this knowledge, sensors had to be chosen to collect the necessary measurements.

The measurements from the different sensors were merged together using a Kalman filter to give an estimation for the tilt and elevation of a ship, which was used for controlling the stepper motor. Moreover, the control of the stepper motor was done using PID control.

The process was simulated in Simulink, making it possible to tune different parameters so that good performance was ensured based on assumed models. This was then implemented on the real system and tests were carried out to verify what was found during simulations. The result from this thesis is a stabilized thermal camera which helps an operator enormously in searching and finding objects at sea.

Acknowledgements

I would first like to thank Johan Karlsson at AES Nordic for making it possible to start this project together with Colorlight. Furthermore, I would also like to thank my supervisor at the company, Henrik Johansson, for his invaluable assistance and for sharing thoughts and ideas which helped in solving many different problems.

I would also like to thank Henrik Berg and Niclas Pettersson at AES Nordic for their help in programming, especially with ideas on what, why and how to use different approaches, as well as explaining the theory behind the ideas.

Many thanks also to my supervisor, Dr. Anders Robertsson, at the Dept. of Automatic Control, LTH, for his continuous help, feedback and ideas to keep the project moving forward.

Finally, I would also like to thank Rebecca Pue for her immense support and for being there when I needed to discuss ideas.

Abbreviations and symbols

ADC	Analog-to-Digital Converter
CAN	Controlled Area Network
DLPF	Digital Low-Pass Filter
I^2C	Inter-Integrated Circuit
IMU	Inertial Measurement Unit
MCU	MicroController Unit
MEMS	MicroElectroMechanical Systems
PID	Proportional-Integral-Derivative
RAO	Response Amplitude Operator
UART	Universal Asynchronous Receiver/Transmitter
${}^{S}a$	Acceleration measurement in sensor frame
${}^{E}a$	Accelerometer measurement in Earth frame
$\omega_x, \omega_y, \omega_z$	Angular velocity measured around x, y and z axis respectively
a_x, a_y, a_z	Acceleration measured around x, y and z respectively
ϕ	Rotation around x-axis
θ	Rotation around y-axis
ψ	Rotation around z-axis
α	Tilt of the thermal camera set by operator
β	Tilt of the ship
γ	Compensated tilt needed for the thermal camera
Δt	Sampling time

Contents

1

Introduction

A searchlight on a ship is traditionally a big lamp which is mostly used to illuminate an area to find or avoid objects, such as humans or icebergs, which can be hard to spot visually by looking out at the surface of the sea. Some modern searchlights, such as certain models made by Colorlight, come with an optional thermal camera. This makes the searchlight more versatile and improves the possibility of finding objects at sea, especially at night. A thermal camera is especially good at sea because of the homogeneous surface temperature, making it easy to spot objects with a different temperature. Interestingly, one can spot a human at sea at a distance of up to 800m by using the thermal camera provided by Colorlight [*Searchlight system by Colorlight* 2014].

The way an area is scanned at sea by an operator differs between an ordinary searchlights and a thermal camera. Although both are operated from the wheelhouse, traditional light can be observed through the windows of the wheelhouse whereas a screen is needed for the thermal camera. Using a screen has the disadvantage that the thermal camera is sensitive to disturbances such as vibrations. The image also follows the motion of the ship, which can cause seasickness. Furthermore, unless compensated for, it does not need much motion from the ship to cause an object on the screen to disappear.

1.1 Background

Colorlight is a company that produces searchlights for ships, and the searchlights can be equipped with an optional thermal camera. As of today, everything is controlled manually by an operator using a joystick and a control panel. While the searchlight can be turned both horizontally and vertically, the thermal camera itself can only be controlled vertically. Manual control is not an issue with the traditional searchlight since the illuminated area is large enough to handle various motions of a ship. The thermal camera, however, is sensitive to a ship's motions since objects moves around much more on the screen.

One way of solving this problem is by controlling the stepper motor that is connected to the thermal camera. Sensors need to be brought in so one can compute and compensate for the motion of the ship, thus stabilizing the thermal camera in a given direction. The greatest challenge with using onboard sensors is that there is no external sensor correcting the internal sensors, which will make them sensitive to drift.

1.2 Aim of this thesis

The primary aim of this thesis is to investigate if it is possible to control the stepper motor to minimize disturbances. The goal is to keep the image stable from the thermal camera in the direction set by the operator. This is critical both when searching for an object and when an object is found and should be kept on the screen.

The secondary aim of this thesis is to see if it is possible to do absolute tracking of an object on the screen by using image processing. The goal is to keep an object centered on the screen.

1.3 Problem formulation and objectives

To be able to fulfill the aims of this thesis, one has to investigate and evaluate feasible equipment, such as sensors, that can be used to achieve the goals. If different sensors are used, then it needs to be investigated if and what kind of sensor fusion that is needed, so that disturbances and inaccuracies are minimized.

The searchlight, in which the stepper motor and thermal camera is mounted, is encapsulated to keep water out. However, the temperature inside can get very high due to the stepper motors. Generally, an increased current to the stepper motor increases the heat generation, and the temperature can get dangerously high which in the worst case scenario could potentially damage electronics inside. Because of this, it is of great importance to minimize the amount of time the stepper motor is fed a high current, but not so much that the stepper motor fails to stabilize the thermal camera.

Finally, everything developed in this thesis has to be implemented on top of the original framework, both hardware and software, and needs to work within the time constraints and other limitations.

1.4 Demarcation

The circuit board available in this thesis, referred to as the secondary circuit board, is used for controlling the thermal camera and stepper motor and is separated from the primary circuit board. The primary circuit board is used for handling the joystick and control panel used by an operator and sends instructions to the secondary circuit board. Unfortunately, the primary circuit board is not available for experiments, thus instructions sent from primary to secondary cannot be handled in this thesis. The control of the stepper motor does not need to handle changes from the operator, e.g., a tilt change of the thermal camera.

External forces, such as waves, cause the ship to be set in motion, and the control of the stepper motor needs to cancel out the effect of this motion. However, the control does not need to perform better than the conditions in which humans can operate. While a ship can tilt up to 60° before capsizing, it is hard to work properly if the tilt exceeds 20° [Pawlowski, 2010]. Therefore, the control does not need to handle more than the latter.

2

Material and system overview

This chapter includes discussion about materials used in this thesis, both hardware and software.

2.1 Hardware

Hardware listed in this section is material which has been actively worked with to accomplish the results. Only technical data about each unit is described here, whereas details and theory is described in Chapter 3.

Colorlight searchlight

Figure 2.1 shows one of the different types of searchlight currently manufactured by Colorlight. The thermal camera is mounted inside the camera house at the top of the module and is connected to a stepper motor as in Figure 2.2.

Thermal camera

The thermal camera, seen in Figure 2.3, is a ***FLIR Tau 2*** and can detect a man/ship at 800m/2000m, recognize a man/ship at 200m/550m and identify a man/ship at 100m/300m.The different categories for detection, recognition and identification is given by Johnson's criteria which simply describes the probability of detecting/recognizing/identifying an object at a certain distance [*Searchlight system by Colorlight* 2014] [CohuHD, 2014].

ARM Cortex-M3

The MicroController Unit (MCU) is an ***NXP LPC1754 ARM Cortex-M3*** which can operate at a CPU frequency up to 100MHz and includes 512kB flash memory together with 64kB data memory. Furthermore, it has four Universal Asynchronous

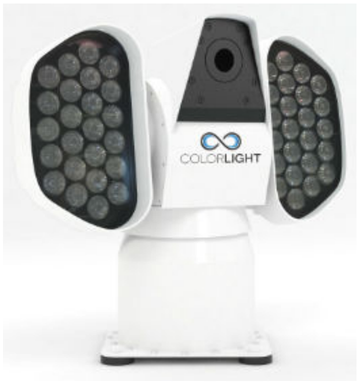

Figure 2.1 A searchlight made by Colorlight. This one is using LEDs. The thermal camera is mounted inside the black camerahouse mounted on top [*CLITE2*].

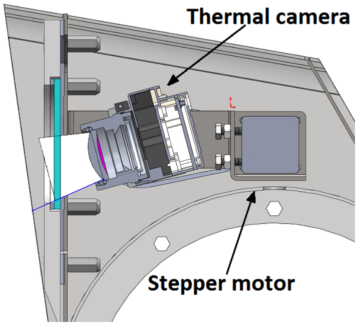

Figure 2.2 The stepper motor, on the right side, is connected to the thermal camera, on the left side, using a belt.

Figure 2.3 FLIR Tau 2, a thermal camera with a 25mm lens [*Tau 2*].

Receiver/Transmitter (UART), two Controlled Area Network (CAN) channels and two Inter-Integrated Circuit (I^2C) bus interfaces [*LPC1759/58/56/54/52/51* 2015]. In the original work, one UART, one CAN-channel and one I^2C-bus was already in use.

Stepper motor

The stepper motor, ***28SH32-0674A***, is the actuator for controlling the tilt of the thermal camera. A motor driver is needed to translate instructions from the MCU into corresponding signals to the stepper motor, see Figure 2.5, and include instructions such as direction and step time. Step time is the time between two steps, and the fastest possible rotation speed is 66.7°/s with a step time of 1ms. A stepper motor has both advantages and disadvantages, as described below.

Advantages

1. The rotation speed is proportional to the frequency of the input pulse.
2. The stepper motor has full torque at standstill if the windings are energized.
3. The positioning error for each step is only 3-5% and does not accumulate.
4. It has excellent response to starting, stopping and reversing.
5. The stepper motor does not need any encoder to know the current position. The position is known simply by tracking input step pulses.

Disadvantages

1. Resonance may occur if not properly controlled.
2. It is not easy to operate at extremely high speeds and may lose steps.
3. If the stepper motor misses a step the positioning will be wrong, and without an encoder there is no way of tracking this error [*A4982 Motor Driver* 2014].

The stepper motor is connected to the thermal camera as in Figure 2.2. The camera can be tilted a total of 40°, defined as ±20° above and below the horizontal axis [*Searchlight system by Colorlight* 2014]. In this interval, the stepper motor is capable of taking 600 steps, or 0.067°/step. When the searchlight is started, the stepper motor rotates clockwise for a few seconds to ensure that the tilt is -20°. This to ensure that the position is known at start.

The stepper motor performs in two modes, *running* and *standby*. When the stepper motor has to take steps the mode is set to *running*, which will feed a higher current to the stepper motor. Once the desired steps have been taken, the mode is set to *standby*. In this mode, the current is lowered so that adequate holding torque is achieved to withstand vibrations and accelerations while keeping the heat generation to a minimum, thus reducing the risk of overheating.

Figure 2.4 The stepper motor which is used for controlling the tilt of the thermal camera [*28SH32-0674A*].

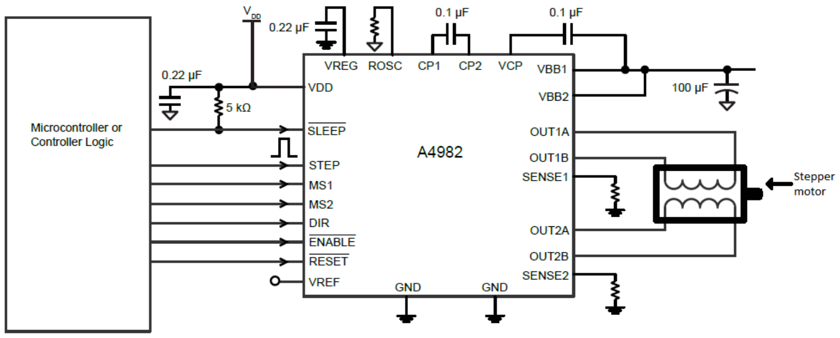

Figure 2.5 The microcontroller sends command pulses to the motor driver, A4982, which in turn translates this to corresponding signals to the stepper motor [*A4982 Motor Driver* 2014].

Inertial Measurement Unit

This Inertial Measurement Unit (IMU), ***MPU-9150*** is a microprocessor which combines three MicroElectroMechanical System (MEMS) sensors - a triple axes accelerometer, a triple axes gyroscope and a triple axes digital magnetometer. However, only the accelerometer and gyroscope were used due to the distance between IMU and stepper motor. While the stepper motor is running, the IMU will be under the influence of a magnetic field much greater than that from Earth, rendering the digital magnetometer useless.

Both the accelerometer and gyroscope use a 16-bit Analog-to-Digital Converter (ADC) for each axis, and the measurements are represented using two's-complement. Thus, measurements can be represent in the interval [-2^{15}, 2^{15}-1], or [-32768, 32767] [Brorsson, 2011]. Furthermore, each sensor has a full-scale range that can be set, where a lower range means higher resolution. The accelerometer can be set to $\pm 2g$, $\pm 4g$,$\pm 8g$ and $\pm 16g$, while the gyroscope can be set to $\pm 250°$/s,

±500°/s, ±1000°/s and ±2000°/s. Thanks to the dedicated microprocessor, this IMU can collect and save bytes of data continuously to registers, as well as sending and receiving data from the MCU using I^2C at 400kHz. Moreover, the IMU also supports an auxiliary sensor, such as a pressure sensor [*MPU-9150* 2012].

Pressure sensor

This digital pressure sensor, ***Bosch BMP085***, was connected as an auxiliary sensor to the IMU and has its own address which enables communication from MCU using I^2C. Apart from measuring pressure, it also measures temperature which can be used for a better altitude estimation. It comes factory calibrated with fixed calibration coefficients that are used when computing altitude. Pressure is measured as hPa and has a range of [300, 1100], corresponding to the height interval [9000, -500] above sea level in meters.

The noise in the measurements, range from 0.06hPa (0.5m) down to 0.03hPa (0.25m), depends on what mode the sensor is running in. Higher accuracy corresponds to a lower sample rate.

Unlike the IMU, this sensor does not have a dedicated microprocessor which can collect data continuously. Instead, collection of data has to be started from the MCU every time it is needed [*BMP180* 2013].

2.2 System architecture

A system overview on how the hardware is connected can be seen in Figure 2.6. A laptop was used and connected to the circuit board using a JTAG-debugger, which was used both for transferring code to the MCU and debugging using Eclipse [*Eclipse*].

Communication to the laptop was done using a Future Technology Devices International (FTDI) cable through UART. The FTDI-cable was used to convert signals from RS-232 to Universal Serial Bus (USB) signal [*FTDI* 2010].

The communication from the MCU to laptop was used for troubleshooting by displaying outputs on the screen of a laptop using the terminal program *Putty* [*PuTTY FAQ*]. Putty could also log the outputs in a text document. The logged outputs could later be used for analysis in MATLAB. Caution had to be taken when sending data from the MCU to laptop, since only integer values are supported.

The MCU, on the other hand, was connected and in control of the thermal camera, the stepper motor and all the sensors, where only the sensors are sending data back to the MCU. Furthermore, the MCU was fitted with a real time operating system called ***FreeRTOS***. This operating system supports the use of threads, protection of variables using methods such as critical section and mutex, and has predefined methods to ensure a fixed sample period [freeRTOS, 2016].

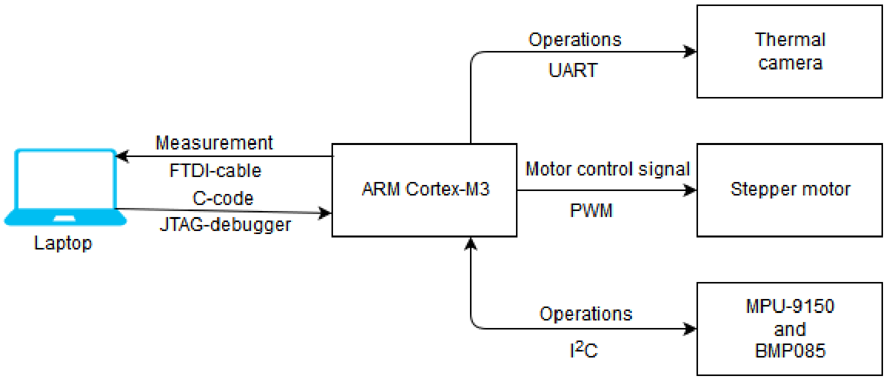

Figure 2.6 Control overview. Laptop communicates with the MCU, which in turn communicates and controls the thermal camera, stepper motor and the sensors.

3

Theory

In this chapter the theory will be presented about different parts such as waves, ships and sensors. Furthermore, any assumptions and simplifications made will be explained as well as how and why they are used.

3.1 Wave theory

When studying waves, two cases are generally examined - *shallow waters* and *deep waters*. Deep water applies if the depth is equal or larger than half of the wavelength of the current waves. Shallow waters typically occurs in lakes, rivers or near a coast [Perez, 2005].

Because ships mostly travel at deep waters, this was the only case that was used in this thesis. Because of this, there are a couple of assumptions and simplifications that can be made.

One type of waves, *regular waves*, is defined as a harmonic wave that is moving along a surface as in Figure 3.1. The equation for such a wave is given by Equation 3.1. The *elevation from equilibrium*, $\zeta(x,t)$, depends on the *wave amplitude*, A, which is half of the *wave height*, H. Furthermore, λ is the *wave length* and ω is the *circular wave frequency*, which depends on the *wave period time* $\omega = 2\pi/T$. Moreover, the crest of the wave is moving with a speed defined as $c = \lambda/T$ and is called *wave celerity*.

From [Perez, 2005] it is shown that one can take the *encounter angle*, i.e., the angle of which the waves are moving along a ship, as well as the forward motion of the ship into consideration. However, when modeling and evaluating the motion of the ship due to waves, only the elevation itself is important. Because of this, it was assumed during simulations that the ship moves only along the vertical axis with the waves and is stationary along the surface, i.e., no forward motion. Due to this, x in Equation 3.1 can be set to zero.

By combining the wave period time and wave celerity and solving for ω, one gets that $\omega = \frac{2\pi c}{\lambda}$. When substituted in Equation 3.1, as well as assuming that a ship is not in motion along the surface, the simplified equation is defined in Equation 3.2.

$$\zeta(x,t) = Asin(\omega t - kx) \tag{3.1}$$

$$\zeta(0,t) = Asin(\frac{2\pi c}{\lambda} t) \tag{3.2}$$

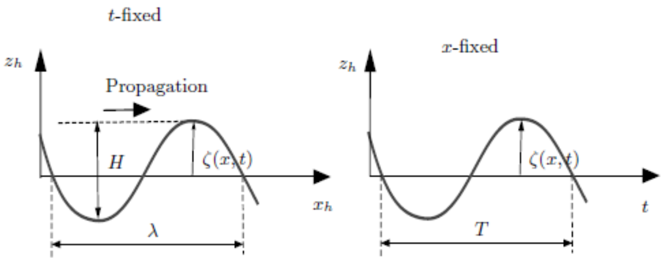

Figure 3.1 A wave can either be described by its propagation (left picture) or by the elevation in a fixed point (right picture) [Perez, 2005].

Equation 3.2 therefore depends on three constants: wave amplitude, celerity length and wave length. Moreover, each constant is limited in one way or another and many assumptions can be made since only deep water is considered. The wave length is limited by $h \geq \lambda/2$, where h is the depth of the sea. Furthermore, using the *dispersion relation*, i.e., $\lambda = \frac{g}{2\pi}T^2$, one can see that the wave length depends on the wave period time T [Perez, 2005]. By substituting T with the wave celerity and solving for c, one gets Equation 3.3.

Finally, according to [Acheson, 1990], only linear terms are considered. Thus, for a wave to exist and not fall apart due to its steepness, the relation $A/\lambda \ll 1$ must hold.

$$c = \sqrt{\frac{g\lambda}{2\pi}} \tag{3.3}$$

A way of finding a reasonable wave amplitude is to use the *significant wave height, H_s*. In the time domain, H_s is defined as the average wave height of the largest one-third of the measured waves as in Equation 3.4. H_i is the individual wave height of each wave measured, sorted in such a way that the first term is the highest and the last term is the lowest. Another way of determining the wave amplitude is to look at the largest one-tenth of the measured waves, called $H_1/10$, which is approximately 1.27 times larger than H_s. This can be used during simulation to see how well the control handles rare waves with a very large amplitude. Generally, H_s for moderate waves is 1.25-2.5m, high is 6-9m and very high is 9-14m [Ainsworth, 2006][Perez, 2005].

$$H_s = \frac{1}{\frac{1}{3}N} \sum_{i=1}^{\frac{1}{3}N} H_i, \tag{3.4}$$

All equations, limitations and assumptions above can be summarized as:

1. Only deep water was assumed.
2. A certain wave height, A, requires a specific wave length, given by the relation $A/\lambda \ll 1$.
3. With the computed λ, the wave celerity c is given in Equation 3.3.
4. The elevation at time t is calculated as $\zeta(0,t) = Asin(\frac{2\pi c}{\lambda}\ t)$.

3.2 Ship theory

In the field of ship design, the Response Amplitude Operator (RAO) is used to determine the likely behavior of a ship at sea based on a set of design parameters. However, the transfer function for RAO is only defined for linear motion [Allan, 1945]. As long as it is linear, the equation of motion is given as Equation 3.5, where x is a degree of freedom, ω is the *oscillation frequency*, M is the *structural mass*, *A(ω)* is the *inertia of added mass*, *B(ω)* is the *linear damping*, C is the *restoring coefficient* and *F(ω)* is the *harmonic excitation force* proportional to x and the wave height [Perez, 2005][Holden et al., 2007].

$$F(\omega) = Cx + B(\omega)\dot{x} + \big(M + A(\omega)\big)\ddot{x} \tag{3.5}$$

The variables in the equation above are highly dependant on the type of ship considered and the sea conditions. Although RAO can potentially be used to improve the response in the control loop or improve simulations, not enough time was put into developing this further.

The simulations in this thesis were performed using a general case, whereas the Colorlight searchlight is used on a great variety of ships ranging from work boats, such as coast guard or fishing boats, to mega yachts and commercial ferries. As a result, RAO was not used directly, but rather to give insight and understanding about the process. This could then be used to verify the simulation results. For example, RAO shows that a ship does not follow the wave motion exactly, but has a phase lag and a lower elevation compared to incoming waves [*CalQlata*].

According to [*GDV*], accelerations that occur on a ship depend on many parameters such as the shape of the ship, its beam, the center of gravity and the center of buoyancy. All motions on a ship can be divided into three linear motions and three rotational motions, see Figure 3.2. *Surge* is the motion alongside the ship, *sway* is the sideways motion, *heave* is the up and down motion while *roll*, *pitch* and *yaw* are the rotational motions along the respective linear motion. A ship's total motion on the sea strongly depends on the character of the waves which affect the ship. For example, perpendicular waves make the ship roll from side to side, while parallel waves alongside the ship will rotate the ship around pitch [Perez, 2005].

The characteristics of the rotational motion also differ. The pitching motion has a fairly low tilt, commonly less than 10°, and alternates between quick and slow rotation between crest and trough. Roll on the other hand can reach up to 50° in rare cases, but is commonly less than 30° during rough weather. This motion also has a long time period, commonly several seconds. In general, a larger tilt corresponds to a longer time period [*GDV*].

3.3 Coordinate systems

In this thesis, two types of coordinate systems have been used. The measurements from the sensors were given in *sensor frame*, while orientation was done using *Earth frame*, and both made use of the Cartesian right-hand coordinate system [Perez and Fossen, 2007]. Rotation from sensor to Earth frame was needed when computing both orientation and elevation of a ship.

Earth frame

The Earth frame coordinate system is shown as O_n in Figure 3.2 and is fixed against Earth as a local tangent plane. This system uses NED-orientation which gets its name from how the axes are pointed, with x_n pointed North, y_n pointed East and z_n pointed Down to the center of Earth [Fossen, 2002][Perez and Fossen, 2007].

Sensor frame

This frame, also referred to as *body frame*, is similar to Earth frame except that it is fixed against an object. In Figure 3.2, it is fixed against the ship and is denoted b_n. In

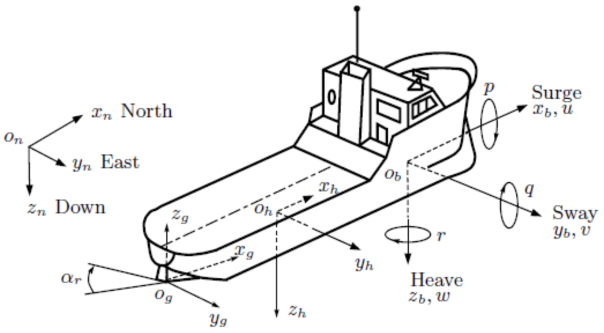

Figure 3.2 Picture showing different types of reference frames and motions [Perez and Fossen, 2007].

this thesis, however, the sensor frame is fixed against the sensors inside the searchlight. NED-orientation applies to this frame as well, where x_b is pointed towards the prow of the ship, y_b in the starboard direction and z_b is pointed downwards.

The aforementioned frames were used and compared against each other. The sensor measurements, given in sensor frame, were rotated into Earth frame to determine the tilt and elevation of the ship. To do this rotation, each rotation along x-, y- and z-axis, denoted as ϕ, θ and ψ respectively, was used. The rotation from one frame to another can be done using different methods, two of which are described below.

Euler angles

A rotation around an axis is defined by its own rotation matrix, see Equations 3.6 - 3.8, by using the Euler angles ϕ, θ and ψ. To get the resulting rotation, i.e., rotation from sensor to Earth frame, one has to multiply each rotation matrix together. However, care has to be taken when rotating using this methods since matrix multiplication is not commutative, that is in general, $A \cdot B \neq B \cdot A$ [Dam et al., 1998]. Therefore, after defining the order of rotation, here chosen as the *aerospace rotation sequence*, one has to maintain this order. The *aerospace rotation sequence*, defined as R_{xyz}, is shown in Equation 3.9. In the equation, $c(X)$ and $s(X)$ is short for cosine and sine respectively [Pedley, 2013][Diebel, 2006].

$$R_x(\phi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & sin(\phi) \\ 0 & -sin(\phi) & cos(\phi) \end{pmatrix} \tag{3.6}$$

$$R_y(\theta) = \begin{pmatrix} cos(\theta) & 0 & -sin(\theta) \\ 0 & 1 & 0 \\ sin(\theta) & 0 & cos(\theta) \end{pmatrix} \tag{3.7}$$

$$R_z(\psi) = \begin{pmatrix} cos(\psi) & sin(\psi) & 0 \\ -sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.8}$$

$$R_x(\psi)R_y(\theta)R_z(\phi) =$$

$$\begin{pmatrix} c(\theta)c(\psi) & c(\theta)s(\phi) & -s(\theta) \\ c(\psi)s(\theta)s(\phi) - c(\phi)s(\psi) & c(\phi)c(\psi) + s(\theta)s(\phi)s(\psi) & c(\theta)s(\phi) \\ c(\phi)c(\psi)s(\theta) & c(\phi)s(\theta)s(\psi) & c(\theta)c(\phi) \end{pmatrix} \tag{3.9}$$

The main advantage with using Euler angles for rotation is that it is quite easy to understand the mathematics behind it and it is possible to represent the angles in an easy way for users, which facilitates troubleshooting and visualization. Euler angles for rotation also have a lot of support and is used in a lot of applications. However, trigonometric functions require a fair amount of computations, thus making Equation 3.9 computationally expensive. The rotation matrix also suffers from gimbal lock, which occurs when one axis coincides with another axis, meaning one loses a degree of freedom in the rotation matrix [Dam et al., 1998].

Quaternion

Quaternions are not as well known as Euler angles, since they are not included in the standard curriculum in modern mathematics. This is a disadvantage since it causes a lack of understanding [Dam et al., 1998].

Quaternions try to generalize complex numbers in three dimensions. To describe a rotation, four variables are needed - one for scaling, one for the degree of rotation and two to describe the plane in which the vector should be rotated [Hamilton, 2000]. The reason for only using two variables to describe the plane is because a plane xy can be rotated to any plane in xyz space through the origin by the rotation about the x and y axes [Sparr, 1997]. Furthermore, the complex numbers can be written as $ix + jy + kz$ where $i^2 = j^2 = k^2 = ijk = -1$ and $x, y, z \in \mathbb{R}^3$ [Diebel, 2006]. Another way of describing quaternions is to define it as $\mathbf{Q} = q_0 + \mathbf{q}$, where q_0 is the scalar value mentioned above and $\mathbf{q}$ is a vector that describes the rotation and the plane.

An advantage with using quaternions is that it does not depend on the convention of rotation, making quaternions more fool-proof. It also requires storing less numbers than Euler angles when computing the rotation. Furthermore, quaternions are also more immune to accumulated computational error, which is good for the stability [Salamin, 1979].

In this thesis, quaternions were used in an algorithm developed by Madgwick in [Madgwick, 2010]. To represent an arbitrary orientation through rotation of angle θ from a frame B to frame A around an axis, ${}^{A}_{B}\hat{\mathbf{q}}$ will henceforth be used. The quaternion representing this orientation is defined in Equation 3.10, where r_x, r_y and r_z are the components of the unit vector ${}^{A}\hat{\mathbf{r}}$ in frame A. The quaternion conjugate, denoted by *, can be used to swap the relative frames, as in Equation 3.11. Furthermore, the quaternion product $\otimes$ is used for describing compound orientations, e.g., ${}^{A}_{C}\hat{\mathbf{q}} = {}^{A}_{B}\hat{\mathbf{q}} \otimes {}^{B}_{C}\hat{\mathbf{q}}$. As with matrix multiplication, quaternion multiplication is not commutive, i.e., $\boldsymbol{a} \otimes \boldsymbol{b} \neq \boldsymbol{b} \otimes \boldsymbol{a}$ [Sparr, 1997].

$${}^{A}_{B}\hat{\mathbf{q}} = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 \end{bmatrix} = \begin{bmatrix} \cos\frac{\theta}{2} & -r_x sin\frac{\theta}{2} & -r_y sin\frac{\theta}{2} & -r_z sin\frac{\theta}{2} \end{bmatrix} \tag{3.10}$$

$${}^{A}_{B}\hat{\mathbf{q}}^{*} = {}^{B}_{A}\hat{\mathbf{q}} = \begin{bmatrix} q_1 & -q_2 & -q_3 & -q_4 \end{bmatrix} \tag{3.11}$$

To rotate a three dimensional vector by quaternions, Equation 3.12 can be used. The vectors ${}^{A}\boldsymbol{v}$ and ${}^{B}\boldsymbol{v}$ are the same vectors described in frame A and frame B, respectively, where each vector contains a zero as the first element. This is done so the vectors are four element row vectors. Furthermore, the orientation described by ${}^{A}_{B}\hat{\mathbf{q}}$ can be represented as a rotation matrix ${}^{A}_{B}\mathbf{R}$, as in Equation 3.13 [Kuiper, 2002][Salamin, 1979]. Thus, by using the rotation matrix ${}^{A}_{B}\mathbf{R}$ one can easily rotate from one frame to another.

$${}^{B}\mathbf{v} = {}^{A}_{B}\hat{\mathbf{q}} \otimes {}^{A}\mathbf{v} \otimes {}^{A}_{B}\hat{\mathbf{q}}^{*} \tag{3.12}$$

$${}^{A}_{B}\boldsymbol{R} = \begin{bmatrix} q_1^2 + q_2^2 - q_3^2 - q_4^2 & 2(q_2q_3 - q_1q_4) & 2(q_2q_4 + q_1q_3) \\ 2(q_2q_3 + q_1q_4) & q_1^2 - q_2^2 + q_3^2 - q_4^2 & 2(q_3q_4 - q_1q_2) \\ 2(q_2q_4 - q_1q_3) & 2(q_3q_4 + q_1q_2) & q_1^2 - q_2^2 - q_3^2 + q_4^2 \end{bmatrix} \tag{3.13}$$

Quaternions can also be transformed back to Euler angles. This was used for troubleshooting, since Euler angles are much easier to interpret than quaternions. Since the aerospace sequence was used for describing the orientation from sensor frame to Earth frame, the conversion from quaternions to Euler angles had to be done in the same order. In Equation 3.15, the rotations ϕ, θ, ψ correspond to rotation around x-, y-, and z-axis [Madgwick, 2010]. The function atan2 is the four-quadrant inverse tangent, which describes a rotation in the closed interval $[-\pi, \pi]$, compared to the ordinary inverse tangent which only returns values within $[-\pi/2, \pi/2]$ [*MathWorks a*][*MathWorks b*].

$$\begin{aligned}\phi &= atan2(2q_3q_4 - 2q_1q_2, 2q_1^2 + 2q_4^2 - 1)\\ \theta &= -asin(2q_2q_4 + 2q_1q_3)\\ \psi &= atan2(2q_2q_3 - 2q_1q_4, 2q_1^2 + 2q_2^2 - 1)\end{aligned} \tag{3.15}$$

In summary, a rotation matrix of some kind was necessary so that sensor measurements could, for instance, be used for orientation. Both methods presented above were used in different parts throughout this thesis.

3.4 Sources of disturbance

To understand what is causing the image to the operator to deviate, one most understand the sources of disturbance. The sources which affect the image the most are presented in this section.

When the ship is at equilibrium, as in Figure 3.3, the thermal camera will be pointed at point $\boldsymbol{A}$ on the surface of the sea with tilt α set by the operator. If the ship were to be tilted by γ and the elevation stayed the same, the thermal camera would then be pointing at point $\boldsymbol{B}$ in Figure 3.3. In other words, the direction is deviated by γ.

Many disturbances, such as waves, are unpredictable while others, such as vibration caused by propeller and engine, are periodic [Jegaden, 2013a]. Furthermore, vibrations on a ship can be divided into different groups: *very low frequencies* ranging from 0-2Hz, *low frequencies* ranging from 2-20Hz and *high frequencies* ranging from 20-1000+Hz. Meanwhile, the accelerations are generally quite low, between 0.006-0.6m/s^2 along each axis depending on sea condition, wind direction and so forth. Moreover, according to [Asmussen et al., 2001], disturbances present on a ship are usually less than about 300Hz, whereas the maximum acceleration is generally less than 1g.

The swell causes random very low frequency vibrations on the whole ship, both in pitching and rolling, and are typically between 0.01-1.5Hz depending on sea conditions. Ship girder vibrations caused by sea conditions, on the contrary, are usually considered to be one of two types, *whipping* and *slamming*. Whipping usually occurs when the ship is traveling and the velocity is great enough to cause the front

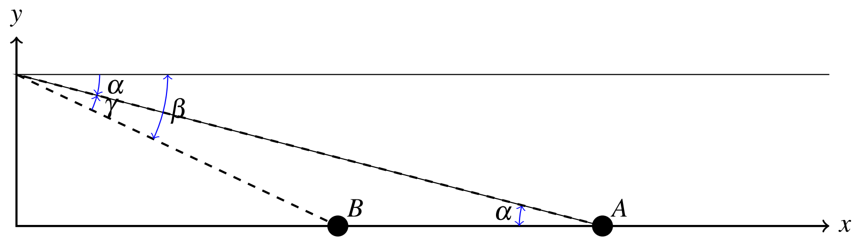

Figure 3.3 Illustration of a ship being tilted by γ while having the same height above water.

of the ship to impact against waves ahead, while *slamming* causes vibrations in the hull when the prow slams into a wave, making the ship suddenly stop in its motion.

There is also *slapping*, which occurs when there are impacts on the flat surface of the stem when the ship has not yet emerged from the water. And finally *springing*, which occurs by excitation caused by hydrodynamic forces created by the swell that causes a ship's girder to vibrate freely [Jegaden, 2013b].

The above vibrations, as small as they may be, will be induced not only in the image but also into the sensors which causes errors in the measurements. It is therefore important to take the aforementioned vibrations into consideration when estimating values such as the tilt of the ship. Although vibrations will cause some errors in the measurements, which will be seen as some deviation from point $\boldsymbol{A}$, it still is the elevation and tilt of a ship that is causing the greater part of deviation. The vibrations had to be considered when performing calculations using sensor measurements. The values found here, such as the range for frequencies and accelerations, are used in latter chapters when deciding the range of measurements for the sensors.

3.5 Model of system and error calculation

Using Figure 3.3, one can use trigonometry for a right-angled triangle to solve the deviation γ. For example, the deviation γ in Figure 3.3 is equal to $\gamma = \beta - \alpha$, where β is how much the ship is tilted from horizontal axis and α is the tilt set by the operator.

In the original work, α is the only known parameter. Solving for the unknown parameters, such as distance to point $\boldsymbol{A}$ or height of the thermal camera above water, in a right triangle requires knowing at least two parameters. One way of solving this is to set a fixed length on the hypotenuse and use this to solve for the unknowns. However, the estimated point will depend on the tilt α, where only one exact angle corresponds to the point being on the surface of the sea. A larger tilt will put the point below the surface, point ***B*** in Figure 3.4. On the contrary, a smaller tilt will put the point above the surface as point ***C***. This of course, is bad for the operator.

Instead, a far more superior way is to measure the height above water for the thermal camera when the searchlight is installed on a ship. The height above water can then be used for solving other parameters such as the distance to point ***A***, which can be computed as $distance = H \cdot tan(\alpha)$. Therefore, in the rest of this thesis it will be assumed that the height above water for the searchlight is known beforehand.

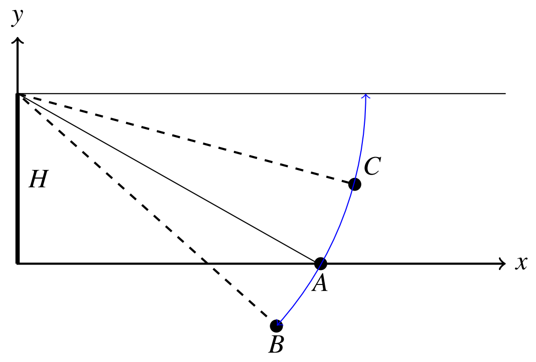

Figure 3.4 Estimation to a point on the surface of the sea using a fixed hypotenuse. Point ***A*** corresponds to the perfect value on α where the point is estimated on the surface. Point ***B*** and point ***C*** corresponds to tilt which is larger and smaller respectively.

3.6 Sensors

A total of three different sensors were used, each explained in its own section. A lot of time was put in this section of the thesis since the final control of the stepper motor relies heavily on the measurements and estimations done using the sensors. Therefore, the sensors were made to be as reliable as possible.

Inertia Measurement Unit - IMU

To get good and stable readings from the sensors it is important to remove as much measurement noise as possible, and the measurement noise is generally of high frequencies [Glad and Ljung, 2007]. Furthermore, as explained in Section 3.4, disturbances found on a ship is generally less than 300Hz. This gives a good indication of the frequencies which should be removed.

The IMU used in this thesis came with a Digital Low Pass Filter (DLPF) that was set to 44Hz, thus removing as much measurement noise and other disturbances as possible [*MPU-9150* 2012].

Each sensor, as mentioned in Section 2.1, has a full-scale range which determines the range of values that can be expressed. The range for each sensor was set high enough to ensure not saturating the measurements, and sufficiently low to get the highest possible resolution.

Accelerometer

In Section 3.4 it was shown that the accelerations on a ship are mostly less than 1g. Therefore, the full-scale range was set to the lowest possible, i.e., $\pm 2g$. With a 16-bit ADC and using two's-complement, the measurement are represented in the range [-32768, 32767] [Brorsson, 2011].

While stationary, an ideal accelerometer would only register the static acceleration. Since 2g is represented as 32767, 1g is simply half this value, i.e., 16384. The value is rounded, since only integer values are valid. However, accelerometers are not perfect and the measurements will be affected by errors which need to be compensated for. According to the the standard [IEEE, 2008], the complete error model is given by Equation 3.17 where

E/K_1 = accelerometer voltage output divided by scale factor K_1
a_i, a_p, a_o = applied accelerations along Input Axis (IA), Pendulous Axis (PA) and Output Axis (OA)
K_0 = the bias expressed in g
K_0' = the bias asymmetry in g
K_2 = non linearity expressed in g/g^2
K_3 = non linearity expressed in g/g^3
∂_o, ∂_p = IA misalignment with respect to OA and PA
K_{ip}, K_{io} = cross coupling coefficients expressed in g/g^2
K_{pp}, K_{oo} = cross coupling non linearity coefficients expressed in $g/cross - g^2$
K_{spin} = spin correction factor expressed as $g/(rad/s)^2$
$K_{ang.accel}$ = angular acceleration coefficient expressed in $g/(rad/s^2)$
w_i, w_p, w_o = angular velocity components along IA, PA and OA expressed in rad/s
$\dot{w_i}$, $\dot{w_p}$, $\dot{w_o}$ = angular acceleration components along IA, PA and OA expressed in rad/s^2.

$$\begin{aligned}\frac{E}{K_1} = K_0 + \frac{K_0'}{2} sign(a_i) + (1 + \frac{K_1'}{2} sign(a_i))a_i + K_{oq}a_i|a_i| + K_2 a_i^2 + K_3 a_i^3 \sum_{i \geq 4} K_n a_i^n \\ + \partial_0 a_p - \partial_p a_0 + K_{ip}a_i a_p + K_{io}a_i a_o + K_{po}a_p a_o + K_{pp}a_p^2 + K_{oo}a_o^2 + K_{spin}w_i w_p \\ + K_{ang.accel}w_0 + \varepsilon\end{aligned} \tag{3.17}$$

Although the equation seems large, some errors can be assumed to be negligible [Lele, 2010]. MEMS accelerometers typically show errors due to both dynamic and static errors, where dynamic errors include measurement noise. Static errors include characteristics and imperfections in the accelerometer where, according to the IEEE standard, the salient errors are due to scale factor, setup misalignment, and bias drift.

Bias and offset error

There are a number of ways for estimating this type of error, such as described in [Liu and Pang, 2001] where a robot arm was used to move the accelerometer back and forward and manually computing the thermal bias drift, or as in [Park, 2004] where a rotating wheel was used to calibrate the accelerometer over 360°. In many cases, calibrations with an accelerometer are carried out by having it lying perfectly horizontal, thus measuring only the static acceleration, also called *zero-g*. If this was done with the current accelerometer and assuming it was perfect, zero-g would be displayed as the integer value 16384. However, due to the errors in Equation 3.17, zero-g will deviate from this value. This is referred to as the *zero-g offset* and is not a constant value, but rather depends on factors such as thermal changes and self heating.

The magnitude of the errors vary depending on the grade of the sensor, see Table 3.1, and this is discussed further in [*Vector Nav*]. If one were to estimate the position over *t* seconds using only the accelerometer measurement, Equation 3.19 can be used. If, for example, an accelerometer is kept stationary only errors will be registered in the measurements. Furthermore, both the starting position and velocity is zero since the accelerometer is stationary, thus follows the second equation. As seen in Figure 3.6, the position estimation escalates over time, why it is of great importance to lower the errors. It was clear that using only the accelerometer measurements without compensating for errors would not suffice.

$$\begin{aligned} position &= position_0 + velocity_0 \cdot t + \frac{a}{2} \cdot t^2 \\ position &= \frac{a}{2} \cdot t^2 \end{aligned} \tag{3.19}$$

Scale factor error

The scale factor is the sensitivity which converts the accelerometer measurements to something useful for the user. In this case, the IMU outputs the measurements as integer values, and the scale factor converts this value to acceleration in g. A perfect accelerometer with a full-scale range of $\pm 2g$ would represent zero-g as 16384. Thus, to convert this value into 1g, one simply divides the output by the very same value. However, due to bias and zero-g offset, and the fact that the accelerometer

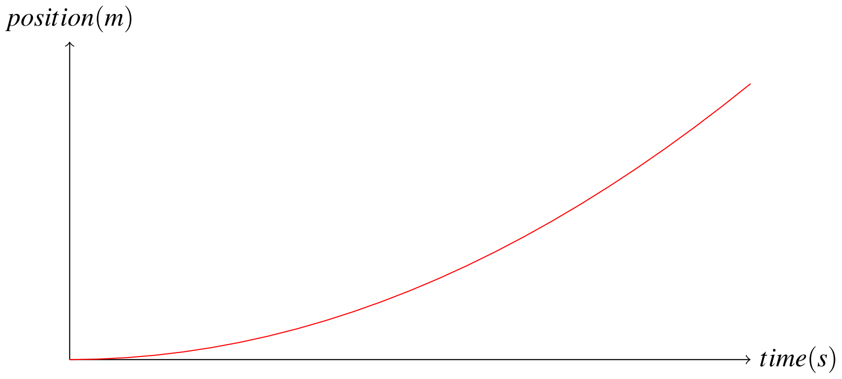

Figure 3.5 Position estimation using accelerometer measurements when the accelerometer is stationary.

Table 3.1 The errors which contributes to the bias error highly depends on the grade of the accelerometer [*Vector Nav*].

Grade	**Accelerometer bias errors (mg)**	**Horizontal position error**			
		1 sec	10 sec	1 min	1 hr.
Navigation	0.025	0.13 mm	0.12 mm	0.44 m	1.6 km
Tactical	0.3	1.5 mm	150 mm	5.3 m	19 km
Industrial	3.0	15 mm	1.5 m	53 m	190 km
Automotive	125	620 mm	60 m	2.2 km	7900 km

representation of acceleration is not linear, the scale factor is not 16384. Instead, the scale factor needs to be established so that the measurements are converted correctly. It is recommended in [IEEE, 2008] that four- or six-point tumble calibration is performed to find a better scale factor.

Due to an aforementioned lack of sophisticated equipment for extensive calibration, such as a robotic arm or a rotating disk, a six-point tumble calibration was performed. This was done by placing the accelerometer as horizontally as possible against a surface, and then the axis pointed towards Earth was measured. The sensor was then turned up-side-down, measuring the same axis. This was repeated for each axis.

Mechanical vibrations and initial misalignment error
Mechanical vibrations in the accelerometer can be observed in the measurements as measurement noise. Vibrations will be a major cause of noise, hence DLPF was used. Another way of minimizing this error is to prevent the vibrations from propagating to the accelerometer. This could be as simple as keeping it mechanically

isolated from the structure of the ship, thus making the module more resistant to vibrations.

The misalignment error is caused by the tilt of which the accelerometer is soldered on the circuit board. However, in this case the misalignment error also depends on how the IMU itself is installed, both inside the searchlight and on a ship. This error will also affects calibrations of the sensor since it is almost impossible to have it perfectly flat against the horizontal plane. Thus, misalignment should be taken into consideration during calibration, installation in the searchlight and when mounting the searchlight on a ship.

Again, looking at Equation 3.17, there are a lot of errors affecting an accelerometer and its measurements. However, [Lele, 2010] considers the bias drift and scale factor to be palpable error contributors for an accelerometer. Thus, the equation can be simplified to Equation 3.20, where A is the measured output from one axis, K_0 is the bias and offset error, K_1 is the scale factor error and ε is the random error residual of the other errors. The result, E, is the total acceleration measured by the accelerometer in g.

$$E = (A + K_0)(K_1) + \varepsilon \tag{3.20}$$

Calibration of accelerometer

Calibration of the accelerometer was done to minimize the errors, mainly for the zero-g offset and the scale factor. Although the misalignment error will be a factor during calibration, the goal of calibration is to minimize this error using the six-point calibration.

Zero-g offset was computed by measuring each axis twice - once having the axis pointing up and once pointing down. The difference between the two measurements was then divided by two to get the zero-g offset. This method was used for each axis by averaging 100 samples for each test. The result is given in Table 3.2.

Table 3.2 Calibration for each axis using the six-point calibration to get the new zero-g offset and sensitivity.

Axis	Pointed downwards	Pointed upwards	Sum	Zero-G offset	Scale factor
Z	-16914	15862	32776	-526	16388
X	-16197	16500	32697	303	16349
Y	-16593	16457	33050	-68	16525

Example

When pointing upwards, the measured static acceleration is 14745, and 18021 when pointing downwards. The difference is 3276, and half this value gives the zero-g offset as 1638. △

As seen in Table 3.2, the static acceleration differs between each measurement. To compute the scale factor, the two tests for each axis were averaged, and the result is seen as the last column in Table 3.2.

The calibrated output from each axis can now be expressed by Equation 3.21 below.

$$a_x = \frac{x - K_{0x}}{K_{1x}},\ a_y = \frac{y - K_{0y}}{K_{1y}},\ a_z = \frac{z - K_{0z}}{K_{1z}} \tag{3.21}$$

When stationary, the squared sum of each axis measurement should after the calibration be approximately 1g, i.e., $(a_x)^2 + (a_y)^2 + (a_z)^2 = 1$, independent of the orientation. Substituting this into Equation 3.21 results in Equation 3.22.

$$\left(\frac{x - K_{0x}}{K_{1x}}\right)^2 + \left(\frac{y - K_{0y}}{K_{1y}}\right)^2 + \left(\frac{z - K_{0z}}{K_{1z}}\right)^2 = 1, \tag{3.22}$$

Furthermore, due to the residual errors ε in Equation 3.20 which is not constant, the static acceleration will not be perfectly represented. In latter chapters of this thesis, ε is modeled as a random bias error and estimated so that the resulting measurements from the accelerometer contained as few errors as possible.

Estimating pitch and roll using accelerometer

An accelerometer can be used for computing the horizontal tilt of the sensor, as in e.g., [Pedley, 2013] where a Cartesian coordinate system is used. Furthermore, the output from each axis is summarized in a vector ${}^{S}A = \left(A_x, A_y, A_z\right)^T$, where A_x, A_y and A_z represent the measured outputs of x-, y- and z-axis respectively.

To get the orientation of the accelerometer with respect to Earth frame, the measurements are rotated as described in Section 3.3. Euler angles were used for the rotation matrix, and the rotation was defined as the *aerospace rotation sequence*, that is, R_{xyz}. To rotate from sensor to Earth frame, the rotation matrix was multiplied with the vector $\boldsymbol{A}$. Setting up an equation system for vector $\boldsymbol{A}$ and the rotation matrix made it possible to solve for the roll ϕ and pitch θ. However, this is done by assuming that the accelerometer is stationary, hence one can normalize vector $\boldsymbol{A}$ and set it as equal to the rotation matrix R_{xyz}.

$$\frac{\mathbf{A}}{||\mathbf{A}||} = R_{xyz} \Rightarrow \frac{1}{\sqrt{A_x^2 + A_y^2 + A_z^2}} \begin{pmatrix} A_x \\ A_y \\ A_z \end{pmatrix} = R_{xyz}$$

The intermediate calculations are omitted here since they are tedious, and the full derivation can be found in e.g., [Pedley, 2013].

Solving for the roll ϕ gives the equation as

$$\tan(\phi) = \left(\frac{-A_x}{\sqrt{A_y^2 + A_z^2}} \right) \iff \phi = \arctan\left(\frac{-A_x}{\sqrt{A_y^2 + A_z^2}} \right), \tag{3.23}$$

where a range restriction of tilt is needed, otherwise an unambiguous answer cannot be found. This is due to roll and pitch having an infinite number of solutions with multiples of 360°. Therefore, different applications use different conventions for the range restriction, e.g., ±90°. In this thesis, however, there is no need for restricting the range of either roll or pitch since it was found in Section 3.2 that the maximum tilt of a ship is usually less than 50°. The only time the tilt might exceed this value is when the ship has capsized, in which the control of the thermal camera is not needed for obvious reasons.

By the same approach, the pitch θ can be expressed as

$$\tan(\phi) = \left(\frac{-A_y}{\sqrt{A_x^2 + A_z^2}} \right) \iff \phi = \arctan\left(\frac{-A_y}{\sqrt{A_x^2 + A_z^2}} \right), \tag{3.24}$$

A problem with computing the tilt using this method is that the accelerometer is assumed to be stationary. As long as this assumption holds, the tilt will be correctly calculated using Equations 3.24 and 3.23. However, if the accelerometer is exposed to a linear acceleration along one axis, Equations 3.24 and 3.24 will no longer show the true tilt. Therefore, any linear acceleration will introduce errors in the orientation.

Gyroscope

The gyroscope was used for calculating the roll and pitch. Finding the appropriate full-scale range for the gyroscope is not as trivial as for the accelerometer, this due to a lack of research of the angular velocity present on a ship. To ensure adequate range, the full-scale range was set to the largest possible, ±2000°/s.

From [Freescale, 2015], as well as [Cao et al., 2015], it was found that the typical MEMS gyroscope usually contains five basic error terms: quantization, angle random walk, bias instability, rate random walk and rate ramp. These terms are generally measured using Allen Variance, as defined in [IEEE, 2014]. The total variance is defined in Equation 3.25, where

$\sigma_A^2(\tau)$ = Total error variance on the average time τ
N = Angle random walk coefficient
B = Bias instability coefficient
R = Rate ramp coefficient
Q = Quantum noise coefficient
K = Rate random walk coefficient

$$\sigma_A^2(\tau) = R^2\frac{\tau^2}{2} + K^2\frac{\tau}{3} + B^2\frac{2}{\pi}ln(2) + N^2\frac{1}{\tau} + Q^2\frac{3}{\tau^2} \tag{3.25}$$

According to [Cao et al., 2015], the exceeding errors for a MEMS gyroscope are angle random walk and rate ramp, where the primary target in this thesis was to minimize the angle random walk.

Angle random walk

A MEMS gyroscope suffers from mechanical imperfections, such as cross axis sensitivity, which affect the output of the gyroscope with white noise, that is, a zero-mean uncorrelated random variable [Glad and Ljung, 2007][Woodman, 2007]. The output of the gyroscope was given as angular velocity, so to yield the angle, or tilt, the value had to be integrated. However, numerical integration is only an approximation and will bring some errors, depending on what type of method is used. In this part, it was chosen as rectangular rule for its simplicity [Sauer, 2011]. Moreover, the noise introduces a zero-mean random walk error into the integrated signal, where the standard deviation

$$\sigma_\theta(t) = \sigma\sqrt{\Delta t \cdot t}$$

can be seen to grow proportionally to the square root of time, with Δt being the time between two successive samples. Thus, during a specified time period t the tilt will increase even if the gyroscope is stationary. This is what is called *angle random walk* [Woodman, 2007].

Bias instability

Due to noise, such as temperature and flickering in the electronics, the bias often changes over time [Vukmirica et al., 2010]. One way of minimizing this bias is to average samples over a long time period which will give a rough estimate about the bias. Although the tilt error caused by bias instability will be similar to the error caused by angle random walk, it differs in the way that bias instability has a range in which it affects the output, unlike angle random walk which can grow to infinity [Woodman, 2007].

Bias instability refers to the change in the bias measurement, and measurement of the bias instability is done at different times to see this change [*Bias Stability Measurement: Allan Variance*]. When tests were performed for the gyroscope it was found that it measured 2-3 bits of rotation even though it was stationary. This corresponds to a rotation of $\sim 0.17°/s$. Uncompensated, this bias will cause problems when using it to estimate tilt. This is handled more in later chapters.

Pressure sensor

The pressure sensor used, BMP085, did not have a dedicated microprocessor and could not collect and save data to a buffer continuously. Instead the MCU, which is taking measurements from the IMU with a sampling time of 10ms, had to start and collect pressure and temperature measurements separately. Pressure and temperature measurements are saved in the same register, hence only one measurement should be started at a time to prevent overwriting a value [*BMP180* 2013].

The sequence for starting and collecting data can be seen in the flow chart in Figure 3.6. Before altitude could be computed, the MCU had to first collect *calibration coefficients*, which consists of 11 coefficients. These were used for converting the measured temperature data to °C and pressure data to hPa. How this is done can be seen in detail in Appendix A. Since the pressure sensor is calibrated with unique coefficients at the factory, there was no need for extra calibration.

After the calibration coefficients were read, a temperature measurement had to be started, collected and calculated so that one more calibration coefficient was given. This coefficient was used for calculating the actual pressure, hence the temperature had to be read once before pressure.

Both temperature and pressure have a conversion time, i.e., a time delay after a measurement is started before a value can be collected. For temperature, this delay is 4.5ms. Moreover, since temperature vary at a slow rate it is sufficient to read the temperature once every second, or once every 100th sample [*BMP180* 2013]. The time delay for a pressure measurement, however, depends on the current mode in which the pressure sensor is running in, see Table 3.3. A more accurate measure-

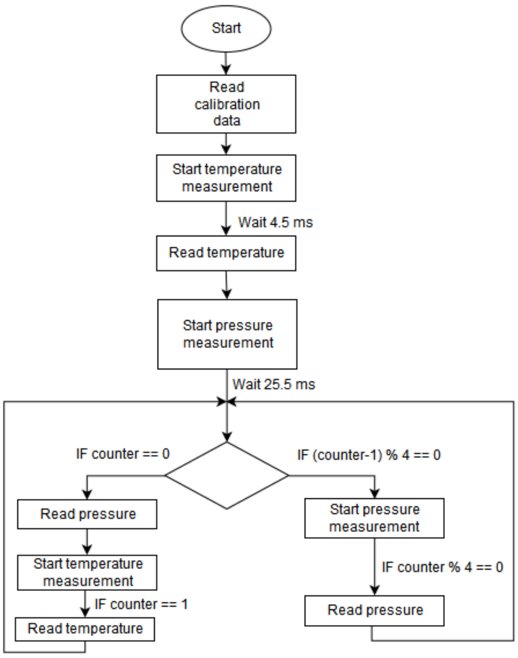

Figure 3.6 Flow chart showing the progress of the pressure sensor. The upper part is when the pressure sensor is started and the lower part shows a loop to emulate the scheduler.

ment is achieved by averaging more samples which in turn increases the conversion time. This has to be considered when creating a scheduling method, such as Figure 3.7.

Table 3.3 How different modes affect conversion time and noise.

Mode	Oversample parameter	Internal number of samples	Conversion time max. [ms]	RMS noise typ. [hPa]	RMS noise typ. [m]
Ultra low power	0	1	4.5	0.06	0.5
Standard	1	2	7.5	0.05	0.4
High resolution	2	4	13.5	0.04	0.3
Ultra high resolution	3	8	25.5	0.03	0.25

Picking the right mode for the pressure sensor is a trade-off between faster measurements versus more accurate measurements. When taken into account that the

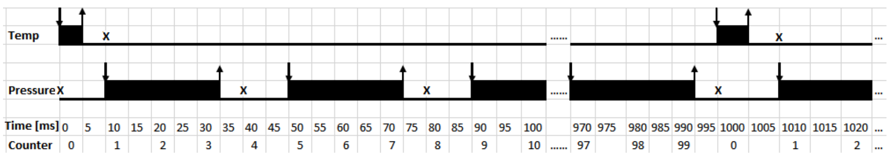
Temp
X
......
X
...
Pressure X
X
X
......
X
...
Time [ms] 0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 970 975 980 985 990 995 1000 1005 1010 1015 1020 ...
Counter 0 1 2 3 4 5 6 7 8 9 10 ...... 97 98 99 0 1 2 ...

Figure 3.7 Time line for temperature and pressure reading. The symbol X means that the MCU reads the value from the register. An arrow pointing down/up means a measurement is started/finished.

pressure sensor will be used for estimating the elevation and the fact that waves have a period time of several seconds, one can easily see that even the longest conversion time, 25.5ms, is sufficiently fast and also gives the most accurate result. Therefore, the mode was set to *ultra high resolution*. To make sure that the MCU did not try to retrieve a value before conversion was done, the MCU needed to wait at least 25.5ms. Since the MCU uses a sample time of 10ms, one can by waiting three samples (30ms) after a pressure measurement is started guarantee that the pressure sensor will finish the conversion in time.

The requirements on the scheduling was that temperature had to be started and read first, and then once every 100th sample. This was implemented using a counter which kept track of the samples. How the counter was used can be seen in the lower part of Figure 3.6. A measurement had to be collected before a new measurement was started, and a measurement could not be started or collected at the same time. This is marked with an X and arrow pointing down in Figure 3.7. Furthermore, the MCU had to wait 3 samples after a pressure measurement was started before the measurement was retrieved. With this setup, the pressure sensor had a sample rate of 40ms (25Hz) which is considered adequate for this thesis.

Altitude estimation

Altitude estimation can be represented as relative or absolute, where the *relative altitude* is the elevation from equilibrium and the *absolute altitude* is the true height above water. To convert pressure into altitude, Equation 3.26 was used. However, the pressure at sea level, p_0, had to be defined before altitude was computed. There are basically two ways in which this is done.

$$altitude = 44330\left(1-\left(\frac{p}{p_0}\right)^{\frac{1}{5.255}}\right) \tag{3.26}$$

One way is to use 1013.25hPa, which is the standard pressure at sea level, [SMHI, 2015]. But this is only the average pressure, while the true pressure normally varies between 950-1050hPa, or 540m above sea level to 301m below sea level. This can be seen in Figure 3.8, which is plotted using 3.26 with p_0 set to the standard pressure at sea level. Thus, on a good average day the corresponding altitude estimation will be very accurate. But for the rest of the year, the ship could either be estimated to be flying above water or submerged under water. The advantage using the standard pressure, however, is that the pressure at sea level does not require recalculation since the pressure is considered to be constant.

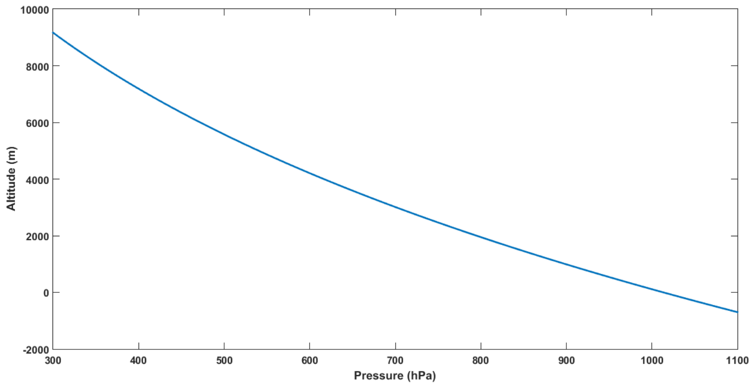

Figure 3.8 Altitude as a function of pressure. As the pressure increases, the altitude decreases.

The other way is to estimate the current pressure above sea level using Equation 3.27. This requires that the height above water for the searchlight, *lightHeight*, is known beforehand. This can, for example, be measured when the searchlight is installed on a ship. Using Equation 3.27, the current pressure above sea level is estimated which then can be used in Equation 3.26 to get the current altitude above water, i.e., the absolute altitude. To get the relative altitude, one simply subtracts the absolute altitude from the known starting point, *lightHeight*.

$$p_0 = \frac{p}{(1 - \frac{lightHeight}{44330})^{5.255}} \tag{3.27}$$

3.7 Sensor fusion

Sensor fusion was done to give more accurate, stable and reliable measurements compared to using each sensor on its own. In this thesis, two filters were investigated, complementary filter and Kalman filter, both of which are widely used today [Gui et al., 2015].

Complementary filter

This simple, yet powerful, filter requires little computation and is easy to implement. This is why it has gained so much in popularity compared to Kalman filter, especially in embedded systems [Gui et al., 2015]. The complementary filter works by combining a High-Pass Filter (HPF), used for the gyroscope measurement, and a Low-Pass Filter (LPF), used for the accelerometer measurement as seen in Figure 3.9 [Higgins, 1975].

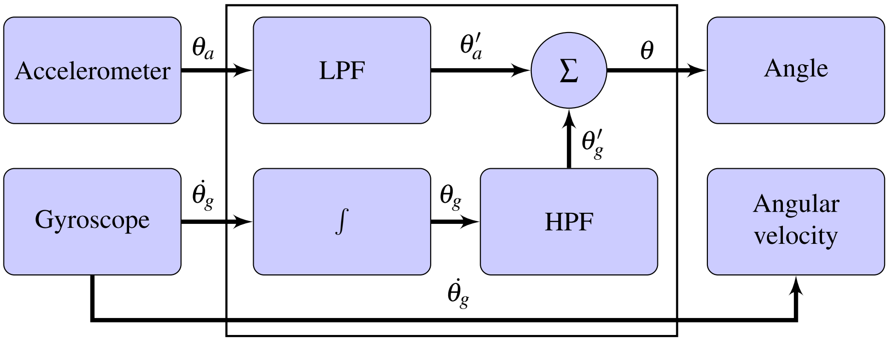

Figure 3.9 Tilt estimation using a complementary filter which combines a low- and high-pass filter for the accelerometer and gyroscope.

Studying the figure, one can see that the tilt θ_a is computed using the accelerometer. However, this signal contains high frequency noise which is filtered out using a LPF. This yields the filtered tilt θ'_a. In other words, the LPF ensures that only long-term changes are passed through, thus filtering out quick fluctuations.

The gyroscope measurement, $\dot{\theta}_g$, is measured as degrees per second. By integrating this value, one gets the tilt θ_g. This value is then passed through a HPF to remove low frequency noise from the gyroscope so that the filtered signal θ'_g is computed. Using a HPF, short-term fluctuations are passed through while long-term signals are filtered out. The long-term signals are typically the angle random walk.

The output from each filter, θ'_a and θ'_g, is combined to form the combined estimated tilt, θ. In the short term, the value from the gyroscope will dominate and the estimated tilt will adapt to quick changes in the real environment, whereas in the longer run the accelerometer will have a bigger impact which will suppress the angle random walk from the gyroscope [Higgins, 1975].

The complementary filter is defined in Equation 3.28, where α is the *filter coefficient*. The value on α determines how fast the filter converges to changes in the input signals. Furthermore, the sum of α and $1 - \alpha$ has to be 1 so that the output of the filter is accurate and with the right units [Gui et al., 2015]. As seen in the equation, the complementary filter works by using only one equation, thus requiring very little computation which is excellent for embedded systems. Furthermore, this filter is easily tuned empirically by simply changing the value of α.

$$\theta_{k+1} = \alpha(\theta_k + \dot{\theta}_{g_k} \cdot \Delta t) + (1-\alpha) \cdot \theta_{a_k} \tag{3.28}$$

Kalman filter

Kalman filter, known as a linear quadratic estimation, is an iterative linear filter which efficiently estimates the desired output [Higgins, 1975]. It takes noise in both processes and measurements into account by using covariance matrices and updates the matrices at every time interval, as well as using a model of the system to estimate the state based on both the current and previous state [Hägglund, 2012]. Although this does give tremendous results, there are some drawbacks. Due to it being an iterative filter, it requires high computational complexity which is computationally expensive, especially for larger systems [Higgins, 1975]. Furthermore, the accuracy of the mathematical model also determines how accurate the estimations are [Simon, 2001].

Since Kalman filter was used in an embedded system, the following is described in discrete time. A linear discrete model is described in Equation 3.30, where $\mathbf{v}_k$ is the process noise, $\mathbf{w}_k$ is the measurement noise and $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are matrices describing the system [Glad and Ljung, 2007][Hägglund, 2012].

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k + \mathbf{v}_k \\ \mathbf{y}_k &= \mathbf{C}\boldsymbol{x}_k + \mathbf{w}_k \end{aligned} \tag{3.30}$$

The disturbances are usually modeled as white noise, i.e., uncorrelated zero-mean Gaussian noise, with the process noise as $\mathbf{v}_k \sim N(0, \mathbf{Q}_k)$ and measurement noise as $\mathbf{w}_k \sim N(0, \mathbf{R}_k)$. Moreover, $\mathbf{Q}_k$ is the process noise covariance matrix and $\mathbf{R}_k$ is the measurement covariance matrix defined in Equation 3.32 [Glad and Ljung, 2007][Simon, 2001][*TKJ Electronics*].

$$\begin{aligned} \mathbf{Q}_k &= E[\mathbf{v}_k\mathbf{v}_k^T] \\ \mathbf{R}_k &= E[\mathbf{w}_k\mathbf{w}_k^T] \end{aligned} \tag{3.32}$$

To calculate the variance one first has to compute the average value, μ, and then the standard deviation, σ, as defined in Equation 3.34. The average value is computed from the acquired samples x$_i$, where $i = 1,2,\ldots,N$. The value of μ is then used for the standard deviation, which in turn is used for the variance which is defined as the square of the standard deviation, i.e., σ^2 [Blom, 2004].

$$\begin{aligned}\mu &= \frac{1}{N}(x_1 + x_2 + \cdots + x_N) \\ \sigma &= \sqrt{\frac{1}{N}\left((x_1-\mu)^2 + (x_2-\mu)^2 + \ldots (x_N-\mu)^2\right)}\end{aligned} \tag{3.34}$$

A Kalman filter can be divided into two steps - *prediction* and *update*. In the first step, the predicted state estimation vector $\hat{\mathbf{x}}_k$ and the predicted estimated covariance matrix $\hat{\mathbf{P}}_k$ is computed as in Equation 3.36. This is done using the updated estimated covariance $\mathbf{P}_{k-1}$ and the updated state estimation $\mathbf{x}_{k-1}$.

$$\begin{aligned}\hat{\mathbf{x}}_k &= \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_{k-1} \\ \hat{\mathbf{P}}_k &= \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^T + \mathbf{Q}_k\end{aligned} \tag{3.36}$$

In the update step described in Equation 3.38, the Kalman filter calculates the Kalman gain matrix, $\mathbf{K}_k$, which is used for updating the state vector $\mathbf{x}_k$ and covariance matrix $\mathbf{P}_k$ [Simon, 2001][Glad and Ljung, 2007]. The Kalman gain determines how much weight is put into the measured outputs, where a measurement with a high certainty corresponds to a greater weight on the Kalman gain [Gui et al., 2015]. Furthermore, the value of K is a trade-off between how fast the estimation converges to the true value and how sensitive the filter is to disturbances and model errors [Glad and Ljung, 2007][Hägglund, 2012].

$$\begin{aligned}\mathbf{K}_k &= \hat{\mathbf{P}}_k\mathbf{C}^T(\mathbf{C}\hat{\mathbf{P}}_k\mathbf{C}^T + \mathbf{R}_k)^{-1} \\ \mathbf{x}_k &= \hat{\mathbf{x}}_k + \mathbf{K}_k(\mathbf{y}_k - \mathbf{C}\hat{\mathbf{x}}_k) \\ \mathbf{P}_k &= (\mathbf{I} - \mathbf{K}_k\mathbf{C})\hat{\mathbf{P}}_k\end{aligned} \tag{3.38}$$

3.8 Image processing for absolute tracking

In this thesis it was investigated if image processing was possible and how it could be used. The idea of using image processing is to be able to do absolute tracking of an object present on the screen. One might, for example, want to keep an object fixed in the middle of the screen, and if said object is below the middle then the thermal camera needs to be lowered and vice versa.

Image processing can be done using OpenCV, which has a lot of support and can operate on embedded systems [Shah, 2014][*OpenCV*]. Mainly two types of problems were investigated - memory storage and processing time.

The thermal camera uses different color pallets to represent temperature differences. The least amount of bits needed is achieved by using grey scale which uses 8 bits, or 1 byte, per pixel. The thermal camera has a resolution of 336x256, or 86016 pixels. Since 1 byte is needed for grey scale per pixel, one image needs 86016 bytes, corresponding to 84kB [*FLIR* 2015]. Therefore, storing a single image needs 84kB of memory storage. Since the MCU has a total of 64kB data memory, storing an image with the current hardware is not possible. It gets worse considering the fact that the thermal camera also supports colors which requires even more memory. This could be solved by using an external storage unit, such as an SD-card, together with a custom hardware such as ThermalCapture [*ThermalCapture*].

The speed of which an image can be processed also has a huge impact on the overall system. If the image processing is too slow, the object on the screen might move very far between samples, potentially going out of screen. There is research, for instance [Shah, 2014] and [Coombs and Prabhu, 2011], where image processing was performed on embedded systems. However, the CPU speed in those studies were generally higher than for the CPU in this thesis and it still took a fairly long time to process an image. Considering the fact that the current CPU has a lower speed, one can expect the processing time to increase. A possible solution would be using a Digital Signal Processing (DSP) unit, as in [Coombs and Prabhu, 2011], which performs the heavy computations and sends the computed result to the MCU.

There are also practical problems to consider, such as an object disappearing behind a wave or more than one object being present on the screen. Furthermore, absolute tracking is not always wanted or needed, e.g., when searching for an object. During a search, keeping the image stable is much more important.

Due to insufficient hardware, one cannot expect image processing to work in this application as is. A lot of additional work is needed for absolute tracking using image processing which is outside the scope of this thesis. Thus, image processing for absolute tracking was not implemented in this work.

4 Estimation

To be able to control the direction of the thermal camera using the stepper motor, one has to know the error. In this case, the error is the deviation from the point at the surface of the sea which the operator wants to look at. To be able to calculate this error, one needs to estimate the tilt and elevation of the ship.

4.1 Tilt estimation

A Kalman filter was used for fusing together the accelerometer and gyroscope measurements. This filter was mainly chosen due to having an error model which when used correctly makes it more robust and less sensitive to disturbances.

As shown in previous sections, the tilt computed using a gyroscope suffers from random angle walk due to bias errors, and to reduce the random angle walk one ought to remove the bias error. This is done in Equation 4.1, where θ_k is the current tilt, θ_{k-1} is the previously calculated tilt, $(\dot{\theta}_g)_k$ is the current gyroscope measurement, $(\dot{\theta}_b)_k$ is the current gyroscope bias and Δt is the sample time. Since the gyroscope measurement and bias is in the unit of degrees per second, the tilt in degrees is simply computed by multiplying with Δt.

$$\theta_k = \theta_{k-1} + (\dot{\theta}_g)_k \cdot \Delta t - (\dot{\theta}_b)_k \cdot \Delta t \tag{4.1}$$

The equation above could be represented as state space, where the state vector $\mathbf{x}_k$ contains variables which are to be estimated, in this case the tilt and gyroscope bias.

$$\mathbf{x}_k = \begin{pmatrix} \theta \\ \dot{\theta}_b \end{pmatrix}_k$$

The state space representation can be seen in Equation 4.2.

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}u_k + \mathbf{v}_k \iff \begin{pmatrix} \theta \\ \dot{\theta}_b \end{pmatrix}_k = \begin{pmatrix} 1 & -\Delta t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \theta \\ \dot{\theta}_b \end{pmatrix}_{k-1} + \begin{pmatrix} \Delta t \\ 0 \end{pmatrix} \dot{\theta}_k + \mathbf{v}_k \quad (4.2)$$

The process noise was assumed to be uncorrelated zero-mean Gaussian, i.e. $\mathbf{v}_k \sim N(0, \mathbf{Q}_k)$, and $\mathbf{Q}_k$ is the process noise covariance matrix which in this case is the state estimation of the accelerometer and gyroscope bias. Since the process noise was assumed to be uncorrelated, which it is in many cases, $\mathbf{Q}_k$ only contains values in its diagonal [Glad and Ljung, 2007]. In other words, there was no correlation between the gyroscope bias and the accelerometer measurements of the tilt. Observe that the matrix is multiplied by Δt, since $\mathbf{Q}$ is time based [*TKJ Electronics*]. And since Δt is a constant, $\mathbf{Q}$ will also be constant. Thus, the resulting $\mathbf{Q}$-matrix is given in Equation 4.4.

$$\begin{aligned} \mathbf{R} &= E[\mathbf{w}_k \mathbf{w}_k^T] = var(w_k) = var(w) \\ \mathbf{Q} &= \begin{pmatrix} Q_\theta & 0 \\ 0 & Q_{\dot{\theta}_b} \end{pmatrix} \cdot \Delta t \end{aligned} \quad (4.4)$$

The measurement matrix $\mathbf{y}$ is defined as $\mathbf{y}_k = \mathbf{C}\mathbf{x}_{k-1} + \mathbf{w}_k$, where $\mathbf{C}$ is the observation model and $\mathbf{w}_k$ is the measurement noise. The state vector $\mathbf{x}$ contained the tilt and the gyroscope bias, but only the tilt was measured directly using the accelerometer. Thus, $\mathbf{C} = \begin{pmatrix} 1 & 0 \end{pmatrix}$. Furthermore, the measurement noise $\mathbf{w}_k$ was assumed to be uncorrelated zero-mean Gaussian with the covariance matrix $\mathbf{R}$, so that $\mathbf{w}_k \sim N(0, \mathbf{R})$. Additionally, only the tilt was measured directly, hence $\mathbf{R}$ only contained the variance of the accelerometer measurement noise [*TKJ Electronics*]. Since the covariance of the same variable is equal to the variance, $\mathbf{R}$ is defined as in Equation 4.4 [Blom, 2004]. Moreover, the measurement noise was also assumed to be constant, although this is not fully true due to small errors such as thermal self-heating in an accelerometer. But as stated in the section about accelerometers, those errors are negligible.

In summary, Kalman filter for estimating the tilt uses three variables that can be tuned, Q_θ, $Q_{\dot{\theta}_b}$ and the measurement noise of the accelerometer, R. To find the correct values, simulations were first performed before doing empirical experiments to fine tune it for the real process [*TKJ Electronics*].

4.2 Gravity compensation for linear acceleration

In Section 3.3, two frames of references were presented - sensor frame S and Earth frame E. In this section, the static acceleration is removed from the accelerometer measurements, leaving only the linear acceleration. Before removing the static acceleration, however, the measurements need to be rotated from sensor to Earth frame.

Rotation and gravity compensation was based on Madgwick's algorithm, where rotation was performed using quaternion. This algorithm was chosen because it is computationally inexpensive while still being effective at low sampling rates. Madgwick's algorithm is based on an optimized gradient-descent algorithm and utilizes both gyroscope and accelerometer measurements [Madgwick, 2010].

Orientation from gyroscope

The orientation of the sensors were calculated using quaternions together with the angular velocity measured from the gyroscope, where each axis was represented as ω_x, ω_y and ω_z. Each axis was then arranged into the quaternion vector ${}^S\boldsymbol{\omega} = \begin{bmatrix} 0 & \omega_x & \omega_y & \omega_z \end{bmatrix}$.

Using the quaternion arithmetic, one can describe the rate of change of orientation of the Earth frame relative to the sensor frame as ${}^S_E\dot{\mathbf{q}} = \frac{1}{2}{}^S_E\hat{\mathbf{q}} \otimes {}^S\boldsymbol{\omega}$. By numerically integrating this value, i.e., multiplying with the sample time Δt and adding that to the previously quaternion orientation estimate ${}^S_E\hat{\mathbf{q}}_{est,t-1}$ as in Equation 4.6, one gets the orientation of the Earth frame relative to the sensor frame at time t [Madgwick, 2010].

$$\begin{aligned} {}^S_E\dot{\mathbf{q}}_{\omega,t} &= \frac{1}{2}{}^S_E\hat{\mathbf{q}}_{est,t-1} \otimes {}^S\boldsymbol{\omega}_t \\ {}^S_E\mathbf{q}_{\omega,t} &= {}^S_E\hat{\mathbf{q}}_{est,t-1} + {}^S_E\dot{\mathbf{q}}_{\omega,t} \cdot \Delta t \end{aligned} \tag{4.6}$$

Orientation from accelerometer

To get the magnitude and direction of the static acceleration, the measurements from the accelerometer were used. To represent this using quaternions, Madgwick showed that the orientation of the accelerometer, ${}^S_E\hat{\mathbf{q}}$, could be optimized as in Equation 4.7 with the objective function as Equation 4.8. In these equations, ${}^E\hat{\mathbf{d}}$ is a predefined reference direction of the field in Earth frame[1] and ${}^S\hat{\mathbf{s}}$ is the measured direction of the field in the sensor frame. The components of the three vectors are defined in Equation 4.10.

[1] In Earth frame, static acceleration only exist in the z-axis.

$$\min_{{}^S_E\hat{\mathbf{q}}\in\Re^4} \mathbf{f}({}^S_E\hat{\mathbf{q}}, {}^E\hat{\mathbf{d}}, {}^S\hat{\mathbf{s}}) \tag{4.7}$$

subject to

$$\mathbf{f}\left({}^S_E\hat{\mathbf{q}}, {}^E\hat{\mathbf{d}}, {}^S\hat{\mathbf{s}}\right) = {}^S_E\hat{\mathbf{q}}^* \otimes {}^E\hat{\mathbf{d}} \otimes {}^S_E\hat{\mathbf{q}} - {}^S\hat{\mathbf{s}} \tag{4.8}$$

where

$$\begin{aligned} {}^S_E\hat{\mathbf{q}} &= \begin{bmatrix} q_1 & q_2 & q_3 & q_4 \end{bmatrix} \\ {}^E\hat{\mathbf{d}} &= \begin{bmatrix} 0 & d_x & d_y & d_z \end{bmatrix} \\ {}^S\hat{\mathbf{s}} &= \begin{bmatrix} 0 & s_x & s_y & s_z \end{bmatrix} \end{aligned} \tag{4.10}$$

There are a number of ways for solving the minimization problem, where Madgwick uses the gradient descent algorithm because of its simplicity and efficiency [Böiers, 2010]. Equation 4.11 shows the gradient descent algorithm for n iterations which computes the orientation estimation ${}^S_E\hat{\boldsymbol{q}}_{n+1}$ based on an initial guess, ${}^S_E\hat{\boldsymbol{q}}_0$, and a constant step-size μ [Madgwick, 2010]. The step-size μ determines how fast the result converges, but might converge to the wrong value if the value is too large. One could make μ time dependent and changed by each iteration, but this increases the work load significantly since it has to use the second derivative of the objective function, also called *Hessian* [Böiers, 2010]. However, if the convergence rate governed by μ is equal or greater than the rate of change of orientation in the real process, it is sufficient to use a constant μ [Madgwick, 2010]. Since the rate of change of a ship is fairly slow, it is acceptable for μ to be constant in this thesis.

Equation 4.11 is solved by computing the gradient, ∇, of the objective function. The gradient of the objective function is defined in Equation 4.12, which uses the Jacobian $\mathbf{J}({}^S_E\hat{\boldsymbol{q}}_k, {}^E\hat{\boldsymbol{d}})$ and the objective function $\mathbf{f}({}^S_E\hat{\boldsymbol{q}}_k, {}^E\hat{\boldsymbol{d}}, {}^S\hat{\boldsymbol{s}})$ itself [Böiers, 2010][Madgwick, 2010]. However, the Jacobian and the objective function are expressed in the general form without a predefined direction of field. Since Earth frame is considered, one can set the direction of field ${}^E\hat{\boldsymbol{d}}$ to ${}^E\hat{\boldsymbol{g}} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}$ by using the gravity field. Furthermore, the normalized accelerometer measurements can be written as ${}^S\hat{\boldsymbol{a}} = \begin{bmatrix} 0 & a_x & a_y & a_z \end{bmatrix}$. Substituting ${}^E\hat{\boldsymbol{g}}$ and ${}^S\hat{\boldsymbol{a}}$ for ${}^E\hat{\boldsymbol{d}}$ and ${}^S\hat{\boldsymbol{s}}$, one gets the new and simplified Jacobian and objective function as Equation 4.13 and 4.14.

$${}^S_E\mathbf{q}_{k+1} = {}^S_E\hat{\mathbf{q}} - \mu \frac{\nabla\mathbf{f}\left({}^S_E\hat{\mathbf{q}}_k, {}^E\hat{\boldsymbol{d}}, {}^S\hat{\mathbf{s}}\right)}{\|\nabla\mathbf{f}\left({}^S_E\hat{\mathbf{q}}_k, {}^E\hat{\boldsymbol{d}}, {}^S\hat{\mathbf{s}}\right)\|}, k = 0, 1, 2, \ldots, n \tag{4.11}$$

$$\nabla\mathbf{f}\left({}^S_E\hat{\mathbf{q}}_k, {}^E\hat{\boldsymbol{d}}, {}^S\hat{\mathbf{s}}\right) = \boldsymbol{J}^T\left({}^S_E\hat{\boldsymbol{q}}_k, {}^E\hat{\boldsymbol{d}}\right)\mathbf{f}\left({}^S_E\hat{\boldsymbol{q}}_k, {}^E\hat{\boldsymbol{d}}, {}^S\hat{\mathbf{s}}\right) \tag{4.12}$$

$$\mathbf{J}({}^S_E\hat{\boldsymbol{q}}) = \begin{bmatrix} 2q_3 & 2q_4 & -2q_1 & 2q_2 \\ 2q_2 & 2q_1 & 2q_4 & 2q_3 \\ 0 & -4q_2 & -4q_3 & 0 \end{bmatrix} \tag{4.13}$$

$$\mathbf{f}({}^S_E\hat{\boldsymbol{q}}, {}^S\hat{\boldsymbol{a}}) = \begin{bmatrix} 2(q_2q_4 - q_1q_3) - a_x \\ 2(q_1q_2 + q_3q_4) - a_y \\ 2(\frac{1}{2} - q_2^2 - q_3^2) - a_z \end{bmatrix} \tag{4.14}$$

Equation 4.11 can now be rewritten as Equation 4.15, which computes the estimated orientation, ${}^S_E\boldsymbol{q}_{\nabla,t}$, at the current time t based on the previous orientation estimation ${}^S_E\hat{\boldsymbol{q}}_{est,t-1}$ and the objective function gradient $\nabla\mathbf{f}$ sampled at time t. The value of μ_t has to be big enough so that the convergence rate of ${}^S_E\boldsymbol{q}_{\nabla,t}$ is faster than the rate of change of the real process, but small enough to avoid overshooting due to larger step size. According to Madgwick, μ_t can be calculated by Equation 4.16, where ${}^S_E\dot{\boldsymbol{q}}_{\omega,t}$ is the angular velocity from the gyroscope and β is an augmentation of μ to account for noise in accelerometer measurements [Madgwick, 2010].

$${}^S_E\boldsymbol{q}_{\nabla,t} = {}^S_E\hat{\boldsymbol{q}}_{est,t-1} - \mu_t \frac{J^T({}^S_E\hat{\boldsymbol{q}}_{est,t-1})\mathbf{f}({}^S_E\hat{\boldsymbol{q}}_{est,t-1}, {}^S\hat{\boldsymbol{a}}_t)}{\|J^T({}^S_E\hat{\boldsymbol{q}}_{est,t-1})\mathbf{f}({}^S_E\hat{\boldsymbol{q}}_{est,t-1}, {}^S\hat{\boldsymbol{a}}_t)\|} \tag{4.15}$$

$$\mu_t = \beta \|{}^S_E\dot{\boldsymbol{q}}_{\omega,t}\|\Delta t, \beta > 1 \tag{4.16}$$

Sensor fusion

So far, the estimated orientation has been presented for the gyroscope, ${}^S_E\boldsymbol{q}_{\omega,t}$ and the accelerometer, ${}^S_E\boldsymbol{q}_{\nabla,t}$ separately. In this section the estimations are fused together to obtain a better estimated orientation of the sensors relative the Earth frame, ${}^S_E\boldsymbol{q}_{est,t}$. This is done using the simple, yet powerful, complementary filter as described in Section 3.7 and defined in Equation 4.17.

$${}^S_E\boldsymbol{q}_{est,t} = \alpha_t {}^S_E\boldsymbol{q}_{\nabla,t} + (1-\alpha_t){}^S_E\boldsymbol{q}_{\omega,t} \tag{4.17}$$

The optimal value for α_t is when the weighted divergence of ${}^S_E\boldsymbol{q}_\omega$ is equal to the weighted convergence of ${}^S_E\boldsymbol{q}_\nabla$, as in Equation 4.18. The variable γ is the divergence rate of ${}^S_E\boldsymbol{q}_\omega$ expressed as the magnitude of a quaternion derivative corresponding to the gyroscope measurement error, and $\frac{\mu_t}{\Delta t}$ is the convergence rate of ${}^S_E\mathbf{q}_\nabla$ as defined in Equation 4.16.

$$(1-\alpha_t)\gamma = \alpha_t \frac{\mu_t}{\Delta t} \iff \alpha_t = \frac{\gamma}{\frac{\mu_t}{\Delta t} + \gamma} \tag{4.18}$$

As seen in Equation 4.16, the value of μ_t depends on the value of β. Since the orientation changes slowly in the process, β can be set to be very large, thus making μ_t very large as well. And if μ_t is very large, then the first term in Equation 4.15, ${}^S_E\hat{\boldsymbol{q}}_{est,t-1}$, becomes negligible. Therefore, Equation 4.15 can be rewritten to Equation 4.19.

$${}^S_E\boldsymbol{q}_{\nabla,t} \approx -\mu_t \frac{\nabla f}{\|\nabla f\|} \tag{4.19}$$

The value of α_t in Equation 4.18 also depends on μ_t. Since $\Delta t \to 0$ for a shorter sample period, then $\frac{\mu_t}{\Delta t} \to \infty$, hence γ in the denominator becomes negligible and the equation can be simplified to Equation 4.20. Moreover, $\mu_t \gg \gamma\Delta t$ means $\alpha_t \approx 0$.

$$\alpha_t \approx \frac{\gamma \Delta t}{\mu_t} \tag{4.20}$$

Substituting the angle estimation ${}^S_E\boldsymbol{q}_{\omega,t}$ from Equation 4.6 and Equations 4.19-4.20 into the complementary filter above results in Equation 4.21, where α_t is substituted both as Equation 4.20 and $\alpha_t \approx 0$.

$${}^S_E\boldsymbol{q}_{est,t} = \frac{\gamma\Delta t}{\mu_t}\left(-\mu_t \frac{\nabla \mathbf{f}}{\|\nabla \mathbf{f}\|}\right) + (1-0)\left({}^S_E\hat{\boldsymbol{q}}_{est,t-1} + {}^S_E\dot{\boldsymbol{q}}_{\omega,t}\Delta t\right) = {}^S_E\hat{\boldsymbol{q}}_{est,t-1} + {}^S_E\dot{\boldsymbol{q}}_{\omega,t}\Delta t \tag{4.21}$$

The second term in the equation above can be rewritten as Equation 4.22, where ${}^S_E\dot{\hat{\boldsymbol{q}}}_{\varepsilon,t}$ is the direction of the error of ${}^S_E\boldsymbol{q}_{est,t}$ defined in Equation 4.23.

$${}^S_E\dot{\boldsymbol{q}}_{est,t} = {}^S_E\dot{\boldsymbol{q}}_{\omega,t} - \eta\, {}^S_E\dot{\hat{\boldsymbol{q}}}_{\varepsilon,t} \tag{4.22}$$

$${}^S_E\dot{\hat{\boldsymbol{q}}}_{\varepsilon,t} = \frac{\nabla \mathbf{f}}{\|\nabla \mathbf{f}\|} \tag{4.23}$$

In summary, the proposed filter calculates the orientation ${}^S_E\boldsymbol{q}_{est}$ by numerically integrating the estimated orientation rate ${}^S_E\dot{\boldsymbol{q}}_{est}$. Furthermore, the filter calculates the rate of change of orientation measured by the gyroscope ${}^S_E\dot{\boldsymbol{q}}_{est}$, where the magnitude of the gyroscope measurement error, η, is removed in the direction of the estimated error, ${}^S_E\dot{\hat{\boldsymbol{q}}}_\varepsilon$, computed from the accelerometer. The practical use for this feedback, i.e., subtracting with $\eta{}^S_E\dot{\hat{\boldsymbol{q}}}_{\varepsilon,t}$, is to estimate and remove the bias errors which otherwise would contribute to angle random walk.

The resulting vector ${}^S_E\boldsymbol{q}_{est}$ is the rotation needed to rotate from sensor to Earth frame. It was also mentioned earlier that the static acceleration in Earth frame acts in the vertical z axis, i.e. ${}^E\boldsymbol{g} = \begin{bmatrix}0 & 0 & 1\end{bmatrix}^T$. Before removing the static acceleration, the accelerometer values are mapped from sensor to Earth frame as in Equation 4.25. To extract the vertical acceleration, one takes the sum of z components in the Earth frame vector ${}^E\boldsymbol{a}$, as in 4.26 [Madgwick, 2010][Nair, 2014].

$$\begin{aligned}&{}^E\boldsymbol{a} = \boldsymbol{q} \otimes {}^S\boldsymbol{a} \otimes \boldsymbol{q}^* = {}^S_E\boldsymbol{R}^S \cdot \boldsymbol{a} = \\ &\begin{bmatrix} q_1^2+q_2^2-q_3^2-q_4^2 & 2(q_2q_3-q_1q_4) & 2q_2q_4+2q_1q_3 \\ 2q_2q_3+2q_1q_4 & q_1^2-q_2^2+q_3^2-q_4^2 & 2q_3q_4-2q_1q_2 \\ 2q_2q_4-2q_1q_3 & 2q_3q_4+2q_1q_2 & q_1^2-q_2^2-q_3^2+q_4^2 \end{bmatrix} \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}\end{aligned} \tag{4.25}$$

$${}^Ea_z = (2q_2q_4-2q_1q_3)a_x + (2q_3q_4+2q_1q_2)a_y + (q_1^2-q_2^2-q_3^2+q_4^2)a_z \tag{4.26}$$

The resulting scalar ${}^Ea_z$ is the total vertical acceleration - that is, the sum of both static and linear acceleration - measured on all axis. Since all is measured in g, one can remove the static acceleration by subtracting by 1 from the resulting scalar, thus giving the total vertical linear acceleration as ${}^Ea_{zlinear} = {}^Ea_z - 1$ [Nair, 2014][Varesano, 2011].

4.3 Elevation estimation methods

Different methods, such as Kalman filter, were investigated to estimate the elevation of the searchlight using only the IMU. However, the best approach for finding the elevation was found not by using a filter, but instead using the characteristics of a ship at sea.

In latter parts, a pressure sensor was brought in to see how much better estimations one could get when using a Kalman filter to fuse together both accelerometer and pressure measurements.

Elevation estimation using accelerometer

The easiest, and also the worst, method for estimating the position using only the accelerometer is by integrating the acceleration after it has been compensated for gravity. Due to errors in both the measurements and integration, the position estimation will drift over time. However, by using limitations and characteristics of ships, one can reduce the errors. For example, a ship cannot be infinitely high above water, but will rather stay within a range of ±30m from its equilibrium, 30m being the highest possible wave that a ship might encounter. This can be used when removing the position drift.

To get the position, the acceleration was integrated numerically. Since the sample time was very small, the acceleration between two samples was assumed to be constant. Due to this, the equations of motion were used [Seifert and Camacho, 2007]. The velocity was computed by integrating the acceleration as Equation 4.28. The velocity was then integrated to yield the position, see Equation 4.30.

$$\begin{gathered}
a = \frac{dv}{dt} \\
dv = a \cdot dt \\
\int_{v_0}^{v} dv = \int_{0}^{\Delta t} a \, dt \\
v - v_0 = a \cdot \Delta t \iff v = v_0 + a \cdot \Delta t
\end{gathered} \tag{4.28}$$

$$\begin{gathered}
v = \frac{dp}{dt} \\
dp = v \cdot dt = (v_0 + a \cdot \Delta t)dt \\
\int_{p_0}^{p} dp = \int_{0}^{\Delta t} (v_0 + a \cdot \Delta t) \, dt \\
p - p_0 = v_0 \cdot \Delta t + \frac{1}{2} a \cdot \Delta t^2 \iff p = p_0 + v_0 \cdot \Delta t + \frac{1}{2} a \cdot \Delta t^2
\end{gathered} \tag{4.30}$$

The simple form of integration is visualized in Figure 4.1, where Δt is the difference between two successive samples, e.g. $\Delta t = x_1 - x_0$. To reduce the error, one would like $\Delta t \to 0$, which is not feasible. The integration of the true acceleration, *f(x)*, is simply the sum of every sample, also called *Riemann sum*, as defined in Equation 4.31, where $\Delta x = \frac{b-a}{n}$ and n is the number of samples [Persson and Böiers, 2010].

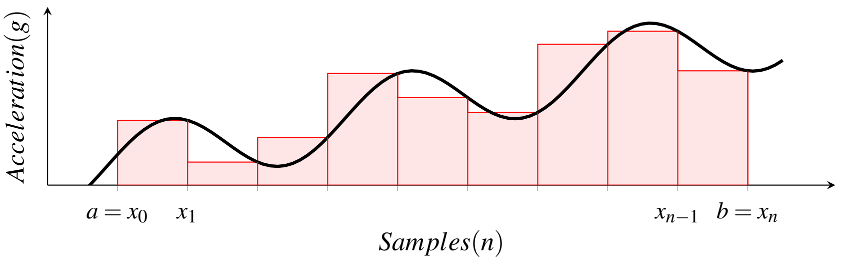

Figure 4.1 Function for the acceleration and the sampled measurements, where the red area is the integrated value using Riemann sum.

$$\int_a^b f(x)dx = \lim_{n\to\infty} \sum_{i=1}^{n} f(x_i)\Delta x \tag{4.31}$$

It is obvious from the plot in Figure 4.1 that there will be errors when integrating, and the errors can be minimized by lowering the sample period. However, the sample period is a fixed value, in this case 10ms, and so errors will be introduced. Since it is of great importance that errors are reduced to minimize drift, the integration error needs to be reduced. This is done using trapezoidal integration, as seen in Figure 4.2, which clearly reduced the total error. One could use other methods, such as Simpson's rule, to further improve integration, but trapezoidal integration was deemed adequate for this application since more accurate methods require more computations. Unlike the previous method, which simply takes each measurement and multiplies it with the sample period, the trapezoidal integration uses a first order interpolation to compute the area. Therefore, the integrated sum is equal to $\sum_{i=1}^{n} area_i$, where $area_i$ is defined in Equation 4.32 [Sauer, 2011][Seifert and Camacho, 2007].

$$area_n = sample_n + \frac{sample_n - sample_{n-1}}{2} \cdot \Delta t \tag{4.32}$$

A first approach to see how good the aforementioned method works for estimating the position is by moving the accelerometer sideways along a y-axis from one position to another. If this motion is plotted, the acceleration would look like picture (A) in Figure 4.3 and the integrated acceleration would give the velocity as picture (B). Note that since errors are present in the integration, the velocity will not return to zero even though the accelerometer is stationary in the end. Looking at picture (C), which plots the position by integrating the velocity, the position can be seen to increase even though the accelerometer is not moving.

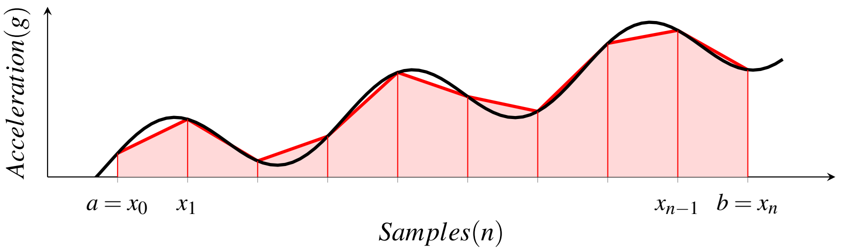

Figure 4.2 Function for the acceleration and the sampled measurements. The red area is the integrated value using trapezoidal rule.

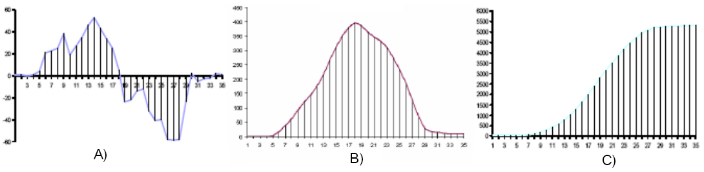

Figure 4.3 Three plots showing the acceleration (A), the integrated acceleration (B), and the integrated velocity (C). Due to errors when integrating, the position will drift [Seifert and Camacho, 2007].

An Exponential Moving Average (EMA) filter, defined in Equation 4.33, was used for reducing the errors in the accelerometer measurements [*Moving Average Filter*]. As seen in the equation, the current filtered measurement y_t is computed from a portion of the current unfiltered measurement x_t with a portion of the previously filtered measurement y_{t-1}. Thus, a bigger α will use a bigger portion of the current measurement, making the filter respond quicker to changes but more sensitive to disturbances, and vice versa.

$$y_t = \alpha \cdot x_t + (1 - \alpha) \cdot y_{t-1} \tag{4.33}$$

By using an EMA filter, errors were reduced but not eliminated. Without any restrictions or constrains on the position, the drift due to the errors will still increase to infinity. As aforementioned, elevation cannot go to infinity since the ship would then be flying in space. Even if a ship were to encounter a monster wave, which is extremely rare, the elevation would not exceed 30m. Therefore, one can assume that elevation cannot possibly reach more than 30m. By this assumption, one can use a feedback to bring the elevation estimation back down to equilibrium.

Elevation estimation using accelerometer and pressure sensor

A pressure sensor was used as an auxiliary sensor to the IMU. The idea was to see how one could combine an accelerometer and a pressure sensor to estimate the elevation and examine if it was better than the previous method, and in such case how much better. This was done using a Kalman filter.

To be able to use the Kalman filter, a dynamic model of the system was set up in state space, where three states were introduced: x_k (elevation), $\dot{x}_k$ (elevation rate) and $\ddot{x}b_k$ (accelerometer bias). To get the elevation, the equations of motion were used as below.

$$\begin{aligned} x &= x_0 + \dot{x}_0 \cdot \Delta t + \frac{a}{2} \cdot \Delta t \\ \dot{x} &= \dot{x}_0 + a \cdot \Delta t \end{aligned} \tag{4.35}$$

The acceleration a is the vertical linear acceleration against Earth frame, ${}^{E}a_{zlinear}$, as calculated in Section 4.2 using an accelerometer. However, the measured acceleration still contains a bias error that should be removed for a more accurate elevation estimation. In other words, to get the true acceleration without the bias, one has to subtracts the accelerometer bias from the measured acceleration, i.e. $a = \ddot{x} - \ddot{x}b$ where $\ddot{x}$ is the vertical linear acceleration in Earth frame. Substituting this expression for the variable a in Equation 4.35 gives Equation 4.37.

$$\begin{aligned} x &= x_0 + \dot{x}_0 \cdot \Delta t + \frac{(\ddot{x} - \ddot{x}b)}{2} \cdot \Delta t^2 \\ \dot{x} &= \dot{x}_0 + (\ddot{x} - \ddot{x}b) \cdot \Delta t \end{aligned} \tag{4.37}$$

Since it was implemented on an embedded system, the equations of motion were converted to a discrete system as in Equation 4.39, where the last state is the accelerometer bias with added random noise ε [Nair, 2014].

$$\begin{aligned} x_k &= x_{k-1} + \dot{x}_{k-1} \cdot \Delta t + \frac{(\ddot{x}_{k-1} - \ddot{x}b_{k-1})}{2} \cdot \Delta t^2 \\ \dot{x}_k &= \dot{x}_{k-1} + ((\ddot{x}_{k-1} - \ddot{x}b_{k-1})) \cdot \Delta t \\ \ddot{x}b_k &= \ddot{x}b_{k-1} + \varepsilon \end{aligned} \tag{4.39}$$

The equations above were represented in a state space as below. Note that the input matrix **B** is zero since no signal is used as an input.

$$\boldsymbol{x}_k = \boldsymbol{A}\boldsymbol{x}_{k-1} + \boldsymbol{B}\boldsymbol{u}_k + \boldsymbol{v}_k =$$

$$\underbrace{\begin{bmatrix} x_k \\ \dot{x}_k \\ \ddot{x}b_k \end{bmatrix}}_{\boldsymbol{x}_k} = \underbrace{\begin{bmatrix} 1 & \Delta t & \frac{-\Delta t^2}{2} \\ 0 & 1 & -\Delta t \\ 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{A}} \underbrace{\begin{bmatrix} x_{k-1} \\ \dot{x}_{k-1} \\ \ddot{x}b_{k-1} \end{bmatrix}}_{\boldsymbol{x}_{k-1}} + \underbrace{\begin{bmatrix} \frac{\Delta t^2}{2} \cdot \ddot{x}b_{k-1} \\ \Delta t \cdot \ddot{x}b_{k-1} \\ \varepsilon \end{bmatrix}}_{\boldsymbol{v}_k}$$

It was only the elevation, x_k, that could be measured directly using the pressure sensor. Due to this, the measurement vector $\boldsymbol{y}$ only contains a scalar value as in Equation 4.40, i.e, $y_k = x_k + w_k$, where w_k is the random scalar measurement noise for the pressure sensor.

$$\boldsymbol{y}_k = \boldsymbol{C}\boldsymbol{x}_k^T + \boldsymbol{w}_k = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_k \\ \dot{x}_k \\ \ddot{x}b_k \end{bmatrix} + \boldsymbol{w}_k \qquad (4.40)$$

The next step is calculating the process noise covariance matrix, $\boldsymbol{Q}_k$ as defined below, where the vector $\boldsymbol{v}_k$ represents the perturbation of the model from the above state space. It is the variables σ_{acc} and $\sigma^2_{accbias}$ in the $\boldsymbol{Q}_k$-matrix that were tuned to get a good response from the filter.

$$\begin{aligned} \mathbf{Q}_k &= E\left(\boldsymbol{v}_k \boldsymbol{v}_k^T\right) \\ &= E\left(\begin{bmatrix} \frac{\Delta t^2}{2}\ddot{x}_{k-1} \\ \Delta t \cdot \ddot{x}_{k-1} \\ \varepsilon \end{bmatrix} \begin{bmatrix} \frac{\Delta t^2}{2}\ddot{x}_{k-1} & \Delta t \cdot \ddot{x}_{k-1} & \varepsilon \end{bmatrix}\right) \\ &= \begin{bmatrix} \frac{\Delta t^4}{4}\sigma^2_{acc} & \frac{\Delta t^3}{2}\sigma^2_{acc} & 0 \\ \frac{\Delta t^3}{2}\sigma^2_{acc} & \Delta t^2 \cdot \sigma^2_{acc} & 0 \\ 0 & 0 & \sigma^2_{accbias} \end{bmatrix} \end{aligned} \qquad (4.42)$$

The measurement noise $\boldsymbol{R}$ also had to be estimated. Since only the elevation was measured directly, the measurement noise was simply a scalar defined as

$$R = E[\mathbf{w}_k \mathbf{w}_k^T] = var(w_k) = var(w),$$

where *w* once again was assumed to be constant. The value of *w* was calculated by keeping the sensor stationary and then calculating the variance as in Section 3.7.

4.4 Error estimation to a point on the sea surface

Previous sections have made it possible to estimate two sources - tilt and elevation - which make the thermal camera direction deviate from the direction in which the operator has set.

In Figure 4.4, a ship is at its equilibrium. Initially, the operator is looking at point A on the surface of the sea with tilt α on the thermal camera. In this thesis, three cases in which disturbances are acting on the ship were investigated: 1) elevated, 2) tilted or 3) elevated and tilted. Below are the two latter cases presented since the first one is covered in the last case.

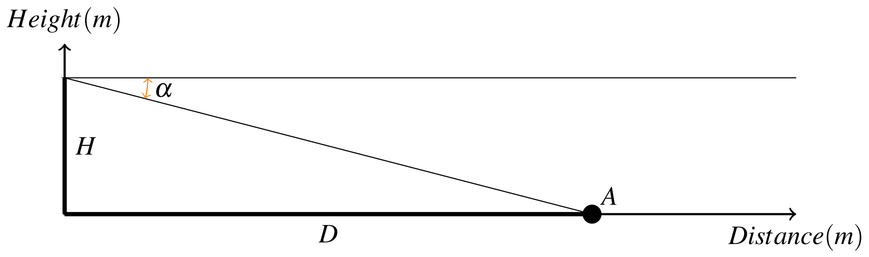

Figure 4.4 Illustration for the case when a ship is at its equilibrium. The searchlight is at height *H* above water and the operator has tilted the thermal camera by α, thus looking at point ***A*** located at a distance *D* from the ship.

The objective is to always keep the camera pointed at point A and to do this, one has to estimate the distance *D* along x-axis from the searchlight to point A. This was done using trigonometric functions, since the height above water of which the searchlight is mounted, *H*, was assumed to be known beforehand. One can then use tangent to determine the distance, since

$$tan(\alpha) = \frac{H}{D} \iff D = \frac{H}{tan(\alpha)}$$

The following sections will estimate the deviation from this point, either from tilt alone or the tilt and elevation.

Error estimation to a point (tilt compensation)

When only compensating for tilt, the elevation was assumed to be at its equilibrium as in Figure 4.5. The idea is simple. If the ship is tilted by β, which is computed using the Kalman filter in Section 4.1, then the stepper motor has to compensate by turning in the opposite direction to keep the thermal camera directed at point A.

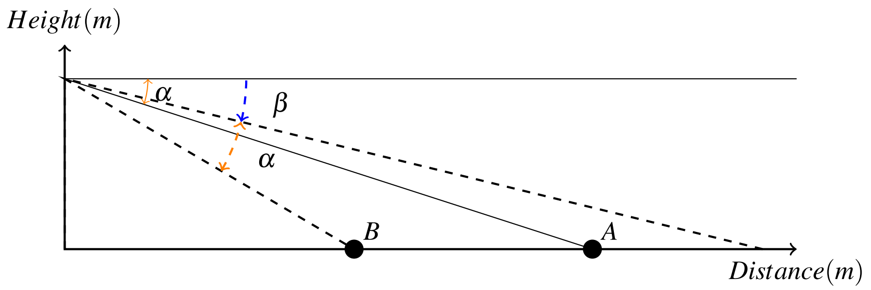

Figure 4.5 A ship is tilted by β and the tilt angle for the thermal camera, α, stays the same. If the tilt is not compensated for, the thermal camera would be looking at point B, whereas it should compensate and look at point A.

Error estimation to a point (tilt and elevation)

In this section, both tilt and elevation will be considered when calculating the error, as in Figure 4.6.

Before computing the error to point A, the distance to said point is estimated as

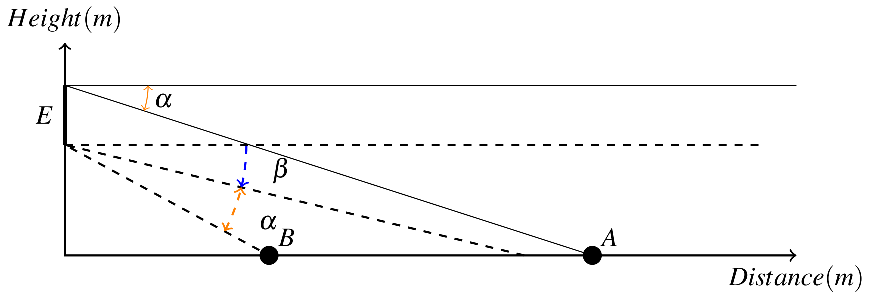

Figure 4.6 Ship is both tilted by β and elevated by E, while the tilt of the thermal camera, α, stays the same. Unless the disturbances are compensated for, the thermal camera will be looking at point B.

$$D = \frac{H}{tan(\alpha)}$$

where D is the length from the ship to point A along the x-axis - or rather, along the surface of the sea - H is the height above water for the searchlight at equilibrium and α is the tilt of the thermal camera.

Example:
The searchlight is mounted 32m up and the operator sets $\alpha = 4°$. According to the equation above, $D \approx 457.62$m. $\triangle$

The elevation of a ship can be compensated for by using the distance computed above. One can again make use of the trigonometric function for tangent as in Equation 4.43, where the new tilt needed, α_{new}, to look at point A when elevated by a distance E is computed.

$$tan(\alpha_{new}) = \frac{H+E}{L} \iff \alpha_{new} = atan\left(\frac{H+E}{L}\right) \tag{4.43}$$

Example (continued):
From equilibrium, a ship is risen 2m. The new desired tilt of the thermal camera is $\alpha_{new} = atan\left(\frac{32+2}{457.62}\right) \approx 4.25°$. Therefore, the thermal camera needs to have a tilt of 4.25° to still look at point A when elevated. $\triangle$

However, the calculations above does not compensate for tilt. To compensate for tilting, one simply compensates the thermal camera in the opposite direction of the tilt. Thus, the tilt needed for the thermal camera, γ, to compensate for both elevation and tilt was done by subtracting the new desired tilt, α_{new} with the tilt of the ship, β, as Equation 4.44.

$$\gamma = \alpha_{new} - \beta \tag{4.44}$$

Examle (continued):
A ship is assumed at one instant to tilt by $-5°$. Since the new desired tilt is $\alpha_{new} = -4.25°$, the final tilt of the thermal camera is computed as $\gamma = \alpha_{new} - \beta = -4.25° - (-5°) = 0.75°$. Therefore, for the thermal camera to look at point A when both elevated and tilted, it has to have a tilt of 0.75°. $\triangle$

In summary, the algorithm for computing the error of which the thermal camera is deviating from point A with regards to both elevation and tilt is done in following order:

1. Read the tilt α set by the operator.
2. Calculate distance D to point A by using $D = \frac{H}{tan(\alpha)}$.
3. Read the elevation E.
4. Calculate the new desired tilt, $\alpha_{new} = atan\left(\frac{H+E}{D}\right)$.
5. Read the tilt of the ship β.
6. Calculate needed tilt of thermal camera, $\gamma = \alpha_{new} - \beta$.

The value of γ was used for finding the error e, defined as the difference between the needed tilt γ and the current tilt of the stepper motor, $\alpha_{stepper}$. However, the output from the algorithm is in degrees while the stepper motor calculates the tilt as number of steps. Thus, one has to convert the tilt into integer values. Conversion from steps to tilt is done in Equation 4.45, which comes from the stepper motor being able to turn from -20° to +20° which corresponds to 0 and 600 steps respectively.

$$\alpha_{stepper} = -20 + steps \cdot \frac{40}{600} \tag{4.45}$$

The equation below was used for converting the needed tilt from degrees to steps. As seen in the equation, the second term is rounded since *steps* has to be expressed as an integer value. However, rounding means there will be *quantization error*.

$$steps = 300 + round\left(\gamma \cdot \frac{600}{40}\right) \tag{4.46}$$

In Figure 4.7, the black line is the tilt in degrees and the blue line is the converted value of *steps*, and the quantization error is the difference between the two lines. In this application, a step is equal to $\frac{40}{600} \approx 0.067°$ and the quantization error will be equal or less than half of this value, i.e., 0.033°. One can see that the quantization error is very small and not noticeable for the operator. Considering the other errors mentioned so far in this thesis, the quantization error can be considered to be negligible.

Summary for the estimations

In Figure 4.8, all previous sections are combined in a flow chart to get an overview of the whole system which combines all sensor measurements.

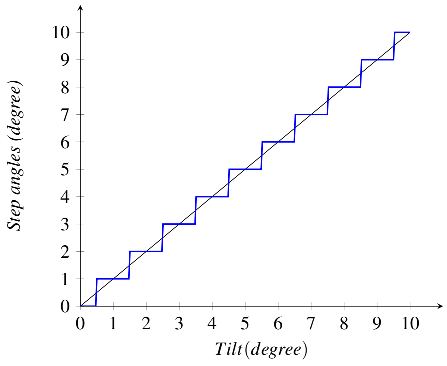

Figure 4.7 Illustration of the quantization error (blue line) caused when converting the needed tilt into discrete values.

Two Kalman filters are working parallel with each other, where one estimates the tilt using accelerometer and gyroscope measurements and the other estimates the elevation using accelerometer and pressure measurements.

The measurements from accelerometer and gyroscope are also used in Madgwick's algorithm to yield the orientation so one can extract the vertical linear acceleration from the accelerometer.

The tilt and elevation estimations are sent to the PID control which includes the error estimation in this section.

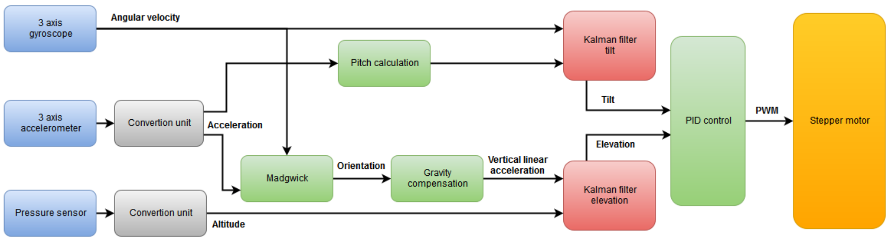

Figure 4.8 System overview over the system.

5
Control

The control of the stepper motor was done by comparing the desired tilt of the thermal camera and the current tilt of the stepper motor. This produced the error e, which was then used for computing the control output u. The control output in this thesis controlled the step time of the stepper motor, i.e. the time between two steps.

A bigger error usually leads to a larger output, e.g., a greater voltage for a DC-motor that has to run quicker. In this thesis, however, the step time needs to decrease as the error increases for the stepper motor to run faster with a bigger error.

5.1 Theory

The proposed controller, Proportional Integral Derivative (PID), is one of the most used controller today thanks to its simplicity while still functioning great for many processes. The idea is to minimize the error e, which is the difference between the setpoint r and the measured output y, i.e. $e(t) = r(t) - y(t)$. In this thesis, the setpoint was the desired tilt while the measured output was the current tilt of the stepper motor, both of which were expressed as integer values [Hägglund, 2012]. The whole system is shown in Figure 5.1, and the PID controller is described in more details below.

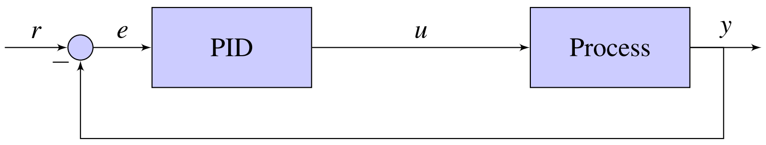

Figure 5.1 PID control which is used for controlling a process. In this case, the process is the stepper motor and the control signal determines the step time of the stepper motor.

Proportional, P

This term uses a scalar gain K_p on the error and is defined as below.

$$P = K_p \cdot e(t)$$

A smaller value on K_p will lead to a larger steady-state error and in general a slower convergence rate. On the other hand, if $K_p \rightarrow \infty$ then the controller will act as an on/off-controller, which in this case means the stepper motor will always be in *running* mode with shortest possible step time. This will wear out the stepper motor quicker, as well as generating a lot of heat. Therefore, too high a value of K_p is not desirable [Wittenmark et al., 2012][Hägglund, 2012].

Integral, I

This term uses an integral to compute the accumulated error over a set time t as

$$I = K_i \int_0^t e(\tau) d\tau$$

where the sum is multiplied by the integral constant K_i. By using the I-term together with P, one can eliminate the steady-state error as well as making the output reach the setpoint faster. However, the error might accumulate and become very large over time which will cause a delay in the output, causing it to overshoot. This can easily be prevented by implementing an anti-windup [Wittenmark et al., 2012][Hägglund, 2012].

The integral term was not used in this thesis since there was no feedback from the stepper motor. Furthermore, since the stepper motor worked with integer values and the fact that it was assumed that the stepper motor did not miss a step, there was no steady-state error to eliminate in the first place, making the integral term otiose.

Derivative, D

This term uses the derivative of the error as

$$D = K_D \cdot \frac{d}{dt} e(t)$$

with the constant derivative gain K_D. By using the derivative of the error one can prevent overshooting, thus increasing the stability of the controller. However, this term might amplify noise which can cause some problems if filtering is not used correctly on the signal. Moreover, it can in some cases slow down the process since this term suppresses fast changes, resulting in a slower converge to a setpoint than it would without the D-term [Hägglund, 2012][Wittenmark et al., 2012].

PID-controller

Combining each term into one controller, one can compute the output u as the equation below. However, any combination of the three terms can be chosen to fit a process, e.g., PI or PD.

$$u(t) = \underbrace{K_p e(t)}_{P} + \underbrace{K_i \int_0^t e(\tau)d\tau}_{I} + \underbrace{K_D \frac{d}{dt} e(t)}_{D} \tag{5.1}$$

5.2 Additional controller implementation

In theory, a PID-controller can be implemented directly in code. However, some aspects have to be considered when implementing a controller, such as saturation, integral windup and derivative filtering [Wittenmark et al., 2012].

Saturation

The output of a controller is generally assumed to be unbounded, i.e., $u(t) = (-\infty, +\infty)$. However, real actuators are almost always limited in some way. Such is the case in this thesis, where the stepper motor is the actuator and the output is the step time. A stepper motor becomes unreliable at high speed, and to ensure that the stepper motor did not miss a step the shortest step time was set to 1ms [*28SH32-0674A*]. Furthermore, the upper limit was set to the sample time of the estimations for tilt, i.e., 10ms. Therefore, the output signal is saturated as $u(t) = [1, 10]$.

Integral windup

When the output signal is saturated the integral term will accumulate, causing a large overshoot once the output signal is unsaturated. To avoid this problem, an anti-windup method has to be used, such as in [Wittenmark et al., 2012] or [Passino and Quij, 2002]. In this thesis, however, anti-windup is not needed since the stepper motor works in absolute values with no error in the position, unless a step is missed. But since no feedback is involved, there is no way of knowing if, when and how many steps the stepper motor has missed. Still, an anti-windup method was implemented if an encoder is used for feedback in the future.

Derivative filtering

A problem with using the derivative term is that the term can become very large due to quick changes, both from high frequency noise and large setpoint changes. One can avoid using the derivative term on high frequency noise by applying a first order LPF as in e.g., [Passino and Quij, 2002], where high frequencies are suppressed.

Furthermore, large setpoint changes can also be suppressed by using a scale factor on the setpoint so that only a fraction of the signal acts on the control signal [Wittenmark et al., 2012].

PID on discrete form

The above PID-controller had to be converted into a discrete controller since it was used in an embedded system. This was done by transforming the system in Equation 5.1 into the Laplace domain as in Equation 5.2, where a first order LPF was used on the derivative term to avoid applying derivation on high frequency noise. The LPF depends on the variable N, called the *maximum derivative gain*, which often is chosen as a value between 3-20 [Wittenmark et al., 2012].

$$U(s) = \Big(\underbrace{K_p}_{P} + \underbrace{K_i \frac{1}{s}}_{I} + \underbrace{K_D \frac{N}{N\frac{1}{s}}}_{D}\Big) E(s) \tag{5.2}$$

Each term was approximated using the sampling time h, with the P-term already represented in discrete form.

To approximate the I-term, Forward-Euler was used - that is, $s \approx \frac{z-1}{h}$ - which when inserted in the I-term gives the new term as $K_i h \frac{1}{z-1}$.

The last term was approximated using Backward-Euler, which was done by using $s \approx \frac{z-1}{hz}$. Thus, the new D-term is $K_D \frac{N}{1+Nh\frac{z}{z-1}}$. Combining the three discrete terms above forms the discrete PID-controller as in Equation 5.3 [Wittenmark et al., 2012].

$$U(z) = \Big(\underbrace{K_p}_{P} + \underbrace{K_i h \frac{1}{z-1}}_{I} + \underbrace{K_D \frac{N}{1+Nh\frac{z}{z-1}}}_{D}\Big) E(z) \tag{5.3}$$

6

Simulation

Simulations were performed using Simulink, which is an extension of MATLAB, due to having plenty of tool-boxes that could be used. In this chapter, each section will explain each part of the simulation such as waves, ship and stepper motor. Furthermore, models and assumptions previously made are tested and verified in this chapter. The performance of the elevation and tilt estimation, as well as the control of the stepper motor, was also tested and verified.

6.1 Waves

As described in Section 3.1, some assumptions about the waves were made and used for the purpose of simulation. During simulations, the sea was considered to either contain a single sine wave or two sine waves with different speed and amplitude. Furthermore, the following was assumed:

1. The height of the wave(s), A, was known.
2. The wave length, λ, was computed so that A/$\lambda \ll 1$.
3. The speed of a wave was calculated as $c = \sqrt{\frac{g \cdot \lambda}{2\pi}}$.
4. Elevation was given as a sine wave, $y = A \cdot sin(\frac{2\pi c}{\lambda} t)$.
5. Velocity was given as $\dot{y} = \frac{2\pi c}{\lambda} A \cdot cos(\frac{2\pi c}{\lambda} t)$.
6. Acceleration was given as $\ddot{y} = \frac{4\pi^2 c^2}{\lambda^2} A \cdot sin(\frac{2\pi c}{\lambda} t)$.

The height of a wave was set manually before a simulation using the significant wave height, H_s, which differs depending on where in the world one is looking and what the conditions are at that place [Ainsworth, 2006][*Ocean Wather Inc.*] By using and evaluating the data from [*Ocean Wather Inc.*] the wave height was decided to range from 0m $\leq$ A $\leq$ 8m.

6.2 Ship

According to [Perez, 2005], it can be hard to model a ship due to many different parameters of both the sea and the ship itself. To simplify simulations, a ship was modeled in such a way that it follows the motion of the sea, meaning a ship was modeled to have the same elevation, velocity and acceleration as the waves defined in above section.

Moreover, the tilt of the ship was needed. This was computed by looking at the derivative of the height of the wave at one instant and then use a fraction of the derivative to form the tilt. Thus, at a crest/trough of a wave the ship will not be tilting, while halfway up/down a wave the tilt of the ship is at its maximum/minimum.

6.3 Sensors

When simulating the accelerometer measurements, the acceleration $\ddot{y}$ of the ship was used but with added noise. The noise added to the measurements were both to simulate the measurement noise from the accelerometer itself and the vibrations from the ship acting on the accelerometer.

Simulation of the pressure sensor measurements was done in the same way, and uses the elevation y of the ship. As for the accelerometer, the pressure sensor measurements include added noise to simulate e.g., measurement errors.

The noise for each sensor was based on the variance of the measurements, which was found by taking a number of samples and then computing the variance for each sensor when stationary. This variance was the baseline needed for simulation. In addition to this, some noises were added to simulate errors such as vibrations.

6.4 Stepper motor

There was a predefined model of a stepper motor in the Simulink library which used a number of parameters, most of which were described in the datasheet of the stepper motor. However, some parameters were not included, such as *maximum detent torque*, *maximum flux linkage*, *total inertia* and *total friction*. Those had to be approximated, which was done by performing empiric experiments on the real stepper motor and then tune the parameters in the model until the simulated stepper motor matched the performance of the real stepper motor [*Stepper motor model*].

The rotational speed of the stepper motor was computed as $v = \frac{v_{max}}{T}$, where v_{max} was the maximum rotation speed defined as the maximum number of steps each second and T was the step time. Since the shortest step time was 0.001s and each step was equal to $\frac{40}{600}°$, the maximum rotation speed for the stepper motor was approximately 66.67°/s. Thus, if the tilt rate of a ship is greater than the maximum rotation speed, then the stepper motor will be saturated.

6.5 Simulink blocks

The figure below shows the idea of the simulation process. Waves, either a single sine wave or two sine waves, are generated in the Wave(s)-block. The output of this block is the elevation, velocity and acceleration of the waves.

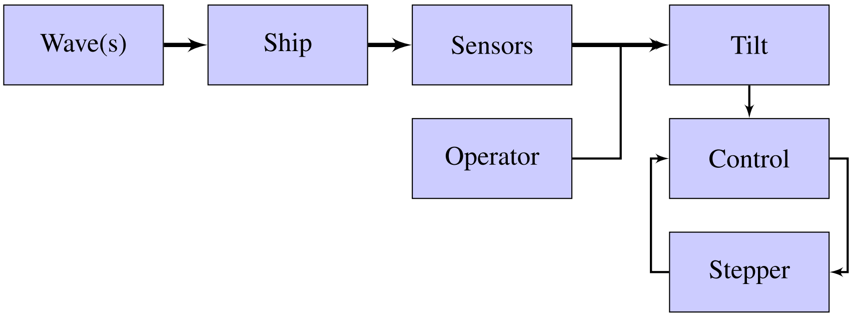

Figure 6.1 Overview of the simulation process in Simulink. Thick arrow means more than one value is passed to the next block.

The output from the Wave(s)-block was the input to the Ship-block. The ship was set to follow the motion of the waves proportionally, i.e., the ship had the same elevation, velocity and acceleration as the waves. Furthermore, the tilt of the ship was computed as a fraction of the derivative of the elevation at each instant.

The Sensors-block took three variables as its inputs: elevation, acceleration and tilt. In this block, noises were added to each variable to simulate errors in the measurements, such as vibrations.

The desired tilt for the thermal camera was computed in the Tilt-block using the sensor measurements together with the setpoint from the operator. The desired tilt was then sent to the Control-block which calculated the error between the desired tilt and the current tilt of the stepper motor. In this block the direction, either up or down, and the step time of the stepper motor was set. These two variables were then set to the Stepper-block, where a step was taken in the direction set by previous block. The current tilt of the stepper motor was then sent back to the Control-block, thus creating a loop between the Control- and Stepper-block.

7 Experiments

Experiments presented in this and following chapters are divided into two groups; one for simulations and one for the real process.

7.1 Simulations

The idea of doing simulations was to be able to validate, verify, tune and evaluate the system before doing tests on the real system. In this thesis, simulations were run and tested for two scenarios - calm and rough sea. Furthermore, each scenario was performed using either one wave or two waves.

Simulations were done to test for both tilt and elevation estimation. Elevation estimation was done for both methods presented earlier, i.e., either using only the accelerometer or a Kalman filter for both accelerometer and pressure sensor.

Finally, the control of the stepper motor was tested to see how well it performed. Two cases were evaluated - one where only tilt was compensated for and one where both tilt and elevation was compensated for.

7.2 Real system

To test the real system, all sensors were attached to one end of a long stick as seen in Figure 7.1. The stick could then be raised and lowered to mimic waves with a wave height of up to 2m.

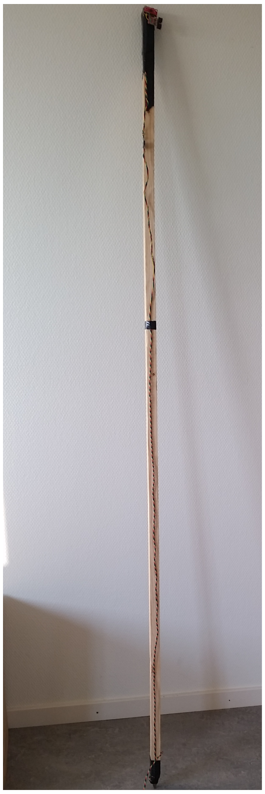

Figure 7.1 Sensors are mounted at the top of the stick. The stick is approximately 2m long.

8

Result

In this chapter, the results from the experiments, both from simulation and real process, are presented.

8.1 Simulations

Four types of tests were simulated, and those are:

1. Kalman filter for tilt estimation.
2. Elevation estimation with only accelerometer.
3. Kalman filter for elevation estimation.
4. Control loop for stepper motor.

Kalman filter for tilt estimation

In this part, the accelerometer measurements θ_a' were modeled to have quite high noise. This was done to mimic vibrations and the distortion in the measurements due to the linear accelerations acting on it and the fact that rotation from sensor to Earth frame is not perfect.

The gyroscope, on the other hand, does not have as much noise since it is not as sensitive to vibrations. However, the gyroscope suffers from angle random walk, thus a bias was added to simulate this error.

The measurements from the sensors were fused together using a Kalman filter, and the resulting estimation of the tilt can be seen in Figure 8.1. This test was done to see how well the tilt estimation could follow a reference signal. As seen in the figure, the estimation overshoots but with merely one degree. Furthermore, over time the estimation follows the reference signal well and deviates with less than half a degree.

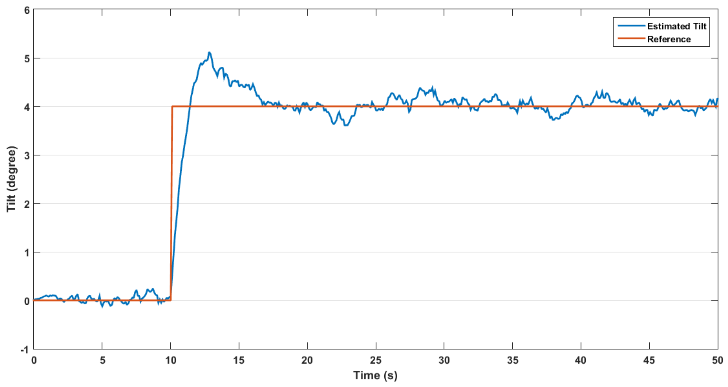

Figure 8.1 Tilt estimation based on accelerometer and gyroscope measurements using a Kalman filter.

Elevation estimation with only accelerometer

The values from the accelerometer were assumed to be given as vertical linear acceleration as computed in Section 4.2. Furthermore, the vertical linear acceleration was also assumed to be equal to the acceleration of the waves. Below are the results from simulations using both one and two sine waves, where both calm and rough sea were tested for in both cases.

One sine wave

Simulations for calm sea were performed with a wave height of 0.5m, while for rough sea the wave height was set to 4m.

The height estimation for calm sea can be seen in Figure 8.2, where the estimation error can be seen to be less than 0.1m. However, the elevation estimation started to drift after some time, as in Figure 8.3. Luckily, the drift decreased when simulated for an even longer period, which was due to the feedback which brought the estimation back to equilibrium.

For rough sea, the estimations are as in Figures 8.4 and 8.5, where the estimation error was approximately 1m. As seen in Figure 8.5, the estimation did not drift even after it had been running for a while. However, the elevation was instead underestimated.

Two sine waves

To simulate two sine waves during calm sea, one wave was set with a wave height of 0.25m and the other was set to 0.5m. As seen in Figure 8.6, the elevation was

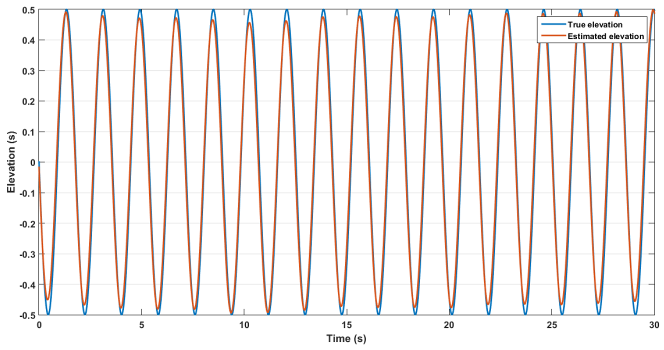

Figure 8.2 Estimation using only accelerometer at calm sea with 0.5m wave height. The elevation estimation is somewhat underestimated.

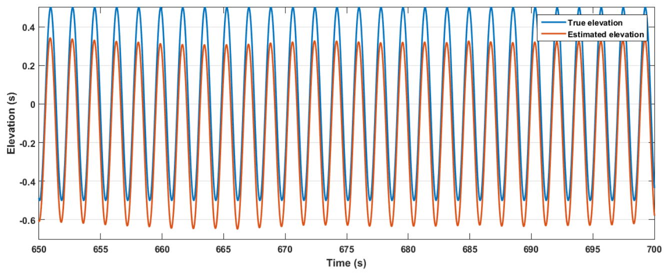

Figure 8.3 Estimation using only accelerometer at calm sea with 0.5m wave height. Drift causes the elevation estimation to deviate from the true elevation.

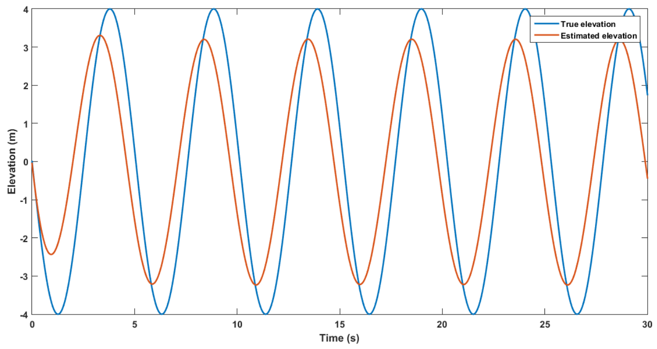

Figure 8.4 Estimation using only accelerometer during rough sea with 4m wave height. The estimation is underestimated by approximately 1m.

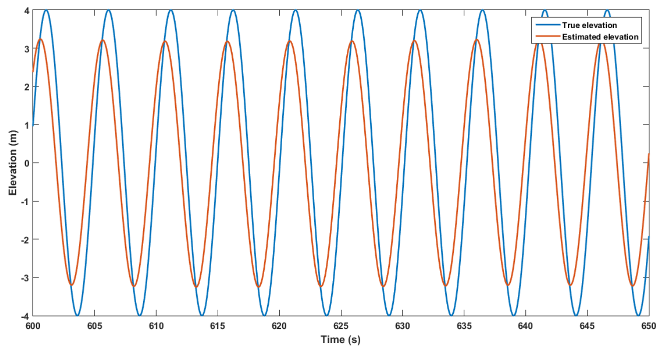

Figure 8.5 Estimation using only accelerometer during rough sea with 4m wave height. There is no drift in the estimations. However, the elevation is underestimated.

estimated fairly good with minor errors at the crest and through. However, as seen in Figure 8.7, the estimation started to drift as time progressed. Although returning to the equilibrium eventually, this drift might still cause problems when stabilizing the thermal camera.

To simulate rough sea, one wave was set with a wave height of 3m and the other to 5m. The resulting simulation is shown below in Figures 8.8 and 8.9. From the first figure, one can see that the estimation at crest and trough is off by as much as 2m. While the elevation was underestimated during the whole simulation, it did not drift at all as seen in Figure 8.9.

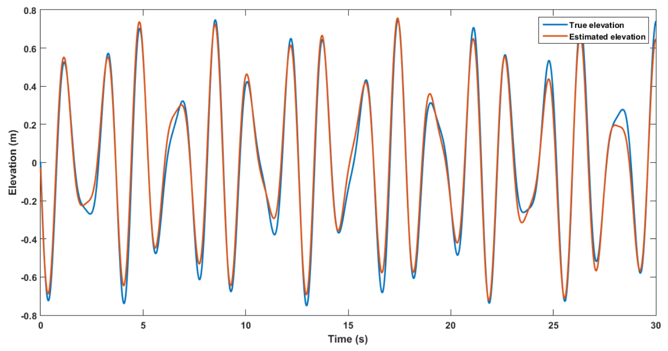

Figure 8.6 Estimation using only accelerometer during calm sea. It is simulated using two waves with wave height 0.25m and 0.5m respectively.

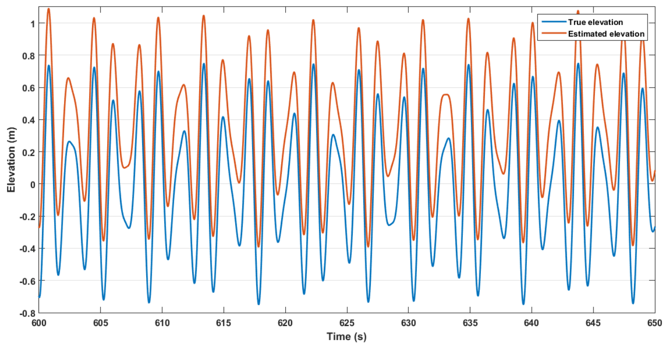

Figure 8.7 Estimation using only accelerometer during calm sea. It is simulated using two waves with wave height 0.25m and 0.5m respectively. Drift causes the elevation estimation to deviate from the true elevation.

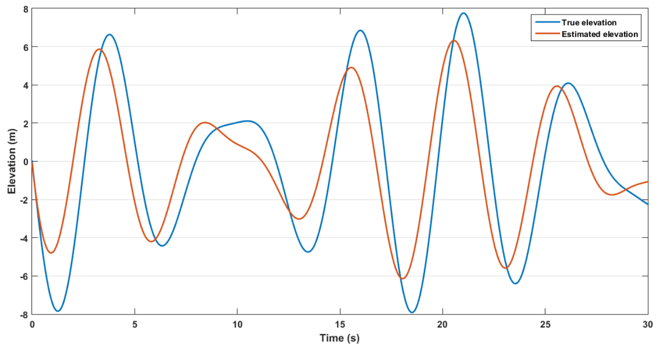

Figure 8.8 Estimation using only accelerometer during rough sea. It is simulated using two waves with wave height 3m and 5m respectively. The elevation is underestimated.

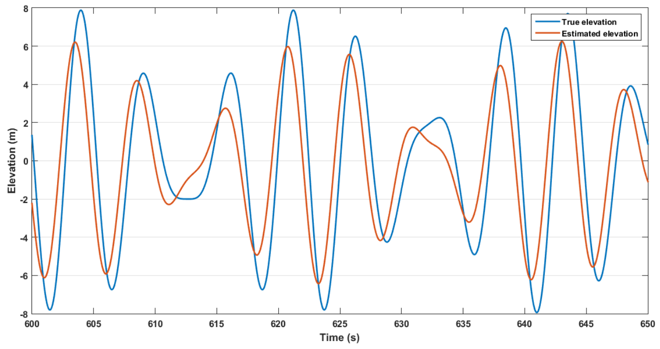

Figure 8.9 Estimation using only accelerometer during rough sea. It is simulated using two waves with wave height 3m and 5m respectively. The elevation is still underestimated.

In summary, the elevation estimation highly depended on how much feedback was used to bring the estimation back to equilibrium. The feedback was implemented as a constant value and the value for the feedback was a trade-off between how much drift to suppress against how accurate the elevation was estimated. As seen in the above figures, the chosen feedback did a fairly good job at estimating the elevation for small waves, but was prone to drift. Even though drift was reduced over time, it might still cause problems for the control of the stepper motor if the estimation is too far off. For larger waves, the drift was eliminated but at the cost of underestimating the elevation.

Kalman filter for elevation estimation

The Kalman filter in this section used the measurements from both the pressure sensor and accelerometer to estimate the elevation of the ship. The acceleration measurements from the accelerometer were assumed to be given as vertical linear acceleration in Earth frame. Furthermore, the same tests were performed as in previous section.

One sine wave

The resulting simulation for calm sea using only one sine wave with a wave height of 0.5m can be seen in Figure 8.10. It took the Kalman filter a few seconds before the estimation converged to the true elevation. Once it had converged the filter did a fair job at estimating the elevation, considering the high variance of the pressure sensor measurements. As time progressed, the estimation error for the crest and trough was still present and were both over- and underestimated, as in Figure 8.11. However, unlike the previous method using only the accelerometer, this method did not drift.

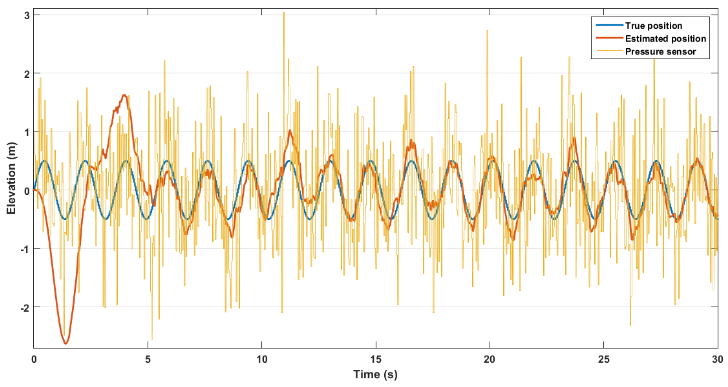

Figure 8.10 Estimation using Kalman filter for one sine wave with a 0.5m wave height.

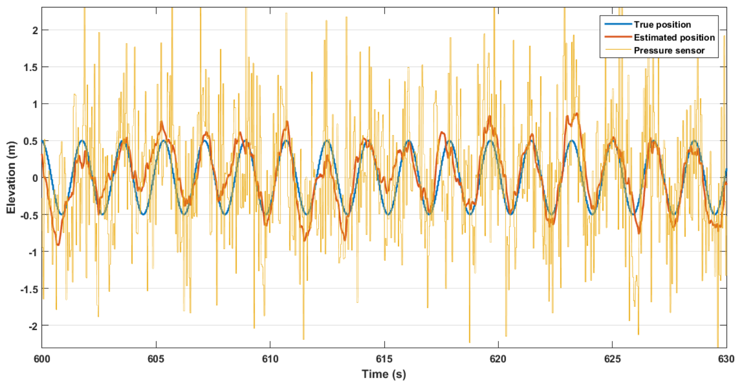

Figure 8.11 Estimation using Kalman filter for one sine wave with a 0.5m wave height after 600 seconds.

Simulation for rough sea was done with a wave height of 4m, and is shown in Figure 8.12. Again, it took the Kalman filter a few seconds in the beginning before converging to the true elevation. When run for a longer period, as in Figure 8.13, it was clear that the estimated elevation was much more accurate during rough sea than calm sea.

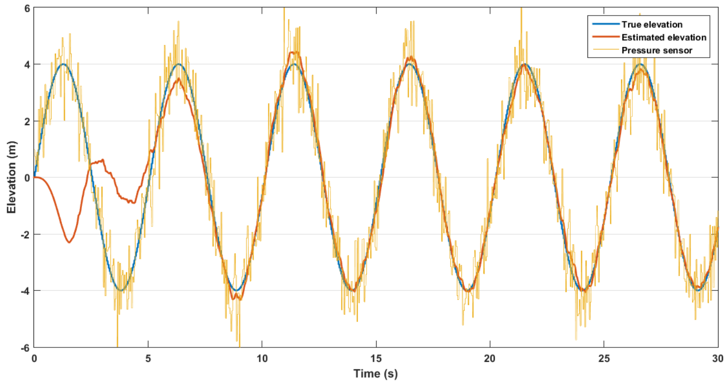

Figure 8.12 Estimation using Kalman filter for one sine wave with 4m wave height.

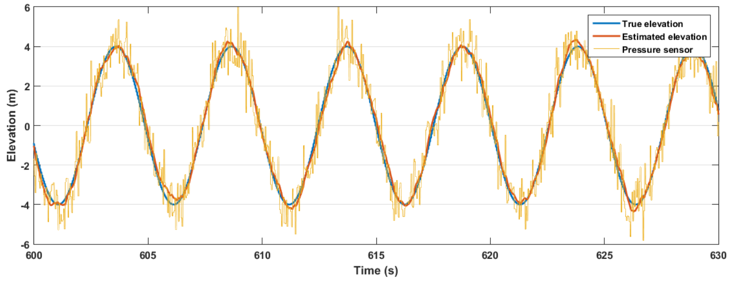

Figure 8.13 Estimation using Kalman filter for one sine wave with 4m waves after 600s.

Two sine waves

At calm sea, one wave was set to 0.5m and the other to 0.25m. Simulation with this setup can be seen in Figure 8.14 below. The estimation during calm sea still contained errors at crest and trough. Even after a few minutes, see Figure 8.15, the estimations at the crest and trough were still a bit off.

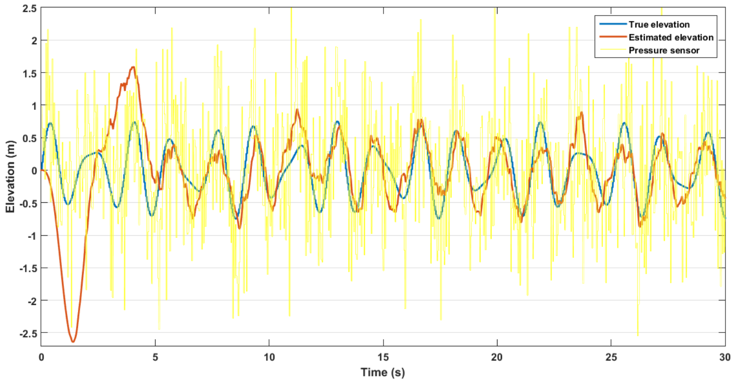

Figure 8.14 Estimation using Kalman filter during calm sea with two waves of 0.25m and 0.5m wave height respectively.

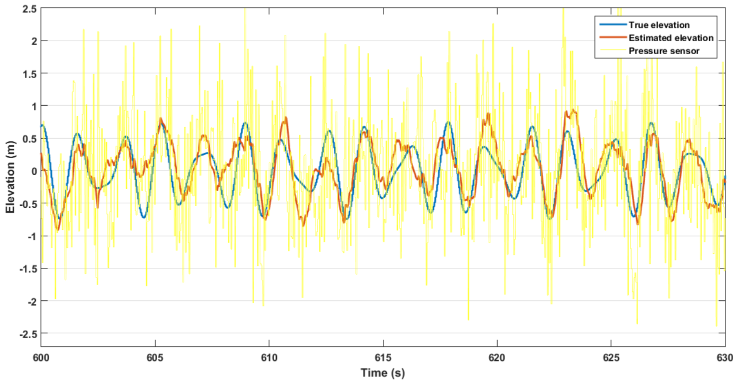

Figure 8.15 Estimation using Kalman filter during calm sea with two waves of 0.25m and 0.5m wave height respectively, after 600s.

When rough sea was simulated with two sine waves, the wave height was set to 3m and 5m respectively. As seen in Figure 8.16, the estimation converged fairly quickly to the true elevation and did a very good job at estimating the true elevation. Even after the simulation had been run for a couple of minutes, the estimated elevation was very close to the true elevation, as seen in Figure 8.17.

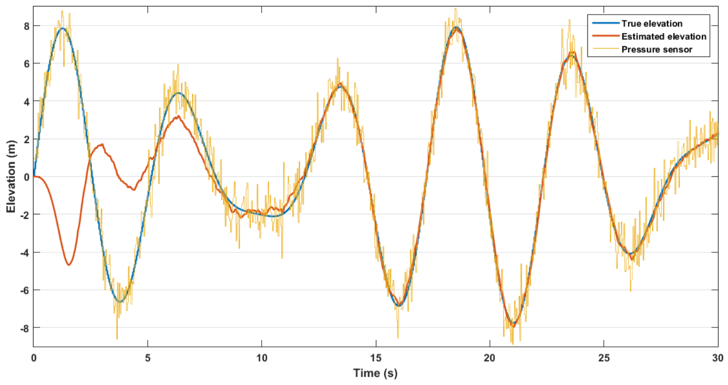

Figure 8.16 Estimation using Kalman filter during rough sea with two waves of 3m and 5m wave height respectively.

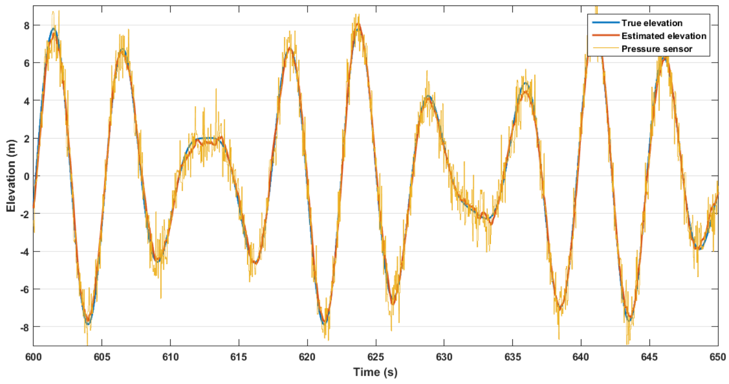

Figure 8.17 Estimation using Kalman filter during rough sea with two waves of 3m and 5m wave height respectively, after 600s.

In summary, the Kalman filter did a better job at determining the elevation during rough sea with bigger waves compared to calm sea with smaller waves. This is due to the trade-off for the Kalman filter, which is to either be robust to disturbances but having a slow convergence rate or a fast convergence rate but more sensitive to disturbances. For this thesis, more emphasis was put into a more robust filter.

Control loop for stepper motor

The previous sections show how well the estimation was for tilt and elevation, both of which were needed for controlling the stepper motor. In this section, the control of the stepper motor was simulated for two cases: compensation for tilt alone or compensation for both tilt and elevation. The ship was modeled to follow the motion of the sea proportionally.

Tilt compensation

Tilt compensation is merely about comparing how much the ship is tilting and compensating by turning the stepper motor in the opposite direction of the tilt. For example, if the ship at one instant is tilting -10°, the stepper motor needs to compensate by having a tilt of +10°.

In Figure 8.18 below, the simulation was performed for a ship where the searchlight was set at a height of 5m above water and the tilt set by the operator was 0°. As seen in the figure, the stepper motor could not handle quick and large changes on the desired tilt. For example, at 1s the desired tilt changed from about 440 steps ($\sim$29.3°) to 160 steps ($\sim$10.7°), which the stepper motor could not compensate perfectly for. However, one has to consider the fact that the tilt varies as approximately $\pm$9°, which is a fairly high value.

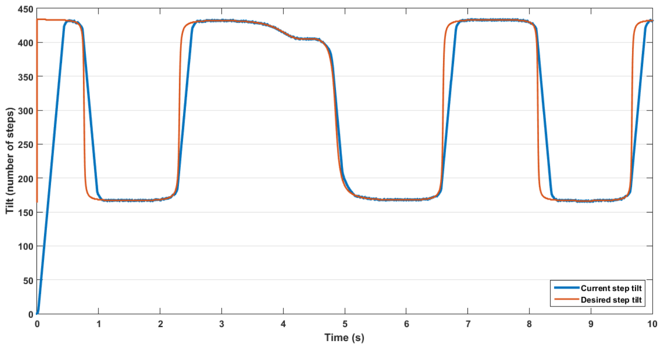

Figure 8.18 Comparison between the current step tilt (blue) and the desired step tilt (orange). The stepper motor was not fast enough to compensate for swift tilt changes.

Tilt and elevation compensation

In this case, the searchlight was again mounted 5m above water. Moreover, the tilt set by the operator was changed to -5° to more clearly see how elevation affects the desired step tilt. Elevation was modeled using two sine waves with wave height of 1m and 2m respectively. As seen in Figure 8.19, the desired step tilt looks different than Figure 8.18 since elevation was taken into consideration when calculating how much the thermal camera had deviated. One can see that the stepper motor followed the desired step tilt well, except when the value of the desired step tilt changed rapidly.

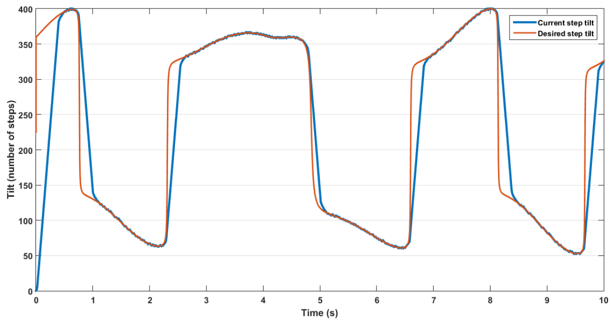

Figure 8.19 Comparison between the current step tilt and the desired step tilt. The stepper motor follows the desired step tilt well, except when the tilt changed rapidly.

8.2 The real system

To mimic the motion of waves for the real system, the sensors were mounted on top of a long stick, see Figure 7.1. The end of the stick was then moved up and down to simulate the wave motion. At stationary, it sat on top of a desk 0.8m above the floor. This was the reference height which was used to see if and by how much the elevation estimation deviated from a known height. Furthermore, the distance from desktop to ceiling was approximately 1.5m.

Elevation estimation was performed for both methods presented in this thesis, i.e., accelerometer only and Kalman filter, to see the differences between said methods. As seen in Figure 8.20, the elevation estimation using only an accelerometer contained a bias which the Kalman filter did not suffer from. One can also see that the Kalman filter was able to estimate the elevation from desktop to ceiling satisfactorily, as well as from desktop to floor. During the first 40s, the sensors were moved in such a way to simulate larger waves. From 40s and further, the sensors were moved slightly up and down to see how well the estimations were for smaller waves.

Since the Kalman filter was able to estimate the elevation without any bias, this method was chosen when testing the performance for the control of the stepper motor. The control signal, which determines the time for succeeding step, of the stepper motor was set to have an upper limit of 20ms and a lower limit of 1ms. The upper limit is an arbitrary value, and was chosen as twice the update time from

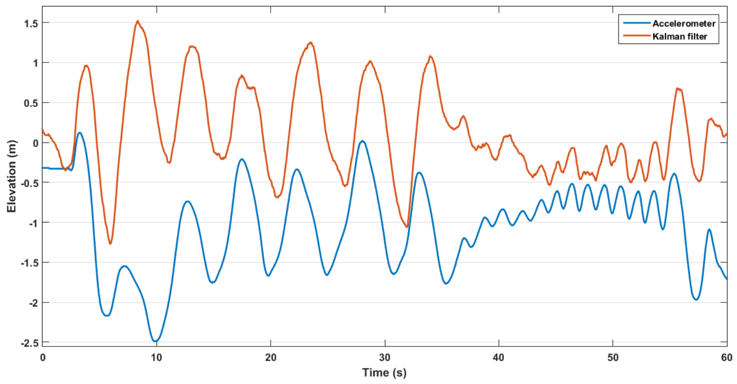

Figure 8.20 Elevation estimation using both estimation methods. The elevation has its maximum against the ceiling (1.5m) and its minimum against the floor (-0.8m).

the sensors, while the lower limit was set to ensure a step was not missed while still maintaining highest possible speed. Moreover, a bigger error corresponded to a smaller step time.

The tilt set by the operator was set to -0.5° to simulate looking for an object at sea. The result from the real test is shown in Figure 8.21. During the first 10s, the whole system was initialized.

From 10s-30s, a rolling motion of the ship from side to side was mimicked while the thermal camera was assumed to be pointed perpendicular to a ship. As seen in the figure, the control signal was saturated most of the time and did not quite compensate for the rolling motion.

From 40s-65s, a ship traveling alongside waves was mimicked, which means having a steady ascend and descend. The stepper motor was once again saturated most of the time.

In the final phase, from 70s to the end of the test, a ship traveling against waves was mimicked. Traveling against waves means a slow ascend and quick descent, resulting in slamming of a ship. Looking at the difference between the desired and actual tilt of the thermal camera, one can see that this was the worst case since the stepper motor was not quick enough to compensate for the rapid changes. Furthermore, the control signal was saturated at 1ms most of the time meaning the stepper motor was almost never in standby mode.

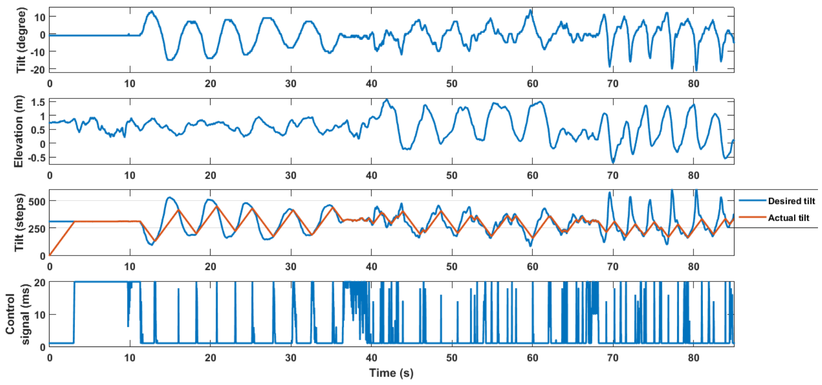

Figure 8.21 Control of a stepper motor. The elevation had its maximum against the ceiling (1.5m) and its minimum against the floor (-0.8m).

As a proof of concept, the step time was reduced from 1ms to 0ms, i.e., the stepper motor turned continuously. Furthermore, the step size was increased so that the stepper motor took 300 steps within 40°, thus reducing the resolution per step from 0.067°/step to 0.13°/step. Running the same test again gave the result in Figure 8.22. Comparing this result with Figure 8.22, it is clear that the stepper motor could follow the desired tilt much better than previously. However, the problem with this new approach is that stepper motor was running continuously, hence there is a risk of missing a step. Without any feedback of the position for the stepper motor, there is no way of telling if, when and how many steps that have been missed. The control of the stepper motor will still function, but the operator will notice that the thermal camera is not looking in the direction set by said operator. Furthermore, the operator might notice that the image is not quite smooth when the stepper motor is moving due to the increased step size.

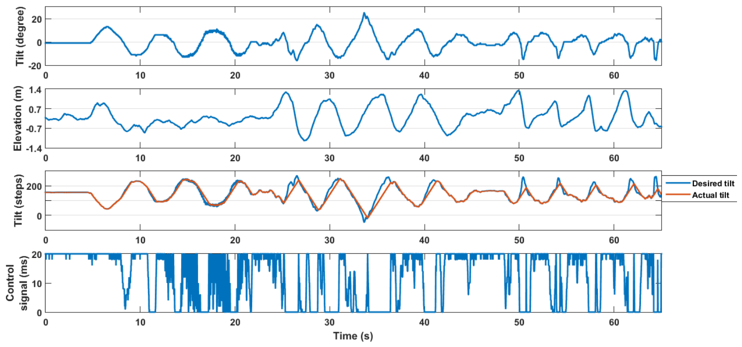

Figure 8.22 Control of a stepper motor. The elevation had its maximum against the ceiling (1.5m) and its minimum against the floor (-0.8m).

9

Discussion

The work done in this thesis was tried out on a real searchlight to see if there was any improvements for the operator. This test was, however, not included in the result since it was hard to document the visual result. Although only tilt was compensated for during the test, it was clear that the image was much better than previously. This was mostly due to the operator needing to manually tilt the stepper motor up and down to compensate for disturbances. Not only is this tedious work, but the stepper motor in the original work had a fairly large step time (5ms), thus making it turn slowly. Therefore, simply by compensating for the tilt of the ship the performance was increased significantly.

Stabilization of the thermal camera was improved further by also taking into account the elevation. Two methods to estimate the elevation has been presented in this thesis - one which only used an accelerometer, and one which used both an accelerometer and an auxiliary pressure sensor in a Kalman filter. Both methods worked to some extent, but the elevation estimation seemed to work better when using the second method, especially for eliminating drift caused by integrating the accelerometer measurements to yield the position. The two methods, however, could not estimate the elevation equally good between the cases studied, where the biggest difference seemed to be between calm and rough sea. When estimating using a Kalman filter, the estimations were much better for rough sea compared to calm sea. This was partly due to making the Kalman filter more robust to disturbances, which makes the filter converge at a slower rate. One would prefer the estimations to be good for all cases, but since it is more crucial for stabilization to work during rough sea rather than calm, more effort was put into the former.

Adding to this, one can see that during calm sea the deviation of the thermal camera caused by tilt exceeds that which was caused by elevation itself. Therefore, the estimation error for elevation is not as important during calm sea as it is during rough sea.

Elevation estimation for calm sea using only an accelerometer was shown to be good in a short period of time before it started to drift. However, as the wave height increased, the drift itself decrease. Once the wave height reached a certain value, the drift was eliminated but instead the elevation was underestimated. To prevent the elevation from being underestimated, the value of the feedback should be reduced. However, reducing the feedback will cause the estimation to drift even more during calm sea. Choosing the correct value is therefore a trade-off for when one would want the estimations to work well or poorly. This method does indeed give a rough estimation, and arguably it is better than no estimation at all.

Looking back to the sections about the sensors and the result of fusing the sensor measurements together, one can see that the noise from the sensors were fairly low, despite the sensors being in the very low price range. Since this application relies heavily on those sensors, one could argue that it would benefit to choose sensors that cost a bit more but comes with less noise. Another option is to buy two cheaper sensors and fuse the measurements together using e.g., a Kalman filter. This would yield much more accurate measurements, rather than relying on a single measurement unit.

The calibration method for the accelerometer in this thesis was done to estimate the zero-g offset and the scale factor for each axis. However, the way it was performed was not optimal. First of all, the accelerometer was placed on top of a desk which might not be perfectly flat. Furthermore, the accelerometer was held in place by hand, and by doing so some vibrations might have been induced, thus causing some errors in the calibration. This could have easily been improved by using a stand to keep the sensors perfectly still.

9.1 Estimation

A couple of different methods were developed and tried in this thesis for different estimations, such as tilt and elevation. One could argue that such an approach is not optimal (author agrees), but in the purpose of learning it is. For example, both Kalman and complementary filter were investigated, but Kalman filter was chosen for both tilt and elevation estimation. Instead, the complementary filter was indirectly used in the rotation of the sensor values.

Two different methods, Euler angles and quaternions, were also investigated for the rotation from sensor to Earth frame. Quaternions, although seemingly complex and hard to use, performed surprisingly well while being computationally inexpensive. This was not the case for Euler angles, since it uses computationally expensive trigonometric functions.

Studying the elevation estimation using Kalman filter, both in the simulation and the real process, it seemed as if the values jumped up and down by a few decimeters at crest and trough. This is due to the pressure sensor which contains very high noise. However, the jumping does not seem to be causing too much trouble when using elevation to control the stepper motor. This is partly due to the fact that elevation does not affect the deviation of the thermal camera as much as tilt. Moreover, the stepper motor itself is not quick enough to handle those quick changes in elevation.

Estimating the elevation using only the accelerometer was seen as a challenge early on, since numerical integration causes errors which leads to drift. Even though quite a few different methods were investigated, as well as using limitations and boundaries of a ship, they all fell short due to not having any external method to compensate for the drift. In the end, the drift was handled using a feedback on the position as aforementioned.

9.2 Control

As long as the value of the desired tilt did not change too quickly, the control of the stepper motor did a good job at following it. If the rate of change of the desired tilt exceeded the maximum rotational speed of the stepper motor, the control became saturated. This thesis has shown that even though the stepper motor moves more quickly than originally, it was still not sufficient to compensate for all cases one might encounter at sea. Even as the step size was increased as a proof of concept, as well as decreasing the step time to 0ms, there were still a few times where the stepper motor could not follow the desired tilt. However, the overall performance of the stepper motor far exceeded previous work and did make a huge difference for the operator.

The control of the stepper motor was performed using a simple P-regulator. Different regulators were investigated, such as PI, PD, and PID, but the P-regulator was deemed sufficient since the control was saturated most of the time. It was noticed that using the derivative term made the control slower than without it, but this term might be useful if the stepper motor is made to rotate faster. The derivative term would then make the stepper motor turn more smoothly, thus giving a better image to the operator. The reason for not using the integral term was because the stepper motor worked in absolute values and with no error in the position. Due to this, there was no steady state error for the integral to remove, thus making it obsolete.

10 Conclusions and improvements

In the original work, the control of the stepper motor was done by the operator. If, for example, the ship was tilted one way, the operator had to manually tilt the thermal camera in the other direction to compensate for this. Therefore, simply by making the thermal camera compensate for the tilt of the ship, much has been improved for the operator. This was improved further by taking into account the elevation of a ship as well.

Previously, the stepper motor could run in two modes, standby and running. If the stepper motor needed to take a number of steps, the mode was set to running before taking the required steps with a step time of 5ms. When it had reached the desired position the mode was changed back to standby. In this thesis, the aforementioned way of taking steps was changed so that the mode was only in running when a step actually was taken, otherwise it was set to standby. This was done to minimize the heat generated by the stepper motor while maintaining a short step time. As a result of this change, one could hold the stepper motor by hand without any pain due to heat, even during the real test when the stepper motor was saturated most of the time, thus generating the most amount of heat. This was not possible when the stepper motor was always kept in running between steps.

Regarding the elevation estimations and the differences between the two methods presented in this thesis, the author argues that the improved estimation from using a pressure sensor and an accelerometer in a Kalman filter was worth it for the extra compensation.

Overall, the stabilization of the thermal camera worked fairly well and did make a huge difference for the operator when comparing the performance from an uncompensated thermal camera. However, some things could be improved in future works, such as implementing image processing for absolute tracking of an object. While stabilizing the thermal camera is great when e.g., searching for an item, one could use tracking for when said item is found. Such is the case during search-and-rescue of a human. The limitations of current hardware, however, prevents the use of image processing since it requires both storage capacity for images and processing power to process each image. In future works, one could try to look into applying an external Secure Digital card (SD-card) together with an external DSP unit to do the pre-computations for the MCU.

RAO, which is a ship's transfer function, was defined but was not used due to lack of time. On the other hand, it could still be used in future work to improve simulations or be used as a model for adaptive control. Adaptive control could also be used to estimate the parameters of the ship, which may improve the response from the stepper motor. The control of the stepper motor could also be improved by looking into an Extended Kalman filter for prediction of waves, or use Model Predictive Control for the motion of a ship to improve the overall performance.

Calibrations of the sensors can also be improved in future work. While the six-point calibration used in this thesis seemed to have reduced a lot of errors, one could still use other methods, such as Gauss-Newton or auto regression, to get an even better calibration. One could also use the methods developed in e.g., [Liu and Pang, 2001] and [Park, 2004]. Furthermore, one could also try to calibrate the gyroscope to reduce the angle random walk when integrating the measurements.

One thing that was not tested for was placing the sensors inside the module. The space inside the module is limited and the sensors might end up very close to the stepper motor, which could cause problems due to the magnetic field generated by the stepper motor. Therefore, in future works it needs to be tested if errors are induced when the sensors are mounted inside the module and the stepper motor is running. Furthermore, if a pressure sensor is used it might act differently if it is mounted inside the module.

If elevation estimation is desired but without a pressure sensor, one should consider improving the method using only an accelerometer. Such improvements include, among other, changing the current constant variable for the feedback to a variable that adapts to the sea condition. One could also use a higher order integration method to reduce integration errors, and look into filtering after integration to reduce the drift in position.

As for the stepper motor, it was clear that it was not sufficiently fast to be able to compensate fully for tilt and elevation. While it was possible to make the stepper motor rotate faster by setting the step time to 0ms, it was not recommended since there was a risk of missing steps. This could, however, be fixed by using some type of feedback, such as an encoder. One could also look into upgrading the stepper motor into a quicker one, or change the gear ratio between the thermal camera and the stepper motor. If the gear ration is changed, one has to make sure that enough holding torque is available so that the stepper motor can withstand disturbances. Changing the gear ratio would be almost the same as increasing the step size of the stepper motor, as was shown as a proof of concept in the results. Increasing the step size showed that it was possible to compensate much better for both tilt and elevation, but the image to the operator was not as good since the rotation of the thermal camera was not as smooth.

Bibliography

A4982 Motor Driver (2014). 5th ed. Data sheet. Allegro.

Acheson, D. (1990). *Elementary Fluid Dynamics*. Clarendon Press, Oxford.

Ainsworth, T. (2006). *Significant wave height*. `http://mxak.org/weather/pdfs/waves.pdf`. Significant Wave Height. Accessed: 2016-03-07.

Allan, J. (1945). "The stabilisation of ships by activated fins". *Transactions of the Royal Institution of Naval Architects* **87**, pp. 123–159.

Asmussen, I., W. Menzel, and H. Mumm (2001). *Gl technology: ship vibration*. Hamburg.

Blom, G. (2004). *Sannolikhetsteori och statistikteori med tillämpningar*. 5th ed. Lund. Studentlitteratur.

BMP180 (2013). Ed. 2.5. Data sheet. Bosch.

Böiers, L.-C. (2010). *Mathematical Methods of Optimization*. 1st ed. Lund. Studentlitteratur.

Brorsson, M. (2011). *Datorsystem*. Lund. Studentlitteratur.

CalQlata. *Calqlata*. `http://calqlata.com/productpages/00059-help.html`. Accessed: 2015-10-14.

Cao, H., H. Lv, and Q. Sun (2015). "International conference on information and automation". In: *Model Design Based on MEMS Gyroscope Random Error*. IEEE. Lijiang, China, pp. 2176–2181.

CohuHD (2014). *How far can I see?* `http://www.cohuhd.com/Files/white_papers/How_Far_Can_I_See.pdf`. Accessed: 2015-09-30.

Colorlight. *Clite2*. `http://www.colorlight.com/products/searchlights/led-searchlights/clite-2.html`, digital image. Accessed: 2015-10-14.

Coombs, J. and R. Prabhu (2011). *OpenCV on TI's DSP+ARM platforms: Mitigating the challenges of porting OpenCV to embedded platforms*. Accessed: 2016-01-09. URL: `http://www.ti.com/lit/wp/spry175/spry175.pdf`.

Dam, E. B., M. Koch, and M. Lillholm (1998). *Quaternions, Interpolation and Animation*. Tech. rep. Department of Computer Science, University of Copenhagen.

DELTALINE. *28sh32-0674a*. `http://www.delta-line.com/28sh32-0674a-000-P73368.htm`, digital image. Accessed: 2015-10-02.

Diebel, J. (2006). *Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors*. MA thesis. Stanford University.

Eclipse. *Eclipse*. `https://eclipse.org/users/`. Accessed: 2016-07-14.

FLIR. *Tau 2*. `http://www.flir.com/cores/display/?id=54717`, digital image. Accessed: 2015-11-04.

FLIR (2015). 141st ed. Product Specification. FLIR.

Fossen, T. I. (2002). *Marine Control Systems: Guidance, Navigation and Control of Ships, Rigs and Underwater Vehicles*. Trondheim. Marine Cybernetics.

freeRTOS (2016). *Corporate presentation*. `http://www.realtimeengineers.com/ReferencedDownloads/Real_Time_Engineers_Ltd_FreeRTOS_Overview.pdf`. Accessed: 2016-03-21.

Freescale (2015). *Allan variance: noise analysis for gyroscopes*. `http://cache.freescale.com/files/sensors/doc/app_note/AN5087.pdf`. Accessed: 2015-10-14.

FTDI (2010). Ed. 2.5. Data sheet. FTDI chip.

GDV. *Gdv*. `https://www.containerhandbuch.de/chb_e/stra/index.html?/chb_e/stra/stra_02_03_03.html`. Accessed: 2015-10-15.

Glad, T. and L. Ljung (2007). *Reglerteori: Flervariabla och olinjära metoder*. Lund. Studentlitteratur.

Gui, P., L. Tang, and S. Mukhopadhyay (2015). "IEEE 10th conference on industrial electronics and applications". In: *MEMS Based IMU for Tilting Measurement: Comparison of Complementary and Kalman Filter Based Data Fusion*. IEEE, pp. 2004–2009.

Hägglund, T. (2012). *Reglerteknik AK: Föreläsningar*. Lund. Lunds Tekniska Högskola.

Hamilton, W. R. (2000). *Elements of quaternions*. Cambridge University Press.

Higgins, W. T. (1975). "A Comparison of Complementary and Kalman Filtering". *IEEE Tranactions On Aerospace and Electronic Systems* **3**, pp. 321–325.

Holden, C., R. Galeazzi, C. Rodriguez, T. Perez, T. I. Fossen, and M. Blanke (2007). "Nonlinear container ship model for the study of parametric roll resonance". *Modeling, Identification and Control* **28**, pp. 87–103.

IEEE (2008). *IEEE Standard Specification Format Guide and Test Procedure for Linear, Single-Axis, Nongyroscopic Accelerometers*. Institute of Electrical and Electronics Engineers.

IEEE (2014). *IEEE Standard for Sensor Performance Parameter Definitions*. Institute of Electrical and Electronics Engineers.

Jegaden, D. (2013a). *Mechanical vibration*. `http://textbook.ncmm.no/index.php/textbook-of-maritime-medicine/49-textbook-of-maritime-medicine/18-vibration/728-mechanical-vibration`. Accessed: 2015-11-22.

Jegaden, D. (2013b). *Vibrations on board ships*. `http://textbook.ncmm.no/index.php/textbook-of-maritime-medicine/49-textbook-of-maritime-medicine/18-vibration/729-vibration-on-board-ships`. Accessed: 2015-11-22.

Kuiper, J. B. (2002). *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace and Virtual Reality*. Princeton University Press.

Lauszus, K. S. *TKJ Electronics*. Accessed: 2015-11-23.

Lele, M. A. (2010). *Evaluation of Solid State Accelerometer Sensor for Effective Position Estimation*. MA thesis. Dalhousie University.

Liu, H. H. S. and G. K. H. Pang (2001). “Kinematic models for manoeuvring and seakeeping of marine vessels”. *IEEE Transactions on Industry Applications* **37**, pp. 812–819.

LPC1759/58/56/54/52/51 (2015). 8.6. Data sheet. NXP.

Madgwick, S. O. (2010). *An efficient orientation filter for inertial and inertial/magnetic sensor arrays*. University of Bristol.

MathWorks. *Mathworks a*. `http://se.mathworks.com/help/matlab/ref/atan.html?searchHighlight=atan`. Accessed: 2016-01-26.

MathWorks. *Mathworks b*. `http://se.mathworks.com/help/matlab/ref/atan2.html`. Accessed: 2016-01-26.

MathWorks. *Moving average filter*. Accessed: 2016-04-23. URL: `http://se.mathworks.com/help/econ/filtering.html`.

Mathworks. *Stepper motor model*. `http://se.mathworks.com/help/physmod/sps/powersys/ref/steppermotor.html?searchHighlight=steppermotor`. Accessed: 2016-01-23.

MPU-9150 (2012). 4th ed. Data sheet. InvenSense.

Nair, H. (2014). *The ultimate super duper altimeter/vario*. Accessed: 2016-01-04. URL: `http://pataga.net/imu_kalman_filter_notes.pdf`.

Ocean Wather Inc. `http://www.oceanweather.com/data/`. Current Marine Data. Accessed: 2016-04-04.

OpenCV. *Opencv*. `http://opencv.org/`. Accessed: 2015-11-23.

Park, M. (2004). *Error Analysis and Stochastic Modeling of MEMS based Inertial Sensors for Land Vehicle Navigation Applications*. MA thesis. Calgary University.

Passino, K. M. and N. Quij (2002). *Proportional-Integral-Derivative Control with Derivative Filtering and Integral Anti-Windup for a DC Servo*. Tech. rep. Department of Electronic Engineering, Ohio State University.

Pawlowski, A. (2010). *How safe is your cruise ship?* `http://edition.cnn.com/2010/TRAVEL/03/05/cruise.ship.safety/`. Accessed: 2015-09-30.

Pedley, M. (2013). *Tilt sensing using a three-axis accelerometer*. `https://www.nxp.com/files/sensors/doc/app_note/AN3461.pdf`. Accessed: 2015-10-14.

Perez, T. (2005). *Ship Motion Control*. Springer London Ltd.

Perez, T. and T. I. Fossen (2007). "Kinematic models for manoeuvring and seakeeping of marine vessels". *Modeling, Identification and Control* **28**, pp. 19–30.

Persson, A. and L.-C. Böiers (2010). *Analys i en variabel*. Ed. 3.1. Lund. Studentlitteratur.

PuTTY. *PuTTY FAQ*. `http://www.chiark.greenend.org.uk/~sgtatham/putty/faq.html`. Accessed: 2016-05-30.

Salamin, E. (1979). *Application of Quaternions to Computation with Rotations*. MA thesis. Stanford AI Lab.

Sauer, T. (2011). *Numerical Analysis*. 2nd ed. Pearson.

Searchlight system by Colorlight (2014). F1.2. Data sheet. Colorlight.

Seifert, K. and O. Camacho (2007). *Implementing Positioning Algorithms Using Accelerometers*. 0th ed. Freescale Semiconductor, AN3397. NXP.

Shah, S. (2014). *Real-time Image Processing on Low Cost Embedded Computers*. Tech. rep. Electrical Engineering and Computer Sciences, University of California.

Simon, D. (2001). "Kalman filtering". *Embedded Systems Programming* **14**. Cleveland State University, pp. 72–79.

SMHI (2015). *Lufttryck*. `http://www.smhi.se/kunskapsbanken/meteorologi/lufttryck-1.657`. Accessed: 2015-09-28.

Sparr, G. (1997). *Linjär Algebra*. Lund. Studentlitteratur AB.

Stockwell, W. *Bias stability measurement: allan variance*. `http://www.moog-crossbow.com/Literature/Application_Notes_Papers/Gyro_Bias_Stability_Measurement_using_Allan_Variance.pdf`. Accessed: 2015-12-02.

Technology, T. *Thermalcapture*. Accessed: 2016-01-05. URL: `http://www.teax-tec.de/index.php?id=23&L=1`.

Varesano, F. (2011). *Simple gravity compensation for 9 dom imus*. Accessed: 2016-01-09. URL: `http://www.varesano.net/blog/fabio/simple-gravity-compensation-9-dom-imus`.

Vector Nav. `http://www.vectornav.com/support/library/imu-and-ins`. Inertial measurement unit and inertial navigation. Accessed: 2015-11-09.

Vukmirica, V., I. Trajkovski, and N. Asanović (2010). *Two Methods for the Determination of Inertial Sensor Parameters*. Tech. rep. Belgrade. Military Technical Institute.

Wittenmark, B., K. J. Åström, and K.-E. Årzén (2012). *Computer Control: An Overview Educational Version 2012*. Available online: http://www.ifac-control.org/. Lund.

Woodman, O. J. (2007). *An introduction to inertial navigation*. Tech. rep. University of Cambridge. Computer Laboratory.

A

Pressure sensor conversion

When the pressure sensor is started, 11 calibration coefficients are collected. These as called AC1, AC2, AC3, AC4, AC5, AC6, B1, B2, MB, MC and MD. One last calibration coefficient is needed, which is computer by computing the true temperature. This is done as below.

$$\begin{aligned}
X1 &= \frac{(UT - AC6) \cdot AC5}{2^{15}} \\
X2 &= \frac{MC \cdot 2^{11}}{X1 + MD} \\
B5 &= X1 + X2 \\
T &= \frac{B5 + 8}{2^{4}}
\end{aligned}$$

The last calibration coefficient, B5, can then be used to calculate the true pressure.

$$
\begin{aligned}
B6 &= B5 - 4000 \\
X1 &= \frac{B2 \cdot \frac{B6 \cdot B6}{2^{12}}}{2^{11}} \\
X2 &= \frac{AC2 \cdot B6}{2^{11}} \\
X3 &= X1 + X2 \\
B3 &= \frac{AC1 \cdot 4 + X3 << oss + 2}{4} \\
X1 &= AC3 \cdot \frac{B6}{2^{13}} \\
X2 &= \frac{B1 \cdot \frac{B6 \cdot B6}{2^{16}}}{2^{16}} \\
X3 &= \frac{X1 + X2 + 2}{2^2} \\
B4 &= AC4 \cdot \frac{X3 + 32768}{2^{15}} \\
B7 &= (UP - B3) \cdot (50000 >> oss)
\end{aligned}
$$

where « is the logical right shift and *oss* is the oversampling setting which determine the mode of the pressure sensor. If B7 is less than 214748364, then the true pressure p is set to

$$p = \frac{(B7 \cdot 2)}{B4}$$

If B7 is equal or greater than 214748364, then the variable is set to

$$p = 2 \cdot \left(\frac{B7}{B4} \right)$$

The final steps for computing the true pressure is then defined as

$$
\begin{aligned}
X1 &= \frac{p}{2^8} \cdot \frac{p}{2^8} \\
X1 &= \frac{X1 \cdot 3038}{2^{16}} \\
X2 &= \frac{-7357 \cdot p}{s^{16}} \\
p &= p + \frac{X1 + X2 + 3791}{2^4}
\end{aligned}
$$

Lund University **Department of Automatic Control** **Box 118** **SE-221 00 Lund Sweden**	*Document name* MASTER´S THESIS *Date of issue* August 2016 *Document Number* ISRN LUTFD2/TFRT--6013--SE
Author(s) Martin Stanic	*Supervisor* Henrik Johansson, AES Nordic Anders Robertsson, Dept. of Automatic Control, Lund University, Sweden Karl-Erik Årzén, Dept. of Automatic Control, Lund University, Sweden (examiner) *Sponsoring organization*

Title and subtitle

Stabilization of a thermal camera at sea

Abstract

During, for example, search-and-rescue operations at sea, technical equipment like spotlights and thermal cameras are important aids. However, heave and sway affect the ship and make it harder for finding a person in distress.

This thesis presents a way of stabilizing a thermal camera by controlling a stepper motor connected to it. Moreover, the thermal camera can only be turned upwards and downwards.

The whole process was investigated to see what can and needs to be measured for stabilization at sea to work. With this knowledge, sensors had to be chosen to collect the necessary measurements.

The measurements from the different sensors were merged together using a Kalman filter to give an estimation for the tilt and elevation of a ship, which was used for controlling the stepper motor. Moreover, the control of the stepper motor was done using PID control.

The process was simulated in Simulink, making it possible to tune different parameters so that good performance was ensured based on assumed models. This was then implemented on the real system and tests were carried out to verify what was found during simulations. The result from this thesis is a stabilized thermal camera which helps an operator enormously in searching and finding objects at sea.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title 0280-5316		*ISBN*
Language English	*Number of pages* 1-99	*Recipient's notes*
Security classification		

http://www.control.lth.se/publications/