NTNU

Norwegian University of
Science and Technology

Developing an Autonomous Tracking System for the Atlantic Salmon

August Ekanger

Marine Technology
Submission date: June 2018
Supervisor: Martin Ludvigsen, IMT

Norwegian University of Science and Technology
Department of Marine Technology

Abstract

This thesis presents an autonomous fish tracking system for a tagged Atlantic Salmon using four unmanned surface vehicles equipped with acoustic receivers. The system can be used to monitor both hatchery farmed and wild Atlantic salmon in an open ocean environment, and scientists can use the data to develop a better understanding of the species and the way it interacts with its environment. Error sources associated with the purposed system are presented. A target localization method based on the principle of time difference of arrival positioning (multilateration), i.e an extended Kalman filter using acoustic data, is implemented and tested in a unified navigation environment (DUNE). The results reveal successful real-time localization of a target fish in a simulation located in the Trondheimsfjord. The hardware implementations presented in this thesis involves the integration of a GPS-receiver, CAN-controller, motor controller and Torqeedo thrusters with a single board computer. Thrust commands and thruster response is successfully tested at NTNU Gløshaugen, Trondheim, Norway.

This thesis examines how important control system components of a fish tracking system can be developed to optimize system performance. A conceptual control system with objectives of keeping receivers within transmission range while minimizing geometric dilution of precision (GDOP), travel distance and need for inter-vehicle communication, is presented. A conceptual formation controller unifies these system objectives in a single algorithm, and positions USVs in circular and circle-arc shaped formations depending on whether the target is located within a restricted area or in the open ocean. The need for inter-vehicle communication is significantly reduced by allowing vessels to operate with independent guidance systems. Positioning models for a cable-connected and fixed acoustic receiver configuration are developed with the purpose of accurately determining their position relative to a GPS aboard the vessel, while being subject to hydrodynamic forces.

Sammendrag

Denne oppgaven presenterer et autonomt fiskesporingssystem for Atlantisk laks bestående av fire ubemannede overflatefartøy utstyrt med akustiske mottakere. Systemet kan brukes til å overvåke både villaks og oppdrettslaks i et åpent havmiljø, og forskere kan bruke dataene til å danne en bedre forstelse av arten og måten den samhandler med omgivelsene på. De største feilkildene for systemet knyttet til lokalisering av målets posisjon er presentert. En målposisjoneringsalgoritme basert på prinsippet om multilaterasjon, mer presist et utvidet Kalman-filter for løsningsestimering av TDOA likninger, implementeres og testes i et enhetlig navigasjonsmiljø (DUNE). Resultatene indikerer vellykket sanntidslokalisering av en fisk i en simulering lagt til Trondheimsfjorden. Maskinvareimplementeringene som presenteres i denne oppgaven innebærer integrering av en GPS-mottaker, CAN-kontroller, motorkontroller og Torqeedo-thrustere med en datamaskin. Thrustkommandoer og thrusterrespons er p NTNU Gløshaugen, Trondheim, med tilfredsstillende resultater.

Denne oppgaven undersøker hvordan viktige systemkomponenter i et optimalt fiskesystem kan utvikles. Et konseptuelt kontrollsystem er presentert med målsetning om å holde mottakerne innenfor akustisk rekkevidde og minimere reiseavstand, systemkommunikasjon og geometriens innflytelse på posisjoneringsnøyaktigheten. En formasjonskontroller forener systemmålene i en felles algoritme og posisjonerer systemknutepunkter i sirkel- og sirkelbueformede formasjoner avhengig av om målet ligger innenfor et avgrenset område eller i det åpne hav. Modeller for kabelforbundne og faste konfigurasjoner presenteres med det formål å posisjonere akustiske mottakerne relativt til en GPS ombord i de ubemannede overflatefartyene, mens motakeren er utsatt for hydrodynamiske krefter.

Preface

This master's thesis is a part of the Master of Science (M.Sc) in Marine Technology at the Norwegian University of Science and Technology (NTNU) with specialization in marine cybernetics. The following is written in its entirety by August Ekanger during the spring of 2018.

My desire to understand more about the oceans, where so much is yet to be explored, served as the main motivation throughout this project. I believe the NTNU fish tracking project is important because it can help us understand how human actions impacts ocean environments and wildlife.

I want to thank professor Tor Arne Johansen and professor Martin Ludvigsen for supervising my project. I also want to thank students Artur Piotr Zolich and especially Stian Fredheim for their assistance in implementing hardware solutions and allowing me to learn from them. I want to thank all my friends and especially Axel Berggraf Egenæs whom designed the illustration on the front page of this thesis.

I want to thank my dear Maja for her love and support. Finally, I want to thank my loving family for our very special bond. My dear parents, Tina and Harald, for raising me with love and inspiring me to become the best version of myself. My brother Henrik, for being the funnies and kindest little brother anyone could ever wish for.

August Ekanger

Table of Contents

List of Figures

Abbreviations

TDOA	-	Time Difference of Arrival
TOA	-	Time Of Arrival
DPPM	-	Differential pulse-position modulation
GDOP	-	Geometric Dilution Of Precision
GNSS	-	Global Navigations Satellite System
GPS	-	Global positioning system (American system)
USV	-	Unmanned Surface Vehicle
LOS	-	Line-of-sight
FPR	-	Formation reference point
SNR	-	Signal to noise ratio
Timestamp	-	Reception time at single node associated with signal

Chapter 1

Introduction

1.1 Background

Lack of effective implementation of monitoring, control and surveillance in the fishery and aquaculture industry is a global problem. According to the food and agriculture organization of the united nations (FAO) the number of fish within biologically sustainable levels decreased from 90 to 68.6 percent in 2013, i.e the overfished share amounted to 31.4 percent. [1]

The Atlantic Salmon is a large pelagic ray-finned fish within the family of Salmonidae. Pelagic fish live in the Pelagic zone in the ocean, which is often referred to as the open ocean. Species within the Salmonidae family, including other species such as trout and freshwater whitefish, spawn in fresh water but spend the majority of their lives at sea. The sea wandering phase is very important for the Atlantic Salmon being the phase in which most of its individual growth occur, however, it is the phase scientists know the least about. Salmon are known to school when leaving the estuary and tend to live in the Pelagic zone. Fish in the Salmonidae family tend to return to their original habitat to reproduce migrating long distances up rivers. Atlantic salmon is therefore known as an anadromous species. The salmon related tourist industry began in 1830 in Norway when English tourist travelled to Norway to fish. Every year about 100 000 people participate in salmon fjord and river fishing.

Numbers from the Statistics Norway (Statistisk Sentralbyrå) indicate that approximately 1234 tonnes of farmed salmon was produced by aquaculture companies in Norway during 2016, with salmon amounting to 93 percent of the total export in terms of weight. [2] Also according to SSB Norwegian fish farms sold fish and shellfish for a total amount of approximately 43.6 billion Norwegian krones (NOK) in 2014. Aquaculture continues to be the fastest growing animal-food-producing sector in the world, and Norway is the seconds largest fish exporter, after China. [1] Norwegian authorities regulate the aquaculture industry through policies, regulations and required standards. To achieve sustainable growth in

the aquaculture industry new incentives to improve technology, development and research must be introduced.

In mainstream media is the relationship between the aquaculture industry and local authorities often portrayed as bad, and public discussions addressing the problems are often associated with polarizing parts from the aquaculture industry and environmental organizations. A report from 2012 published by the Norwegian aquaculture and fishery research firm Nofima, however, shows that the majority of municipalities in Norway are positive towards the industry and in favour of more aquaculture in their respective coastal zones. [3] Municipalities are mainly concerned with the lack of repercussions in the local economies and want compensation for the lack of value creation for local authorities.

Many fishermen and environmentalists are concerned about the environmental implications of the industry with genetical influence and escape, contamination and emissions, diseases and parasites, use of coastal areas and fish feed being major concerns. Many sports-fishers are afraid that the wild river stocks eventually will go extinct. A report from 2013 by the Norwegian research group SINTEF supports this claim. [4] This study presents a bio-economic model that describes the impacts of genetic interactions between wild and escaped farmed salmon. The results from the study indicate that the composition of the spawning population will change dramatically when escaped farmed salmon participate in the river spawning, with a complete replacement of the wild stock with farmed offspring being the worst case scenario. Others argue that the wild Arctic Salmon is not a threatened species despite loss of several wild stocks in some areas. Skeptics claim that factors such as electricity production, the parasite gyrodactylus salaris and acidification due to physical engagement in rivers serves as the main reasons to decline in fish stocks, and that the aquaculture industry is not to blame. Analyses from a study conducted in the Muchalat Inlet region of Canada over a period of 10 years, however, indicated a significant positive association between the sea lice abundance on Atlantic salmon farms and out-migrating infested wild Chum salmon. [5] Scientists within the field, such as R. Hillborn from School of Aquatic and Fishery Sciences at the University of Washington Seattle, stress the importance of analyzing the number of wild salmon migrating past fish farms. [6] Hillborn further wants estimate the total wild populations that are exposed using this data.

1.2 Aquatic telemetry

In a report from 1987 by NASA telemetry is defined to be the process of ”reliably and transparently convey measurement information from a remotely located data generating source to users located in space or on Earth.” [7] The word telemetry is derived from two words with Greek origin: tele and metron, which translate to remote and measure, respectively. Aquatic telemetry is the process of collecting measurements in a remote marine environment. Aquatic animals are defined to be the animals which are living solely or chiefly in water.

This section is mainly based upon the 2015 science magazine article *”Aquatic animal*

telemetry: A panoramic window into the underwater world" by N. E. Hussey et al. [8] As pointed out by this article aquatic telemetry has transformed our ability to observe and measure the behaviour of animals living in water environments. Recent technological advances in ultra-low powered electronic systems, digital signal processing and MEMS sensors offers new and exciting possibilities within the field of aquatic telemetry. [9] Telemetry data coupled with genetics, biochemical tracers and biologgers provides new and innovative insight into the behaviour of marine species. As radio waves lack the ability to penetrate water the field of aquatic telemetry is divided into two principle approaches; satellite and acoustic telemetry.

Satellite telemetry is suitable for tracking a variety of animals ranging from fish to marine mammals and reptiles. It is complementary to acoustic telemetry in the sense that a GNSS array of transmitters provide the data necessary to determine the animal's position and only a single receiver is needed. Satellite tags can transmit real-time position estimates and the data logged during dive each time the tagged animal surfaces. Satellite tags are, however, unable to determine the horizontal position of animals during their dives. The size of the tags often limit the possibility of tracking smaller fish and mammals.

Acoustic telemetry is based on the principle of using hydrophones to record the transmitted signal from a tagged animal. Acoustic transmitter tags (also known as fish tags) originating from the aquaculture industry are used to monitor the behavior of fish. These tags are able to measure a variety of variables such as temperature, depth, acceleration, dissolved oxygen, salinity and tilt/inclination. Some tags are able to both store and transmit the data. A tag only able to store data is often referred to as a biologger. Fish tags transmit data through an acoustic signal. One or several acoustic receivers (also known as hydrophones) can be used to gather the signal. It is possible to estimate the horizontal location of the fish tag using time of arrival measurements. Time of arrival is for the purpose of this thesis defined as the time an acoustic receiver obtains the signal transmitted by a fish tag. A large variety of position estimation techniques exist but due to the difficulties introduced by the harsh environment of underwater communication channels are TDOA localization methods recognized as the most rigorous. Acoustic telemetry is divided into two primary approaches: 1) receivers moored at fixed locations, and; 2) mobile receivers.

Moored receivers are applicable for monitoring fish in limited environments, e.g. farm cages in marine aquaculture or rivers. Fixed receiver networks have also proven useful in open environments to measure predictable behaviour of animals such as the Atlantic salmon *Salmo salar*. In an experiment performed in the river *Lærdalselva*, in Western Norway, the differences in behavioural patterns and migration timing between wild and hatchery-reared salmon smolts was examined using a receiver network of this kind. [10] A downside of moored receivers, however, is their inability to reconstruct detailed behaviour pattern for fish migrating over large ocean areas.

In recent years have acoustic receivers been attached to marine vessels forming a receiver network able to detect and track tagged animals. In august 2011 was a tagged leopard shark tracked by an AUV equipped with two acoustic receivers in the SeaPlane Lagoon in

Los Angeles, California. [11] A study conducted in southern Portugal involved satellite tags in addition to underwater and surface robotic vehicles to measure both the movements and the contextual environment of an Ocean Sun fish *Mola mola*. [12] The study reports "near-real time monitoring of finescale ($<$ 10 m) behaviour of sunfish". An AUV was used for video recording of the fish and a *WaveGlider* USV was used for measuring the contextual environment of the fish. The project is a perfect example of how several types of unmanned vehicles can collaborate in obtaining data in an open sea environment.

1.3 Context

This section aims to put this thesis in a contextual setting. Some of the results presented in a specialization project *Examining Use of Various Unmanned Vehicles in Single Fish Tracking*, written by this author in December 2017, are presented. [13] Related work by other peers at NTNU is presented.

Different unmanned vehicles can be used in an attempt of creating an optimal tracking system. Some of the advantages and disadvantages associated with the use of various unmanned vehicles in an autonomous fish tracking system are presented in *Examining Use of Various Unmanned Vehicles in Single Fish Tracking*. The paper considers use of unmanned surface vehicles (USVs), unmanned aerial vehicles (UAVs) and autonomous underwater vehicles (AUVs) as vehicle-hosts for acoustic receivers in a mobile tracking system. The following sections present the results which are relevant and to some extent form a basis for the control system presented in this master's thesis.

One of the main reasons why it is beneficial to use unmanned surfaces vessels (USVs) lay in their unique ability to communicate with both submerged, floating and flying nodes simultaneously. A surface vehicle connected to a submerged cable-connected hydrophone has the ability of sharing acoustic data through radio-communication, only limited by software and instrumental-induced delays and the speed of light. Near co-planar receiver-arrays are expected for a USV constellation, and associated with high geometric dilution of precision values in the orthogonal direction of the surface plane, i.e the depth direction.

UAVs and AUVs are possible mobile vehicle-hosts for acoustic receivers, with the main advantages being high speed and maneuverability for the former and possibility of obtaining receiver formations associated with lower degrees of geometric dilution of precision for the latter. Dynamic forces and noise induced by surface waves decay exponentially with increasing depth and submerged AUVs are therefore expected to have both a larger operational window and be subject to less noise. The main disadvantage of using UAVs is their inability to communicate acoustically while airborne. Communication delays and less accurate receiver clock synchronization are expected to be large sources of error using AUVs.

An unmanned fish tracking system should ideally be designed with respect to characteristics of the animal it is supposed to monitor and examine. A large variety of monitoring scenarios can be envisioned, e.g. tracking a pelagic fish along the shore or a coral

reef fish in a tropic environment. The behavior of a fish is dependent upon many variables such as species, age and surrounding environment. A variety of unforeseen events leading to demanding tracking conditions and potential target loss exist. Some examples are predator-prey interactions involving the target fish, sudden malfunction in one of the acoustic receivers or increased acoustic noise in the surrounding environment. If a target fish is lost in an open sea environment is it likely to never be detected again.

Research within the field of autonomous monitoring operations and acoustic telemetry have been the subject of several NTNU projects of recent. Two master's thesis' by Efteland [14] and Løvskar [15] from 2016 and 2017 examine use of TBR 700-RT acoustic receivers in underwater acoustic positioning systems. The former also examines time synchronization using GPS technology. A paper by Norgren and Ludvigsen from 2015 presents results from a tracking and remote monitoring experiment of an autonomous AUV, using an unmanned surface vehicle. [16] A paper from 2016 by Zolich, Johansen, Alfredsen and peers from KTH Royal Institute of Technology in Stockholm Sweden examines tracking of an acoustic fish-tag using unmanned vehicles, with a field validation giving proof-of-concept. A field experiment took place in Agdenes, Norway, during September 2016 monitoring a target AUV equipped with an acoustic transmitter. [9]

An extensive area of the Trondheimsfjord has been designated for testing of autonomous ships, with several projects planned and already conducted. Projects are often held in cooperation between commercial companies and academic research groups, such as NTNU's Centre for Autonomous Operations and Services (AMOS) and the Norwegian Marine Technology Reasearch Institute (MARINTEK). This test area is the first of its kind and it can be argued that projects held at this site are taking lead in transforming the maritime industry towards an autonomous future.

The Department of Engineering Cybernetics at the Norwegian University of Science and Technology (NTNU) has recently received funding for a fish tracking project involving the Atlantic salmon. The current project outline is to monitor a fish tagged with an acoustic transmitter utilizing a formation of four Otter USVs. Two Trondheim-based companies Thelma Biotel and Maritime Robotics deliver acoustic equipment and vessels to the project, respectively. The ultimate project goal is to conduct a tracking experiment with a wild salmon, likely located in the Trondheimsfjord, giving proof-of-concept. NTNU professor Tor Arne Johansen and PhD candidate Arthur Piotr Zolich are responsible for the project. The cost of the fish tracking system is significantly reduced by employing open-source software, low-cost acoustic systems, controllers and computers.

1.4 Motivation

The system presented in this thesis can be used as a surveillance tool for fish, and hence participate in developing a better understanding of interactions between species and environment. The fish tracking system could potentially contribute to global knowledge building by tracking a variety of species. A broader knowledge base and access to more data will ultimately result in better regulations for traditional fishery, locally and glob-

ally. Accurate geospatial data, coupled with other sources of data such as geologgers and weather data can be used by marine biologists to develop biological models for various species, e.g the Atlantic Salmon. Geospartial data is defined as geographic locations of a natural or constructed features above or below the earth's surface.

The Atlantic salmon is a natural target of the proposed tracking system because of the many stakeholders the species involves in Norway. Stakeholders involve sports fishermen, environmentalists, the aquaculture industry and its costumers. The presented system can gather data of wild salmon migrating close to fish farms. Scientist can further use this data to examine the exposure of the aquaculture industry and determine whether it should be held partially responsible for the declining number of wild Atlantic salmon.

A study from 2012 [17] discusses movements and dispersal of farmed Atlantic salmon following a simulated-escape event. Understanding the ecological impact and dispersal pattern of escaped farmed Atlantic salmon are stated as a major challenges, and the study suggests spreading recapture efforts over a relatively large area. The fish-tracking system presented in this thesis might serve as an important tool in examining the natural dispersal pattern of escaped Salmon from fish farms, and hence tactically narrow down the recapturing search area. The aquaculture industry is often obligated to pay fines in the event of fish farm escape events, in addition to costs associated with recapturing programs. In the future, stricter regulations regarding environmental impact of the aquaculture industry are expected. The industry itself therefore might have an economical incentive to invest in the development of, or data provided by, systems such as the one presented in this thesis.

1.5 Problem Definition

Aim
Contribute to developing an autonomous fish tracking system for Atlantic salmon.

Research Questions

1. How can important control system components of an optimal fish tracking system be developed?

2. What are the error sources associated with the proposed system?

This thesis presents a fish tracking system consisting of affordable vessels, acoustic system, software, computers and controllers. For the purpose of this thesis, an optimal fish tracking system is defined as a system delivering accurate and reliable geospatial data of its target during the entire operation, whilst being affordable and practically implementable. A proposed control system is optimized towards 5 objectives, and it is based on its performance in unifying these objectives that the quality of the system is assessed. Control system components, e.g a target estimator, formation controller, guidance system and receiver positioning models, are developed with the aim of optimizing the fish tracking system. Determining error sources of the proposed fish system is an important step towards understanding how the system can be further enhanced.

1.6 Method

Conclusions presented in this thesis are based upon results obtained by this author, other NTNU master's thesis' and credible literature within the fields of cybernetics and marine science. The university library of Norwegian University of Science and Technology has an online search engine, www.oria.no, where research articles and books are available in electronic versions. This search engine has been used to a large extent to acquire relevant and trustworthy information in the process of writing this thesis. Literature from the following scientific journals and conferences are used:

Conferences:

- IEEE OCEANS 2013, 2015 & 2017
- IEEE International Conference on Robotics and Automation

Journals:

- Proceedings of the National Academy of Sciences of the United States of America (PNAS)
- Environmental Biology of Fishes
- Science
- Fisheries Management and Ecology
- Fish and Fisheries
- ASME Journal of Basic Engineering
- Control Engineering Practice

The citations and bibliography presented in this thesis follow the Vancouver reference style. Pictures are cited with the associated URL addresses of where they are acquired and in general collected from the websites of the original manufacturer.

The developed extended Kalman filter is tested in a real-time simulation in DUNE: Unified Navigation Environment. Experiments and hardware implementations are conducted at *forsøkshallen* experiment lab, elektrobygg D at NTNU Glshaugen, Trondheim, Norway.

1.7 Thesis structure

The overall structure of this thesis is presented followingly.

- The second chapter elaborates on relevant theory: coordinates systems, localization and communication.

- A proposed fish tracking system is presented in the third chapter. An overview of the current system components and associated equipment precision is presented. Important control system components of an optimal fish tracking system are developed, i.e answering the first research question.
- The fourth chapter presents hardware implementations of a GPS receiver (Garmin 18x 5Hz), CAN controller (PiCAN2), motor controller (Torqeedo interface board) and Torqeedo thrusters with a Raspberry Pi 3 single-board computer. The developed target localization software in DUNE is presented.
- The fifth chapter presents a fish tracking simulation located in the Trondheimsfjord, with the aim of testing and verifying the developed software.
- Both research questions are further examined in the sixth chapter, i.e the discussion chapter. An evaluation of the validity of underlying assumptions leading to the developed receiver positioning models are presented.
- Conclusions are presented in the seventh chapter.
- The appendix contains documentation on developed software and hardware implementations presented in this thesis, including relevant information on various system components. The appendix aims to assist readers and future students in reusing and replicate solutions.

Chapter 2

Theory

2.1 Coordinate Systems & Transformations

The mathematics necessary for denoting a vector in various coordinate systems are presented in this section. This section is largely based upon theory presented in the book Handbook of Marine Craft Hydrodynamics and Motion Control by Fossen. [18]

ECEF Coordinate System

The Earth-centered Earth-fixed (ECEF) coordinate system has its origin in the center of the earth, i.e earth centered, with its axis fixed towards a reference pole, meridian and surface of the earth, i.e earth fixed. The coordinate system is often used by global guidance, navigation and control systems. The ECEF reference frame coincides with the Earth-centered inertial (ECI) frame once every 23 hours, 56 minutes, and 4 seconds.

NED Coordinate System

The North-East-Down (NED) coordinate system is defined with an origin CO relative to the World Geodetic System 1984 (WGS84), with axis' oriented towards North, East and towards the centre of the earth. The reference frame is frequently used to describe the position of objects on the surface of the earth.

Body Coordinate System

The body-fixed coordinate system has its origin in CO, a point which is usually set midships in the waterline. [18] The body coordinate system consist of three axis': x, y and z; the longitudinal, transversal(starboard), and normal, respectively. Motion of vessels in x, y and z direction are frequently referred to as surge, sway and heave, and rotation about the axis' as roll(ϕ), pitch(θ) and yaw(ψ), respectively.

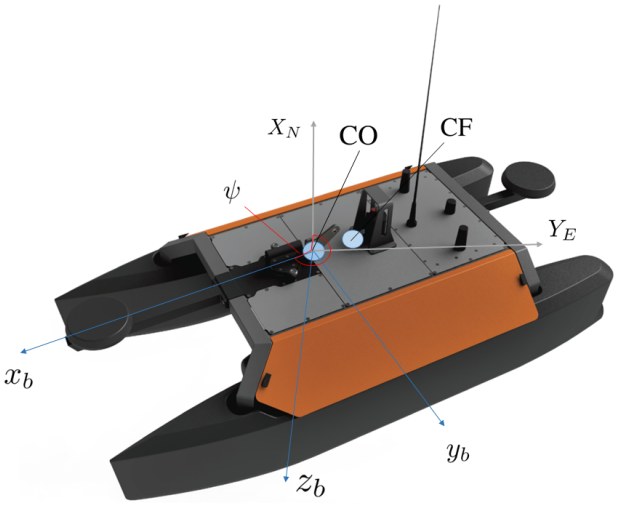

Figure 2.1: Otter USV: Body and NED coordinate frame [19]

By allowing the origins of the NED and body coordinate systems to coincide in the CO point can the yaw-angle ψ be defined as the necessary rotation for the X_N (North) axis to coincide with the x_b axis, i.e the axis pointing in surge direction. The center of flotation denoted CF is a point which coincides with the vessel centroid point in the waterline, and is the point a surface ship roll and pitch around. An Otter USV with a corresponding body and NED coordinate system is presented in figure 2.1. The yaw angle ψ is indicated by a red angle in CO.

LLH Coordinate System

In everyday life people normally navigate referring to their terrestrial position in ellipsoidal parameters longitude and latitude. Measurements from a Global Navigation Satellite System (GNSS) are usually presented to the user in ellipsoidal parameters. Geographical coordinates longitude, latitude and ellipsoidal height defines the position of an objective in the LLH coordinate system. A reference ellipsoid which is frequently used for navigation systems is the rotating ellipse WGS-84.

The WGS-84 is a global ellipsoid which can be used as an approximation of the mean sea level of the earth. The geoid is defined as the equipotential surface which coincides with the mean surface of the Earths oceans while not being subject to any forces, such as wind or tides, except the Earth's gravity and spin. The true topography of the earth does however rarely coincide with either of these models and the ellipsoidal height h is often approximated by the orthometric height, as given in equation 2.1,

$$h \approx H + G \tag{2.1}$$

where H is the orthometric height and G is the geoidal height. The deflection of the vertical is a measure of gravity field shift due to local anomalies such as mountains, and must

be taken into account in order to calculate the exact value of the ellipsoidal height. The deflection of the vertical is the angle between the true zenith and the line perpendicular to the surface of the reference ellipsoid. The associated errors from not considering the deflection of the vertical are small and can be neglected for practical purposes.

Notation

Let the position an object in the ECEF coordinate system be denoted by vector $\mathbf{p}^E$.

$$\mathbf{p}^E = \begin{bmatrix} x & y & z \end{bmatrix}^T \tag{2.2}$$

Let the position an object in the NED coordinate system be denoted by vector $\mathbf{p}^N$.

$$\mathbf{p}^N = \begin{bmatrix} N & E & D \end{bmatrix}^T \tag{2.3}$$

Let the position an object in the body coordinate system be denoted by $\mathbf{p}^b$.

$$\mathbf{p}^b = \begin{bmatrix} x^b & y^b & z^b \end{bmatrix}^T \tag{2.4}$$

Coordinate Transformations

From Body to NED coordinates

A position in the body frame can be transformed to the NED frame by three principal rotations about the x, y and z axis', as given in equation 2.5,

$$\mathbf{p}^N = \mathbf{R}_b^N \mathbf{p}^b \tag{2.5}$$

where the rotation matrix from body to NED coordinates denoted $\mathbf{R}_b^N$ is given by equation 2.6.

$$\mathbf{R}_b^N = \begin{bmatrix} \cos(\psi)\cos(\theta) & -\sin(\psi)\cos(\phi) + \cos(\psi)\sin(\theta)\sin(\phi) & \sin(\psi)\sin(\phi) + \cos(\psi)\cos(\phi)\sin(\theta) \\ \sin(\psi)\cos(\theta) & \cos(\psi)\cos(\phi) + \sin(\phi)\sin(\theta)\sin(\psi) & -\cos(\psi)\sin(\phi) + \sin(\theta)\sin(\psi)\cos(\phi) \\ -\sin(\theta) & \cos(\theta)\sin(\phi) & \cos(\theta)\cos(\phi) \end{bmatrix} \tag{2.6}$$

Flat Earth Navigation

For navigation in a local and limited area can the NED tangent plane be assumed to be fixed to the surface of the Earth. This is often referred to as flat earth navigation and gives a simple relationship between positions denoted in the NED and ECEF coordinate system.

$$\mathbf{p}^N = (\mathbf{R}_N^E)^T (\mathbf{p}^E - \mathbf{p}_{ref}) \tag{2.7}$$

where $\mathbf{R}_N^E$ is the rotation Matrix denoting the position of an object from NED to the ECEF coordinate system, given by equation 2.8,

$$\mathbf{R}_N^E = \begin{bmatrix} -\cos(l)\sin(\mu) & -\sin(l) & -\cos(l)\cos(\mu) \\ -\sin(l)\sin(\mu) & \cos(l) & -\sin(l)\cos(\mu) \\ \cos(\mu) & 0 & -\sin(\mu) \end{bmatrix} \tag{2.8}$$

where l is longitude, μ is latitude and $\mathbf{R}_N^E$ is part of a special orthogonal group of rotation matrices denoted **SO(3)** where the following mathematical conditions are valid: $\mathbf{R} \in \mathbb{R}^{3x3}$, det($\mathbf{R}$) = 1, and $\mathbf{R}^T\mathbf{R} = \mathbf{R}\mathbf{R}^T = 1$.

From NED to ECEF coordinates
Transformation from NED to ECEF reference frame is given by equation 2.9,

$$\mathbf{p}^E = \mathbf{R}_N^E(\mathbf{\Theta})\mathbf{p}^N \tag{2.9}$$

where $\Theta = \begin{bmatrix} l & \mu \end{bmatrix}^T$ and l and μ is longitude and latitude, respectively.

From ECEF to LLH coordinates
When transforming from ECEF to LLH coordinates are latitude and ellipsoidal height calculated implicitly whereas longitude can be found directly. The radius of curvature in the prime vertical is denoted N and is calculated using equation 2.10,

$$N = \frac{{r_e}^2}{{r_e}^2\cos^2(\mu) + {r_p}^2 sin^2(\mu)} \tag{2.10}$$

where r_p and r_e denotes the Polar axis and Equatorial radius of the WGS 84 ellipsoid, respectively.

Longitude l is found using equation 2.11,

$$l = \text{atan}(\frac{y}{x}) \tag{2.11}$$

and latitude and ellipsoidal height using implicit equations 2.12 and 2.13,

$$\tan(\mu) = \frac{z}{p}(1 - e^2\frac{N}{N+h})^{-1} \tag{2.12}$$

$$h = \frac{p}{\cos(\mu) - N} \tag{2.13}$$

where p is $\sqrt{x^2 + y^2}$ and e is the eccentricity of ellipsoid. The two implicit equations can be solved iteratively using a six-step algorithm presented in Fossen [18] given in equation 2.14,

$$\begin{aligned}
&\text{Step 1)} && p = \sqrt{x^2 + y^2} \\
&\text{Step 2)} && \tan(\mu_{(0)}) = \frac{z}{p}(1 - e^2)^{-1} \\
&\text{Step 3)} && N = \frac{{r_e}^2}{{r_e}^2\cos^2(\mu_{(0)}) + {r_p}^2\sin^2(\mu_{(0)})} \\
&\text{Step 4)} && h = \frac{p}{\cos(\mu_{(0)}) - N_{(0)}} \\
&\text{Step 5)} && \tan(\mu) = \frac{z}{p}(1 - e^2\frac{N_{(0)}}{N_{(0)}+h})^{-1} \\
&\text{Step 6)} && \text{if}(|\mu - \mu_{(0)}| < \text{tol}) \rightarrow \text{exit loop}
\end{aligned} \tag{2.14}$$

where tol is the error tolerance ending the iterative process.

From LLH to ECEF coordinates
Transforming positions in the ECEF reference system to LLH coordinates is done using the direct relationships presented in equation 2.15.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} (N+h)\cos(\mu)\cos(l) \\ (N+h)\cos(\mu)\sin(l) \\ \frac{{r_p}^2}{{r_e}^2}(N+h)\sin(\mu) \end{bmatrix} \tag{2.15}$$

2.2 Localization

For the purposes of this thesis let localization be defined as the process of positioning an object in four dimensions in time and space. Three important measures in localization are time of transmission (TOT), time of arrival (TOA) and pseudorange. TOT is the time it takes for a signal to propagate from the transmitter to the receiver. TOA is the time-stamp a receiver conducts a signal. A pseudorange is an estimate of the distance between a receiver and a transmitter.

TOA and TDOA localization
TOA and TDOA localization are two different positioning methods. The main difference is that TOA localization methods rely on the absolute time of arrival measurements whereas TDOA methods uses time difference of arrival measurements. The position of receivers obtaining measurements must be estimated with sufficient precision as they appear directly in positioning equations for both methods. The corresponding equations are quadratic of nature for both methods and an additional equation with regard to the number of dimensions is therefore needed to determine distinct solution.

A major difference between TOA and TDOA localization methods is the nature of the solutions. In TOA localization methods are solutions found as the intersections of circles and spheres in $\mathbb{R}^2$ and $\mathbb{R}^3$, respectively. The position of the signal source coincides with the intersection of these circles or spheres. The solutions of TDOA equations correspond to hyperbolic curves in $\mathbb{R}^2$ and hyperbolic surfaces in $\mathbb{R}^3$. The position of the signal source coincides with the intersection point of these curves or surfaces.

TOA localization methods estimate the pseudorange distance in between receivers and source of transmitted signal, using an estimate of the time in which the signal was sent. TDOA localization can be used in situations where the time of which a signals is sent is unknown. By subtracting two TOA measurements is this term eliminated. A TOA measurement obtained by one the receivers must used as a reference for comparison with other measurements, and TDOA methods therefore require an additional TOA measurement compared to TOA localization methods. A TDOA measurement corresponds to the Euclidean distance difference in between two observers and the source of the signal divided by the speed of the signal.

2.2.1 GNSS Positioning

Global Navigation Satellite Systems (GNSS) utilize atomic clocks aboard satellites to time-stamp signals, and satellite positioning is based upon the principle of TOA localization. This section is largely based upon theory from the book Integrated Satellite and Inertial Navigation Systems by Vik [20].

Nowadays, refers a Global Navitation Satellite System (GNSS) to any satellite-based system providing global real-time positioning and timing services to users. Most commercial GNSS receivers generate position, velocity and heading estimates. A GNSS consist of three main segments: the space, control and user segment. In the world today are the American and Russian systems Global Positioning System (GPS) and Globalnaja navigatsionnaja sputnikovaja sistema (GLONASS) fully operational, with the European and Chinese systems Galileo and Beidou currently under development. The foundation for the Global Positioning System was laid by the U.S army in the late sixties, and has developed from military purposes to include a large variety of civilian applications and hence turned into a big industry. A large variety of space (SBAS) - and ground (GBAS/LAAS) - based augmentation systems exist, and have the objective of enhancing GNSS accuracy and integrity. SBAS systems determine errors and transmit corrections and ionospheric maps to satellites, and integrity data for users. Ground stations main objective are to determine and broadcast local errors, mainly used in precision operations.

GNSS use two main sources of raw data in estimating the pseudorange distance in between the user and a satellite; pseudo-random sequences and carrier-phase measurements. A Doppler measurement and carrier to noise density is also available. In general is the carrier phase measurement more accurate but less reliable than the pseudorange measurement, because the carrier-phase measurement demands satellite lock and loss of lock occur quite frequently.

The main sources of error in GNSS are satellite based errors; clock and orbit error, propagation channel errors; multipath, interference, ionospheric and troposphere effects, receiver related errors; antenna effects, receiver clock error and bias, noise, and finally dilation of precision due to poor satellite geometry. From these many error sources are ionospheric delay and scintillation (due to free electrons in upper atmosphere) and geometric dilution of precision the main contributors. Many methods for enhanced accuracy exists, such as ionospheric compensation (frequency dispersive phenomena which allows modeling) and timing corrections provided by various systems.

The strength of GNSS signals are low due to heavy propagation loss traveling long distances from satellites to receivers. GNSS signals lack the ability of penetrating electrical conductive materials, such as water, which is the reason to why GNSS can not be used for underwater positioning. The signals are naturally able to penetrate non-conductive materials such as air, and to some extent also thin walls and glass (windows).

2.2.2 Acoustic Positioning

As GNSS signals are unable to penetrate water are other localization methods needed for monitoring of submerged fish. Methods utilizing acoustic signals are widely used in underwater operations. Fish tags are able to transmit acoustic signals containing messages of a large variety of data, such as ID and depth. These tags are, however, not capable of time-stamping their signals due to the fact that atomic clocks are too large and expensive this purpose. TDOA localization is the natural choice for fish tracking applications.

2.2.3 TDOA Localization

This chapter examines TDOA equations and their solutions in $\mathbb{R}_2$ and $\mathbb{R}_3$. Firstly, is the concept of noise amplification in TDOA localization examined. Secondly, are TDOA equations for three separate localization scenarios presented: i) four receivers in a horizontal plane, ii) four receivers in 3-space and iii) three receivers with a depth measurement in 3-space. Finally, are their solutions examined and graphically illustrated.

TDOA noise amplification
Noise is always present in acoustic signals. This section illustrates how noise is amplified in the process of subtracting TOA measurements, i.e creating a TDOA measurement. Noise amplification is one of the disadvantages of utilizing TDOA localization methods. Let y_1 and y_2 and be two TOA measurements given in equations 2.16 and 2.17,

$$y_1 = \frac{1}{\mathrm{c}}||\mathbf{x}_{R_1} - \mathbf{x}|| + w_1 \tag{2.16}$$

$$y_2 = \frac{1}{\mathrm{c}}||\mathbf{x}_{R_2} - \mathbf{x}|| + w_2 \tag{2.17}$$

where w_1 and w_2 are zero-mean Gaussian white noise, $\mathbf{x}_{R_1}$ and $\mathbf{x}_{R_2}$ are the positions of two acoustic receivers and $\mathbf{x}$ is the position of the target. The resulting TDOA measurement is given by equation 2.18,

$$\begin{aligned} \text{TDOA} &= (y_2 - y_1) \\ &= \frac{1}{\mathrm{c}}(||\mathbf{x}_{R_2} - \mathbf{x}|| - ||\mathbf{x}_{R_1} - \mathbf{x}||) + (w_2 - w_1) \end{aligned} \tag{2.18}$$

In the process of subtracting TOA measurements increases the mean value of the noise term $(w_2 - w_1)$ by a factor $\sqrt{2}$ and its variance by a factor of 2.

TDOA Equations

i) Four receivers in $\mathbb{R}^2$
Let R_1, R_2, R_3 and R_4 be four receivers in a two-dimensional plane. Let $\mathbf{x}_{R_j} = \begin{bmatrix} x_{R_j} & y_{R_j} \end{bmatrix}^T$

for j = 1,2,3,4, where $\mathbf{x}_{R_j} \in \mathbb{R}^2$ denotes the position of a single receiver, and let a target fish at an unknown target be: $\mathbf{x} = \begin{bmatrix} x & y \end{bmatrix}^T \in \mathbb{R}^2$. Let the target emit an acoustic signal with constant propagation speed c. By assigning the TOA measurement obtained by the receiver in position $\mathbf{x}_{R_1}$ to be the reference receiver three TDOA measurements can be obtained, as indicated in equation 2.19,

$$\text{TDOA}_{j-1} = \text{TOA}_j - \text{TOA}_1 \tag{2.19}$$

for j = 2,3,4. Utilizing the linear relationship between speed, time and position, the can equation 2.19 can be transformed into equation 2.20

$$\begin{aligned} \text{TDOA}_{j-1}\, \text{c} = &\sqrt{(x_{R_j} - x)^2 + (y_{R_j} - y)^2} \\ &- \sqrt{(x_{R_1} - x)^2 + (y_{R_1} - y)^2} \end{aligned} \tag{2.20}$$

for j = 2,3,4.

ii) Four receivers in $\mathbb{R}^3$
Let R_1, R_2, R_3 and R_4 be four receivers in a three-dimensional space, and let $\mathbf{x}_{R_j} = \begin{bmatrix} x_{R_j} & y_{R_j} & z_{R_j} \end{bmatrix}^T$ for j = 1,2,3,4, and $\mathbf{x}_{R_j} \in \mathbb{R}^3$ denote their positions. Let the position of target fish at an unknown location be denoted by $\mathbf{x} = \begin{bmatrix} x & y & z \end{bmatrix}^T, \mathbf{x} \in \mathbb{R}^3$. Given similar constraints and assumptions as presented in equation 2.19, three TDOA equations are obtained and presented in equation 2.21

$$\begin{aligned} \text{TDOA}_{j-1}\, \text{c} = &\sqrt{(x_{R_j} - x)^2 + (y_{R_j} - y)^2 + (z_{R_j} - z)^2} \\ &- \sqrt{(x_{R_1} - x)^2 + (y_{R_1} - y)^2 + (z_{R_1} - z)^2} \end{aligned} \tag{2.21}$$

for j = 2,3,4.

iii) Three receivers in $\mathbb{R}^3$ with a depth measurement
Let R_1, R_2 and R_3 be three receivers in a three-dimensional space such as described above. Two TDOA equations can be obtained and are presented in equation 2.22,

$$\begin{aligned} \text{TDOA}_{j-1}\, \text{c} = &\sqrt{(x_{R_j} - x)^2 + (y_{R_j} - y)^2 + (z_{R_j} - z)^2} \\ &- \sqrt{(x_{R_1} - x)^2 + (y_{R_1} - y)^2 + (z_{R_1} - z)^2} \end{aligned} \tag{2.22}$$

for j = 2,3.

Let z denote a horizontal plane in three-dimensions and D denote a depth measurement as presented in equation 2.23.

$$z = f(x, y) \tag{2.23}$$
$$= D \tag{2.24}$$

TDOA Solutions

i) TDOA Solutions: four receivers in $\mathbb{R}^2$
Four receivers are needed to determine the distinct solution of a transmitter in $\mathbb{R}^2$, i.e finding solutions to equation 2.19 for j = 2,3,4. Each TDOA equations have two solution functions y(x) as a result of its quadratic nature and 6 y(x) solutions in $\mathbb{R}^2$ can therefore be found. Let R_1, R_2, R_3 and R_4 be fixed in point locations (0,1), (7,1), (2,5) and (5,4), respectively. Let the position of the transmitter be (3,3). The distinct solution to the equations is given in the intersection of three hyperbolas in (3,3) as indicated in upper figure in figure 2.2. The corresponding MATLAB script creating the plot is included appendix section E.1.

ii) TDOA Solutions: four receivers in $\mathbb{R}^3$
In three dimensions are TDOA solutions hyperbolic surfaces in $\mathbb{R}^3$, hereby denoted f(x,y,z). Four receivers are normally sufficient to obtain a distinct solution. However, in some configurations can the distinct solution not be obtained. The plot in the middle of figure 2.2 illustrates four receivers and an unknown target indicated by blue and red colored dots, respectively. The solutions are indicated by 3 hyperbolic surfaces in yellow color. A sinusoidal surface plot with mean value 0 in yellow color was included to illustrate a fictive ocean surface. The target location of the fish is given as the intersection of the three TDOA surface solutions. The corresponding MATLAB script is provided in the appendix section E.2.

iii) TDOA Solutions: three receivers in $\mathbb{R}^3$ with a depth measurement
When a depth measurement is available is one less receiver needed to locate a target fish in three-dimensional space. A constellation of three receivers and a target fish with corresponding TDOA solution surfaces and depth measurement plane is illustrated in the bottom plot of figure 2.2. The solution and location of the target is found as the intersection between two TDOA hyperbolic surface solutions and the horizontal depth plane. The corresponding MATLAB script is included in appendix section E.3.

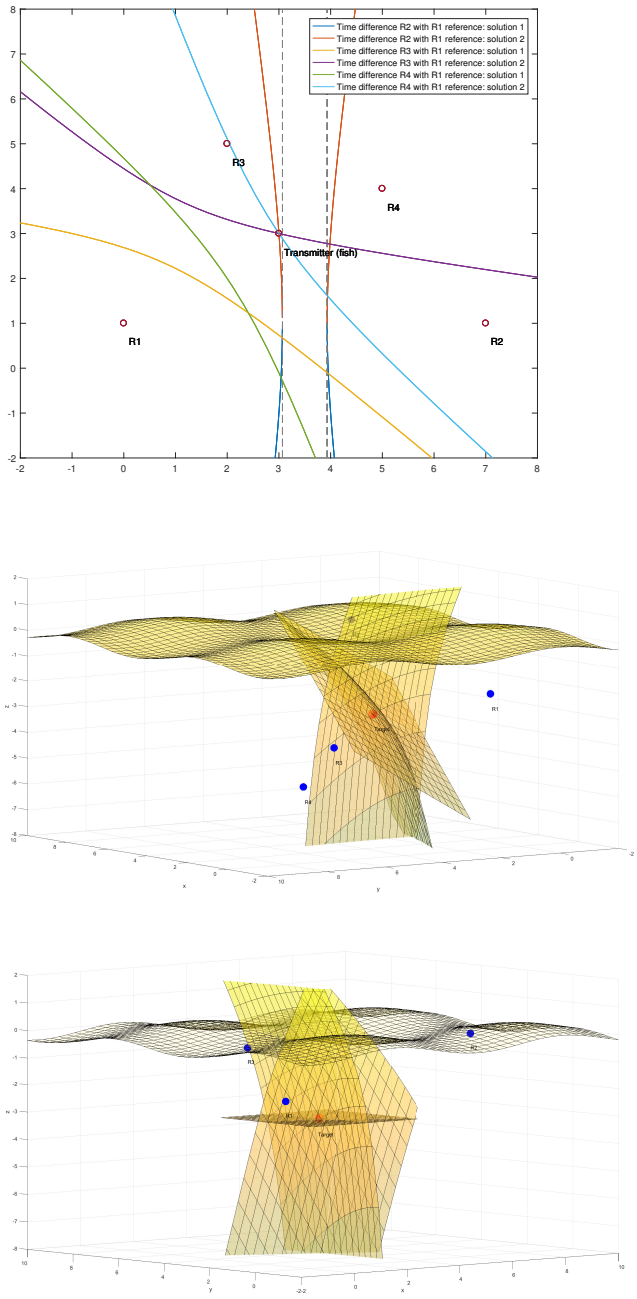

Figure 2.2: TDOA localization solutions: 2D with 4 receivers, and 3D with 4 receiver and 3 receivers & depth measurement

2.2.4 Single Measurement Localization

In some situations is potentially only one receiver able to acquire the acoustic signal transmitted by a fish tag. The principle of TDOA localization demands at least two TOA measurements and is therefore not applicable. A positioning technique using signal strength can be applied.

Let P_r and P_t denote the signal strength of the received and transmitted signal, respectively. Let r denote the Euclidean distance from transmitter to receiver and λ denote the wavelength of the signal. Path loss in free space of an electromagnetic signal is given as the dB-ratio of the relationship $\frac{P_r}{P_t}$, which further is proportional to the relationship presented in equation 2.25.

$$\begin{aligned} \frac{P_r}{P_t} &\propto \frac{\lambda^2}{16\pi^2 r^2} \\ \implies 10\log_{10}(\frac{P_r}{P_t}) &\propto 10\log_{10}(\frac{\lambda}{4\pi r})^2 \end{aligned} \tag{2.25}$$

Let the signal strength loss in decibels be donated by L as given in equation 2.26.

$$L \propto 20\log_{10}(\frac{4\pi r}{\lambda}) \tag{2.26}$$

By assuming that the acoustic signal transmitted by a fish tag decays as electromagnetic signals do in free space, with receiver & transmitter gains equal to one, can a pseudorange estimate $\hat{\rho}$ be found through the relationship presented in equation 2.27.

$$\hat{\rho} = \frac{\lambda}{4\pi} 10^{\frac{L}{20}} \tag{2.27}$$

Wavelength of the acoustic signal is given by the linear relationship $\lambda = \frac{c}{f}$, where c is the speed of sound in water and f is signal frequency ($\approx$ 69 kHz for most fish tags).

Imagine a situation in which a single acoustic receiver has acquired an acoustic signal and the associated depth measurement. In this situation can the location of the target can be determined to be within a circle set of solutions in $\mathbb{R}^3$ as the intersection of a horizontal plane $z = f(x, y) = C$ and a spherical solution set, presented in equation 2.28. Only locations on the bottom dome of the sphere are valid solution due to the fact that the fish is submerged.

$$\begin{aligned} \hat{\rho} &= ||\hat{\mathbf{x}}_{R_j} - \hat{\mathbf{x}}|| \\ &= \sqrt{(x_{R_j} - x)^2 + (y_{R_j} - y)^2 + (z_{R_j} - z)^2} \end{aligned} \tag{2.28}$$

In figure 2.3, a yellow dome represents a spherical solution set derived by a pseudorange. A horizontal plane determines the location of the target in depth direction, with an intersection solution circle indicated with red color.

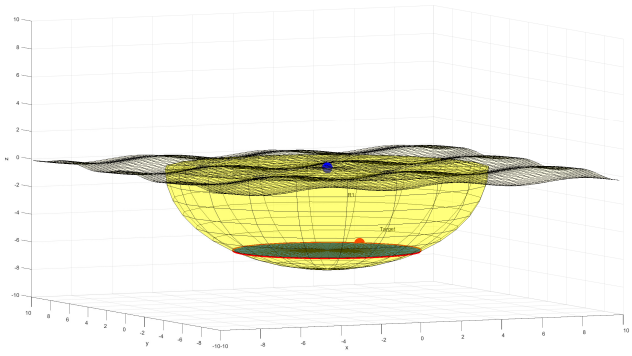

Figure 2.3: Signal strength localization

The true target and receiver position is indicated by a red and blue sphere, respectively.

2.2.5 Geometric Dilution of Precision

Geometric dilution of precision (GDOP) is a measure which represents the degree the relative geometry of the transmitter and receivers, i.e the receiver formation constellation, dilutes the solution accuracy of an estimator. [20] GDOP is defined as the sum of the diagonal elements in a co-factor matrix $\mathbf{G}$, i.e q_{xx} + q_{yy} + q_{zz}, as presented in equation 2.29,

$$\begin{aligned} \mathbf{G} &= (\mathbf{H}^T\mathbf{H})^{-1} \\ &= \begin{bmatrix} q_{xx} & q_{xy} & q_{xz} \\ q_{xy} & q_{yy} & q_{yz} \\ q_{xz} & q_{yz} & q_{zz} \end{bmatrix} \end{aligned} \tag{2.29}$$

where $\mathbf{H}^T\mathbf{H}$ is the Grammian matrix. $\mathbf{H}$ is the geometry matrix connecting TDOA measurements and states and is presented in an equation later in this thesis: equation 3.7.

The optimal receiver constellation is the one which minimizes the sum of diagonal elements in the dilution of precision matrix $\mathbf{G}$:

$$\min_{\mathbf{x_R}\in\mathbb{R}^3} (\text{Tr}(\mathbf{H}^T\mathbf{H})^{-1}) \tag{2.30}$$

where Tr is the trace operator of the matrix (sum of its diagonal elements), and $\mathbf{x_R} \in \mathbb{R}^3$ is the positions of receivers in 3-space, e.g. a USV constellation.

The constellation which minimizes dilution of precision is a receiver array evenly distributed on the surface of a sphere, with target in the sphere origin. Constellations with aligned or co-planar receivers should ideally be avoided, as singularities in the H matrix give large dilutions of precision values. In the case of using USVs for fish tracking are receivers more or less bound to the surface plane, i.e co-planar, and the ideal constellation is therefore not realizable. A depth measurement is needed to determine target position in

depth direction.

2.3 Communication

2.3.1 Serial Communication

Serial communication can be defined as the process of sending data bit-wise through an electric circuit. A serial bus is defined as a shared channel using serial communication. Serial communication is normally divided into two principle groups: synchronous or asynchronous.

Synchronous serial communication pairs the data with a clock signal and all devices connected to the bus share a common clock. In asynchronous serial communication is data transferred without clock synchronization data which minimizes the required wires and I/O (input/output) pins.

Serial communication protocols
Both bit and byte oriented protocols exist. A byte is a 8 bit data message, allowing $2^8 = 256$ unique variables or symbols. The hexadecimal numeral system is widely used in computer science to represent binary coded values using 16 distinct symbols (0-9 and A-F) to represent 4 binary values. Two hexadecimal digits are therefore able to represent 8 bits, i.e 256 various symbols.

Baud rate is often confused with bit-rate (or bps - bits per second), despite the fact that they are only equivalent when there is two symbols in the Baud alphabet (often 1 or 0), which often is not the case.

Serial communication protocols and standards used by the fish tracking system presented in this thesis, such as SPI, ASCII, RS-232, RS-485, USB and UART, are briefly described in the appendix section A. The Controller Area Network protocol is presented in the following section due to its distinctive relevance for the hardware implementations presented in chapter 4.1.

Controller Area Network
A Controller Area Network (CAN bus) is a multi master serial bus standard which enables micro controllers and other devices to communicate with each other without a host computer. CAN is a broadcast type of bus, in which all nodes invariably pick up messages, and local hardware filtering is therefore needed to ensure that components only react to interesting messages. CAN bus uses two wires, one for CAN high and another for CAN low signals. In idle mode are both lines carrying 2.5 volts (recessive voltage). While transmitting data are the high and low line carrying 3.75 and 1.25 volts (dominant voltages), respectively, and hence generating a 2.5 voltage differential.

The CAN bus utilizes a lossless bitwise arbitration method for transmitting data. The method requires nodes to be synchronized for every bit in the network. This is the reason to why CAN networks are sometimes referred to as synchronous despite not transmitting a clock signal with the data. CAN frequently used in the automotive industry and automation environments in general due to the low cost of CAN controllers and processors.

2.3.2 4G Communication

4G is a broadband cellular network, more specifically a distributed network over land areas (cells) where the last link is wireless (antenna). 4G referrers to the fourth generation of mobile communications. The first release Long Term Evolution (LTE) is a standard for wireless communication and has existed commercially in Norway since 2009. 4G LTE is based upon the principle of orthogonal frequency-division multiple access (OFDMA), grating multiple access through assigning users subsets of the radio wave carrier.

2.3.3 Satellite Communication

The GPS signal structure as presented in [20] is given in equation 2.31,

$$\begin{aligned} L_1 &= A_1 p(t) d(t) cos(f_1 t) + A_1 c(t) d(t) sin(f_1 t) \\ L_2 &= A_2 p(t) d(t) cos(f_2 t) \end{aligned} \tag{2.31}$$

where f_1 = 1575.42 MHz and f_2 = 1227.6 MHz which corresponds to the K_u band (band within microwaves spectrum) in the electromagnetic spectrum. The K_u band involve signals with frequencies ranging from 12 to 18 gigahertz (GHz).

GPS signals transmit navigation messages with ephemeris and almanac data which includes satellite orbit parameters and satellite constellation parameters, respectively. When a GNSS signal reaches a receiver various signal processing and decoding stages take place inside the receiver hardware to make use of the data. This involves a two-dimensional acquisition process, signal tracking, signal monitoring, signal observation (pseudorange determination), navigational message decoding, and finally a navigation solution solver.

2.3.4 Underwater Acoustic Communication

DPPM is a form of signal modulation where message bits are encoded by transmitting acoustic waves where each pulse is positioned and encoded relative to the previous pulse. The acoustic receiver measures difference in pulse arrival time to decode the data message. DPPM is inherently less sensitive to multipath interference than other modulation techniques, e.g the pulse-position modulation method, which is a big problem in acoustic communication.

Chapter 3

Fish Tracking System

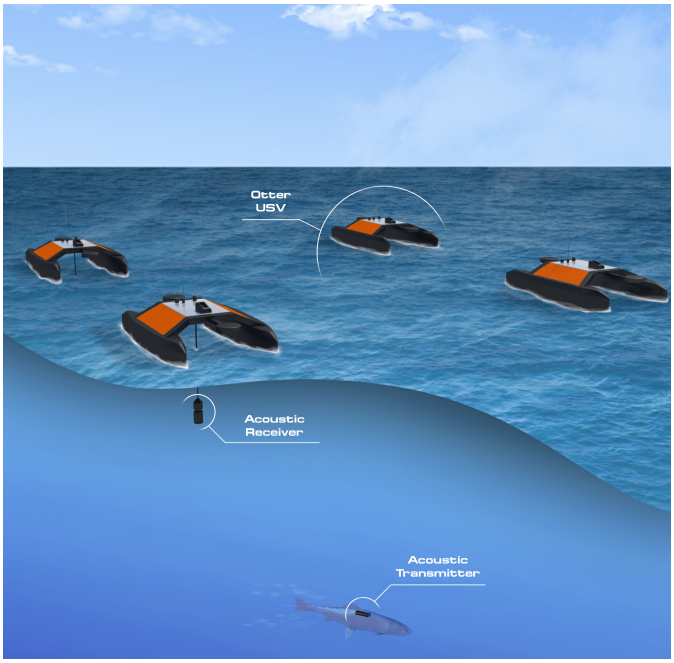

Figure 3.1: Proposed Fish Tracking System

This chapter presents a tracking system which can be used to monitor and track an Atlantic Salmon, utilizing four Otter USVs and acoustic receivers. Its objective is to create accurate geospatial data of the target fish by timestamping and decoding acoustic DPPM signals transmitted by a fish tag.

In theory, the proposed fish tracking system has guaranteed ability of calculating distinct fish position in $\mathbb{R}^3$ as long as a depth and four TOA measurements, i.e three TDOA measurements, are successfully received whilst obtaining formations in which receivers are not aligned and measurements are perfect. In many configurations three TOA measurements, i.e two TDOA measurements, are also sufficient to calculate distinct target position. However, it is greater robustness and less geometric dilution of precision expected using four Otters USVs.

This chapter introduce the various system components currently intended for the NTNU fish tracking system, and describe their purpose in the proposed control system. First, schematics demonstrating the current hardware setup, information flow, communication protocols and connections are presented. Secondly, five system objectives for a proposed control system to fulfill are presented. Finally, are the different layers of control elaborated upon in separate sections.

3.1 Schematics

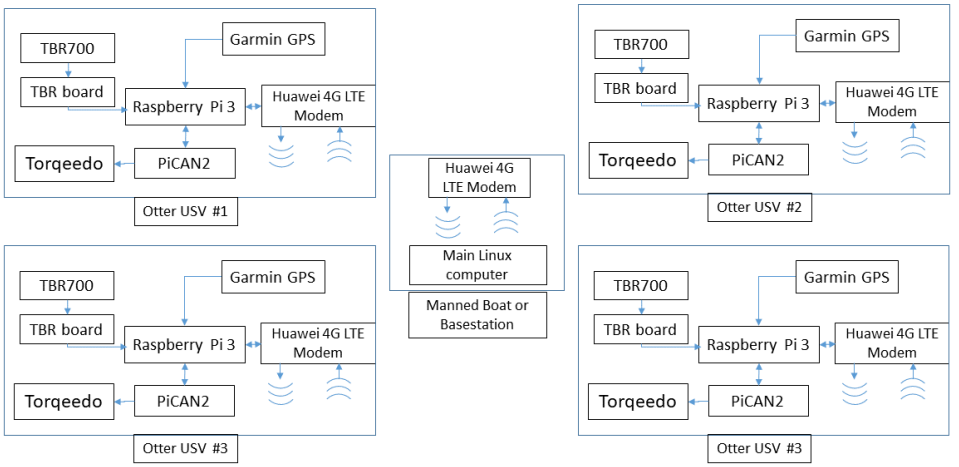

Figure 3.2: Hardware Schematics

The current hardware configurations of the NTNU fishtracking system is presented in figure 3.2. Sensor data obtained by four client nodes (Otter USVs) is transmitted to the centralized system node using 4G LTE communication. The centralized node is a computer running operation software and explained in further detail in the following sections. This computer can be placed in a mission center on land or aboard a vessel. In development stages is it natural to obtain this computer aboard a manned vessel in which operators can interfere through a user interface depending on what they observe during operation. Ideally, as the degree of autonomy in the system increases, it can be possible to position the centralized node aboard one of the four Otter USVs.

A schematics indicating the physical connections in the proposed system is presented in figure 3.3. Note that the PiCAN2 CAN controller can be connected to the Raspberry Pi 3 through two different connections, a 9-way sub-D connector or a 4-way screw terminal.

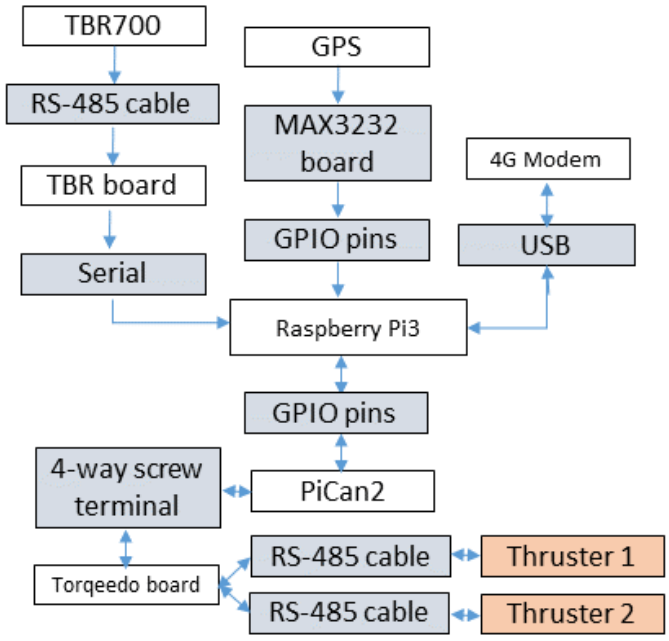

Figure 3.3: Otter USV: Connections schematics

Figure 3.4 indicates information flow from an acoustic transmitter (fish tag) to an acoustic receiver aboard a single Otter USV. Serial communication is indicated by arrows while analog communication methods (FM & acoustic) are indicated by a wave symbol. Color-coded boxes indicate serial, acoustic and radio communication for blue, green and yellow color, respectively. Additional boxes indicating communication protocols used in each communication link are placed next to their associated links. Further details regarding communication protocols used by the system can be found in the appendix section A.

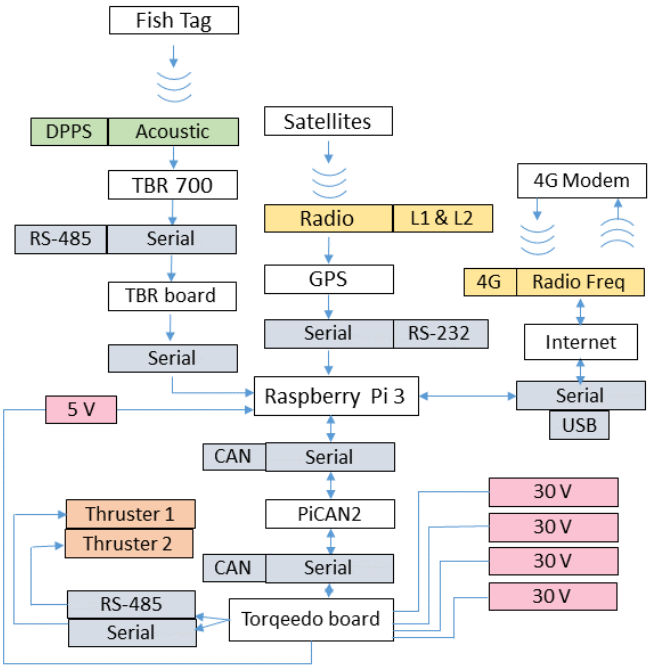

Figure 3.4: Otter USV: communication flow

3.2 Otter USVs

Purpose:
The main purpose of the Otter USVs is to enable positioning of acoustic receivers in time and space.

Connection:
The TBR 700 real-time acoustic receiver must be connected to the Otter USV through a cable-connected or fixed configuration, including a RS-485 cable for serial communication.

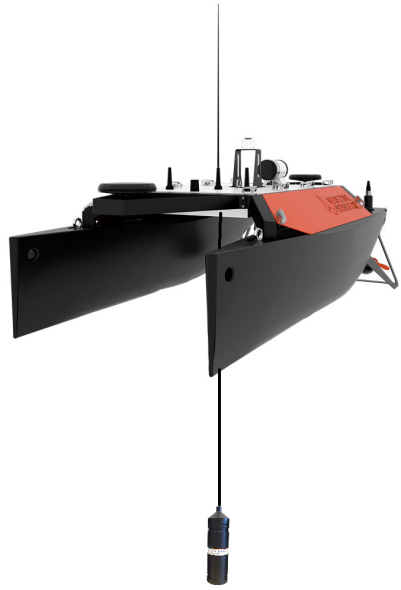

Figure 3.5: Otter USV: cable-connected acoustic receiver [21] [19]

The acoustic sensor must be submerged throughout the entire operation to enable detec-

tion of acoustic signals transmitted by a fish tag. The final receiver configuration, i.e the physical mechanism connecting the receiver to the vessel, is yet to be decided upon.

Specifications:
The Otter USV is an autonomous catamaran-shaped vessel equipped with two fixed electric thrusters weighing about 95 kilo with a max speed of 6 knots. The dimensions of the vessel are 105x200x85 cm for width, overall length and height, respectively. The vessel is conveniently transported through being equipped with detachable pontoons, and is indented for operations in sheltered waters such as smaller lakes, canals, rivers, harbour areas and sheltered fjords in calm weather conditions. The long endurance vessel is powered by up to four interchangeable battery packs with a capability of travelling 2 knots for up to 20 hours. According to the manufacturer is it theoretically possible to get up to 100 hours of endurance in 1,5 knots, with no payload and zero energy reserves, using four 30 Volt (915 Watt hour) batteries. The catamaran-shape of the Otter USV allows it to carry a large variety of sensors for environmental mapping between its pontoons, e.g an acoustic receiver.

Torqeedo thrusters
The Otter USV is delivered with two detachable fixed-position Torqeedo thrusters with a maximum propulsive power of 180 Watts. The thruster is powered by electricity of approximately 28.8 Volts with maximum propeller speed and power supply of 1200 rounds per minute (rpm) and 1200 Watts, respectively.

3.3 Hardware

3.3.1 Sensors

Fish Tag

A Norwegian manufacturer Thelma Biotel delivers fish tags which can be used in fish-tracking experiments, with weights ranging from 17.5 to 2.1 grams. The ADT-16 fish tag is a depth transmitter developed for migrating Atlantic salmon smolt, and a potential candidate for the final system.

Figure 3.6: Acoustic receiver & transmitter: TBR 700 RT & ADT-16 fish tag [22] [23]

Purpose
The purpose of the fish tag is to transmit acoustic signals omnidirectionally through the water column using a piezoelectric transducter. Depth data is transmitted by the principle of acoustic Differential Pulse Position Modulating (DPPM) and carried by acoustic signals with a frequency of 69 kHz.

Connection
The fish tag is either connected to the skin of the fish in a place in which it does not prohibit its movement, or injected into the belly of the animal.

Specifications
According to the manufacturer do their fish tags have a size dependent transmission range up to 290 meters. The depth measurements of the ADT-16 tags have a maximum offset error of +/- 50 cm. The tag has a cylindrical shape with a length and diameter of 55 mm and 16 mm, respectively, and weighs approximately 17.5 grams. Measured depth is continuously encoded as the time delay between two acoustic pulses (DPPS). A delay of 1000 ms corresponds to 0 m and increases by 100 ms per meter. The resolution of the transmitted signal is 10 cm which corresponds to 10 ms.

Acoustic Receiver - TBR 700 RT

Purpose
The purpose of the acoustic receiver is to timestamp, i.e create TOA measurements, and decode DPPM acoustic signals transmitted by a fish tag. Variables such as ID, timestamp in Linux time seconds and milliseconds, code type, pressure data (depth) and signal to noise ratio (SNR) are logged and transmitted in real-time.

Connection
The receiver is communicating serially through a standard multipoint RS-485 cable enabling real-time data acquisition. A transceiver chip which converts UART from the Raspberry Pi 3 to RS-485 is necessary to enable communication with the receiver. Data can also be gathered from the equipment through USB or Bluetooth if used as a standalone logger unit.

Specifications
TBR 700 has full multi-frequency reception in the 63-77 kHz band with advanced digital signal processing for noise reduction, enabling maximum reception. The hydrophone dynamically adjusts threshold levels in order to optimize reception in harsh sound conditions. Acoustic signals are timestamped with an associated uncertainty of approximately $\pm$ 1 ms. The receiver is a long endurance equipment with battery life capability of as high as 16-18 months, weighs about 1.14 kg and is 75 and 230 mm in diameter and length, respectively.

Data message

A data message from the TBR 700 has the following form,
$ ⟨ TBR ID⟩, ⟨ timestamp[s]⟩,⟨ timestamp[ms]⟩,⟨ code type⟩,⟨ tagID⟩,⟨ Pressure data⟩,⟨ SNR ⟩
and an example message can be,
$ TBR05,1446716612,123,S256,2,233,50
where receiver number is 05, 1446716612 is seconds, 123 is milliseconds, code type is S256, tag ID is 2, pressure data is 233 and SNR ratio is 50. The timestamp is a 32 bit signed variable in Unix time, i.e number of seconds since 1 January 1970. The pressure data is a data value ranging from 0 to 255.

Garmin 18x 5Hz Outdoor GPS

Figure 3.7: Garmin 18x 5Hz outdoor GPS & MAX3232 line driver [24] [25]

Purpose
A Garmin GPS 18x 5Hz provides position and velocity estimates of its position aboard an Otter USV.

Connection
The GPS comes with a 5 meter connection cable with 6 wires, Measurement Pulse Output (#1, yellow), Power Vin (#2, red), Ground (#3, black), Transmit Data (#4, white), Ground (#5, black) and Receive Data (#6, green). CMOS serial output level from 0 Vdc to Win, with 4-5.5 Vdc (Asynchrounous serial). The input voltage and current of the Garmin GPS is 4.4-5.5 Vdc (voltage direct current) and 100 mA, provided by the general-purpose input/output (GPIO) pins of a Raspberry Pi 3 in this project. The GPS cannot be connected directly to the GPIO pins of a Raspberry Pi 3. A RS-232 conversion board is needed between the devices to make use of serial GPS data. A Mikroelektronika MAX3232 is an electrical circuit board which enables RS-232 performance from a 3.0 to 5.5 volt operating range which is used for this purpose.

Specifications
The update rate is 5 Hz, i.e 5 measurements per second, with a user equivalent error (UERE) less than 15 meters 95% of the time. The GPS is designed to withstand rugged operating conditions and should be waterproof to about 1 meter for 30 minutes, which allows use in prevailing weather conditions. A clear view to GPS satellites is preferable as GNSS signals in general lack the ability to penetrate mediums between satellite and

receiver. The minimum GPS receiver sensitivity is -185 dBW.

Data
The data format is NMEA0183 electrical and data specification for serial communication, which is often used in marine sensor systems. The default setting is 19200 baud.Data messages from the GPS are transmitted on the following form,
$ ID, timestamp, latitude, longitude,fix quality, No. of satellites, HDOP
where latitude and longitude are given on the form 6435.99677,N and 1051.23965,E, corresponding to 64^o and 35.99677 minutes latitude and 10^o and 51.239625 minutes longitude. Fix quality is a number between 0 and 8, where 0 = invalid, 1 = GPS fix (SPS), 2 = DGPS fix, 3 = PPS fix, 4 = Real Time Kinematic, 5 = Float RTK, 6 = estimated (dead reckoning), 7 = Manual input mode, 8 = Simulation mode. No. of satellites is the number of visable satellites and horizontal dilution of precision (HDOP) is a measure of the horizontal satellite geometry accuracy.

3.3.2 Computers & Controllers

Raspberry Pi 3

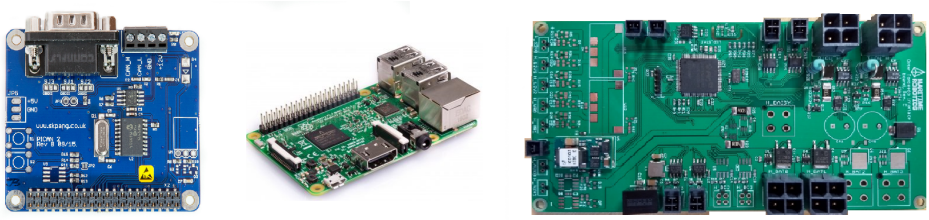

Figure 3.8: Computers: PiCAN2, Raspberry Pi 3 Model B, & Torqeedo interface board [26] [27] [28]

Purpose
The purpose of the Raspberry Pi 3 board is to run Unified Navigation Environment (DUNE) software and connect and communicate with various hardware devices of the system; GPS, PiCAN2, Huawei E3372, and TBR 700 board.

Connections
During fish tracking operation is the Raspberry Pi 3 Model B (RP3) powered by the Torqeedo interface board 5V outlet ”H_5V” through a Molex Nano-Fit 2-ways receptacle connector. In test phases can it be powered by a regular micro USB (phone-charger). A number of GPIO pins provide lower-level output and can be used for serial communication, as indicated in GPIO header schematics in appendix figure 5.

Specifications
The Raspberry Pi 3 Model B is a small single-board computer. The board has a 64 bit quad core processor with a speed of 1.4 GHz and 1 GB RAM. Raspbian is a Debian-based

operating system for Raspberry Pi computers.

Can controller - PiCAN 2

Purpose
The PiCAN 2 controller provides CAN-bus capability to Raspberry Pi 3. The objective of the CAN controller is to store received serial bits from the CAN bus and in that way fetch data messages which can be utilized by the host micro processor (RP3). The host micro processor transmits messages to the CAN controller which is serially transmitted onto the CAN bus.

Connection
PiCAN2 communicates with RP3 through its GPIO pins and with the Torqeedo interface board through a 2-wire CAN-cable connected to a 4-way screw terminal.

Specifications
PiCAN 2 has high speed SPI interface (10 MHz) with CAN v2.0B at 1 MB/s.

Torqeedo Interface Board

Purpose
The main purpose of the Torqeedo interface board is to provide power and control to the thrusters by communicating with the motors and batteries through implementing a gateway between six Torqeedo RS-485 buses and a CAN interface. The motor controller board acts as an intermediary board between the micro-controller boards, batteries and thruster. The Torqeedo board provides a variable d.c (direct current) to the thruster through a series of pulses, i.e pulse width modulation (PWM).

Specifications
The Torqeedo interface board has a total of 4 power rails and 10 controllable outputs, e.g current, voltage, power, temperature, RPM, status and error flags. The board controls power individually to two Torqeedo Ultralight motors and protects the motors with ultra-fast fuses.

3.3.3 Communication

Huawei E3372 LTE 4G Modem

Purpose
Enable communication between the Raspberry Pi 3 computers aboard Otter USVs and a centralized unit, sharing IMC messages containing sensor data and allow system nodes to receive formation control messages.

Connection

Figure 3.9: Huawei E3372 modem ([29])

RP3 USB port

Specifications
The Huawei E3372 is a 4G USB Stick which supports LTE download speed to 150 Mbps and upload speed to 50 Mbps.

3.4 Software

The vessels are delivered without the original navigation system intended and created by the manufacturer. The current plan is therefore to develop a navigation and control system using the LSTS toolchain. LSTS toolchain is open-source for non-commercial users and has been used in several projects at the University of Porto and NTNU. Implementation and testing of a target estimator in DUNE is presented in this thesis.

3.4.1 LSTS Toolchain

LSTS is a software toolchain developed for supporting networked heterogeneous air and ocean vehicles. [30] The toolchain supports the deployment of air and ocean vehicles interacting over limited acoustic and wireless networks combined with disruption tolerant networking protocols. The LSTS Toolchain for Networked Vehicle Systems originated in underwater robotics. The toolchain consists of three main parts: DUNE, Neptus (command and control software) and IMC (inter-module communication protocol). The different components are briefly presented in figure 3.10.

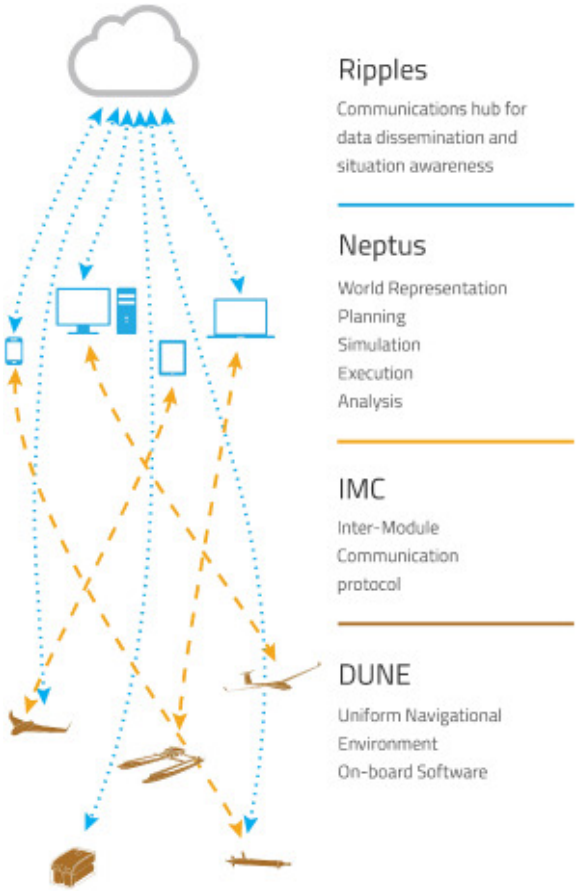

Figure 3.10: LSTS toolchain [31]

3.4.2 Neptus

Purpose
Neptus is used to monitor the tracking system and create a user interface which enables operators to interfere with DUNE.

Communications
Neptus communicates with DUNE through IMC messages.

Specifications
Neptus is a flexible system built to encompass diverse vehicles and scenarios. The software can be installed on a Linux or Microsoft computer. An operator is needed in order to visually plan and review the mission while being executed.

3.4.3 Inter-Module Communication

An IMC specification (message) includes two Extensible Markup Language (XML) documents which are consumed by DUNE at compile time. XML is a software- and hardware-independent tool which is used for data transmission in both a human- and machine-readable format. DUNE tasks communicate with one another, and other instances such as Neptus using the IMC protocol, which acts like a local bus. A figure which indicates how DUNE tasks communicate is presented in figure 3.11.

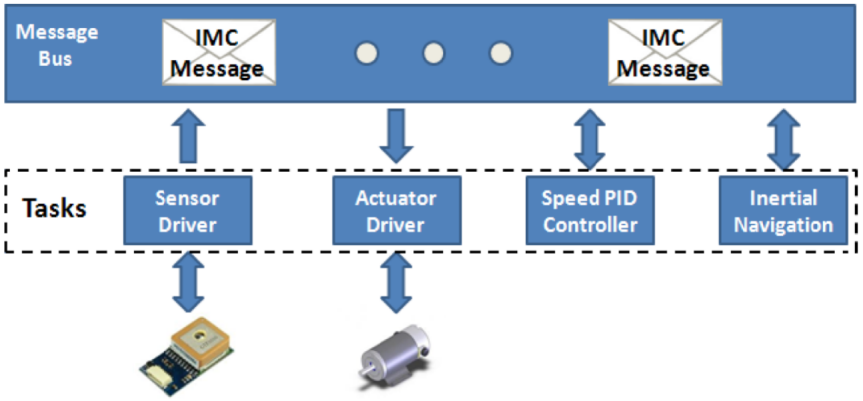

Figure 3.11: IMC bus & DUNE tasks [32]

3.4.4 DUNE: Unified Navigation Environment

Purpose
DUNE is used to write embedded software code for control, navigation, actuators and sensors.

Communications
The DUNE functions consume, bind and dispatch are used to gather and dispatch data to the IMC BUS. The dispatch function allows a task to dispatch any type of IMC message to the bus, whereas the consume function allows a task to subscribe to specific IMC message relevant to the task. The bind and consume function instantaneously load IMC message from the bus, and the developer can adjust how the task should make use of the data.

DUNE tasks
DUNE is built upon a large number of predefined tasks within main folder groups such as Actuators, Autonomy, Control, DUNE, Main, Maneuver, Navigation, Plan, Power, Simulators, Supervisors, Transports, User Interfaces. DUNE tasks are created in the object-oriented programming language C++. The level of software portability in C++ is high and

the language allows developers to manipulate individual bits at specific memory locations, which makes it suitable for writing to hardware device drivers. DUNE runs on a very low footprint and on a variety of operating systems. DUNE runs on a Raspbian operating system in this thesis project.

DUNE task life cycle
All tasks in DUNE executes accordingly to a predefined cycle, involving 8 cycle steps:

Step 1) *Task(const std:string name, Tasks::Context ctx)* - Task constructor used to set default parameter values from INI files
Step 2) *onResourceAquisition(void)* - Used for communication with serial ports and internet sockets, etc.
Step 3) *onResourceInitialization(void)* - Initialization of the aquired resources, determining hardware parameters such as Baud rate, start and stop bits, and parity.
Step 4) *onResourceRelease(void)* - Called at end of task
Step 5) *onUpdateParameters(void)* - Called when parameters are changed
Step 6) *onEntityReservation(void)* - Used to reserve entities and avoid name collisions
Step 7) *onEntityResolution(void)* - Resolve reserved task entities
Step 8) *onMain(void)* - Main loop for functions and code executed continuously in real-time

INI files
INI files are used to initialize and execute DUNE tasks. INI files exist under the DUNE folder ETC. Task parameters can be set and changed directly in the INI file without having to remake DUNE. An example which explains how INI files are used to initialize DUNE tasks is included in appendix section D.1.

3.5 System Objectives

In order for the proposed control system to be deemed optimal, it must meet the requirements of five key system objectives:

1. Keep USVs within acoustic transmission range of target

2. Struggle to maintain vessels in formations which minimizes associated geometric dilution of precision for the receiver array

3. Minimize distance travelled by the USVs

4. Prevent USVs from grounding and operating in restricted areas

5. Minimize radio communication

3.6 Proposed Control System

The control system presented in this section aims to optimize its performance based on the five system objectives presented in the previous section. This section briefly presents the layers of control, and in the following sections are the various control system components described in further detail. The components are described in a clockwise manner as in figure 3.12, beginning with the target observer and ending with the receiver positioning models.

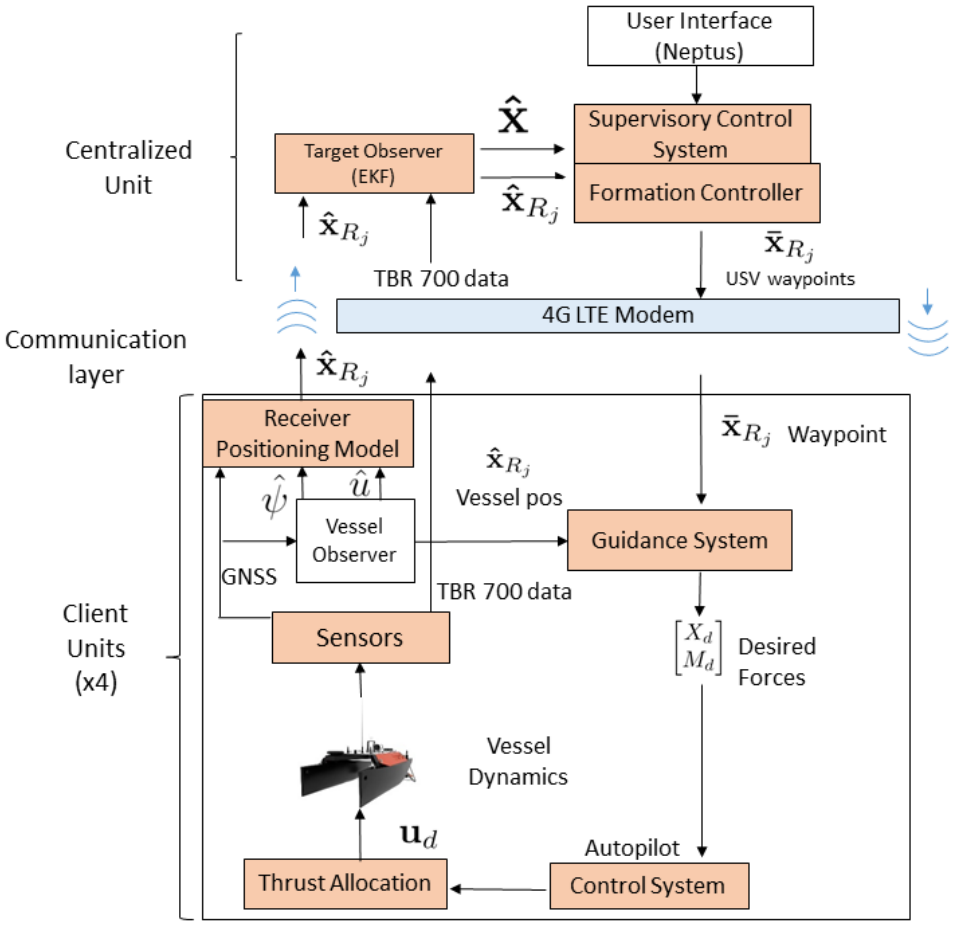

Figure 3.12: Proposed Control System

- Centralized Unit

  - Target Estimator (EKF)
  - Supervisory Control & Decision Making
  - Formation Controller

- Client Units
  - Guidance & Path-planning system
  - Autopilot & Control Laws
  - Sensor data acquisition
  - Receiver positioning model

Centralized unit
The centralized unit computer runs the distributed command and control infrastructure system Neptus, a user interface for system operators. The operators must have the ability to abort mission and retake control over the autonomous vessels throughout the entire operation. The main purpose of the centralized unit is to provide four independent guidance systems aboard the Otter USVs with waypoints, which optimizes the system objectives. The inputs to the centralized unit are TBR 700 messages and position estimates provided by GPS.

Client units
Four client units (Raspberry Pi 3 single-board computers) are distributed on four unmanned surface vessels (Otters USVs).

3.7 Target Observer

The target estimator can be view upon as the most vital part of the system, and additional emphasis put on developing and testing an estimator which enables accurate real-time positioning of a target fish.

Estimating the position of a target fish using TDOA measurements is a non-linear problem. A large variety of estimation techniques exist and could be employed to solve the problem. A robust estimation technique based on a dynamic model is preferable as signal loss and out of range situations are expected to occur frequently.

3.7.1 Kalman Filter

The Kalman filter is an optimal processing algorithm and is chosen based upon its ability to cope with noisy data. Through implementing a dynamic model can transmitter position be estimated also when only parts of the acoustic receiver data-set is available. The method is recursive which enables new measurements to be processed as they arrive, i.e real-time localization. The Kalman filter is based upon two main assumptions [33]:

1. Both the target motion and measurement model is linear

2. The probability distribution of the associated errors of the motion and measurement model are Gaussian

In the case of TDOA localization are neither the motion nor measurement model linear. However, can a variant of the Extended Kalman Filter be employed to estimate the non linear problem. [34] A drawback is that only the first order approximations of the optimal errors are provided which can lead to sub-optimal estimator performance and divergence.

3.7.2 Extended Kalman Filter

An extended Kalman filter algorithm which provides estimated solutions to the time difference of arrival equations is presented in the following section, and given in equations 3.1 and 3.2.

$$\begin{cases} \hat{\mathbf{x}}_{k+1|k} & = \mathbf{F}\,\hat{\mathbf{x}}_k \\ \mathbf{P}_{k+1|k} & = \mathbf{F}\,\mathbf{P}_{k|k}\,\mathbf{F}^T + \mathbf{Q}_k \end{cases} \tag{3.1}$$

$$\begin{cases} \mathbf{K}_k & = \mathbf{P}_{k+1|k}\,\mathbf{H}^T\,(\mathbf{H}\,\mathbf{P}_{k+1|k}\,\mathbf{H}^T + \mathbf{R}_k) \\ \hat{\mathbf{x}}_{k|k} & = \hat{\mathbf{x}}_{k+1|k} + \mathbf{K}_k(\mathbf{y} - \mathbf{h}(\hat{\mathbf{x}}_{k+1|k})) \\ \mathbf{P}_{k|k} & = (\mathbf{I} - \mathbf{K}_k\,\mathbf{H}_k)\mathbf{P}_{k+1|k} \end{cases} \tag{3.2}$$

Equation 3.1 and 3.2 are referred to as the Kalman prediction and measurement update steps, respectively. First, are the predicted state vector $\hat{\mathbf{x}}_{k+1|k}$ and projected error covariance matrix $\mathbf{P}_{k+1|k}$ calculated. In the next step are the Kalman gain $\mathbf{K}_k$, state estimate $\hat{\mathbf{x}}_{k|k}$ and error covariance matrix $\mathbf{P}_{k|k}$ updated.

The state estimate vector $\hat{\mathbf{x}}_k$ is a 6x1 vector with position and velocities estimates of the target in the ECEF coordinate system.

$$\hat{\mathbf{x}}_k = \begin{bmatrix} \hat{x} & \hat{y} & \hat{z} & \hat{u} & \hat{v} & \hat{w} \end{bmatrix}^T \tag{3.3}$$

$\mathbf{P}_{k+1|k}$ and $\mathbf{P}_{k|k}$ is the predicted and updated error covariance matrix, respectively. The values represents the confidence level of the solution.

$\mathbf{Q}_k$ is the process noise covariance matrix. The elements of the matrix amount to the associated uncertainty in the process model. The size of the matrix is equal to the number of states in the system.

$\mathbf{R}_k$ is the measurement noise spectral density matrix. The elements of the matrix amount to the associated confidence in each measurement. The size of the matrix is equal to the number of available measurements $\mathbf{R}_{MxM}$.

$\mathbf{K}_k$ is the Kalman gain matrix, which is a weighting factor matrix. The factor elements of the matrix amount to weighting factor between the filter's use of predicted state estimates and the available measurements.

$\mathbf{F}_k$ and $\mathbf{H}_k$ are the state transition and observation matrix, respectively. $\mathbf{F}_k$ is the linearized nonlinear process function matrix and $\mathbf{H}_k$ is the linearized nonlinear measurement/observation matrix, as given in equations 3.4 and 3.12,

$$\begin{aligned}\mathbf{F}_k &= \frac{\delta \mathbf{f}}{\delta \hat{\mathbf{x}}}\Big|_{\hat{\mathbf{x}}=\bar{\hat{\mathbf{x}}}_k} \\ &= \begin{bmatrix} 1 & 0 & 0 & \Delta T & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta T & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta T \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}\end{aligned} \tag{3.4}$$

where $\bar{\hat{\mathbf{x}}}_k$ is the priori state estimate at time step k and Δ T is the size of the time step,

$$\begin{aligned}\mathbf{H}_k &= \frac{\delta \mathbf{h}}{\delta \hat{\mathbf{x}}}\Big|_{\hat{\mathbf{x}}=\hat{\mathbf{x}}_\mathbf{k}^-} \\ &= M \begin{Bmatrix} \begin{bmatrix} \frac{\delta h_j}{\delta \hat{x}} & \frac{\delta h_j}{\delta \hat{y}} & \frac{\delta h_j}{\delta \hat{z}} & \frac{\delta h_j}{\delta \hat{u}} & \frac{\delta h_j}{\delta \hat{v}} & \frac{\delta h_j}{\delta \hat{w}} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ \frac{\delta h_N}{\delta \hat{x}} & \frac{\delta h_N}{\delta \hat{y}} & \frac{\delta h_N}{\delta \hat{z}} & \frac{\delta h_N}{\delta \hat{u}} & \frac{\delta h_N}{\delta \hat{v}} & \frac{\delta h_N}{\delta \hat{w}} \\ \frac{\delta \text{hae}}{\delta \hat{x}} & \frac{\delta \text{hae}}{\delta \hat{y}} & \frac{\delta \text{hae}}{\delta \hat{z}} & \frac{\delta \text{hae}}{\delta \hat{u}} & \frac{\delta \text{hae}}{\delta \hat{v}} & \frac{\delta \text{hae}}{\delta \hat{w}} \end{bmatrix} \end{Bmatrix}\end{aligned} \tag{3.5}$$

for $j \in \mathbb{J}$. $\hat{\mathbf{x}}_\mathbf{k}^-$ denotes the posterior state estimate at time step k and $\mathbb{J}$ denotes a vector of size Nx1, with N being the total number of TDOA measurements: $\mathbb{J} = \begin{bmatrix} j & \dots & N \end{bmatrix}$. hae denotes the measured depth function. M denotes the sum of TDOA measurements in addition to the depth measurement. The width of the H matrix is equal to the number of states in the state matrix; 6. The height of the H matrix, i.e M, is given by equation 3.6,

$$\text{M} = \text{N} + \text{D} \tag{3.6}$$

where N is the number of TDOA measurements and D holds the value of 0 or 1 depending of whether the depth measurement is available or not.

The measurement function elements h_j ... h_N are created using equation 3.7,

$$\begin{aligned} h_j(\hat{x}) = &\sqrt{(x_{R_j} - \hat{x})^2 + (y_{R_j} - \hat{y})^2 + (z_{R_j} - \hat{z})^2} \\ &- \sqrt{(x_{R_{ref}} - \hat{x})^2 + (y_{R_{ref}} - \hat{y})^2 + (z_{R_{ref}} - \hat{z})^2} \end{aligned} \tag{3.7}$$

for $j \in \mathbb{J}$ with $x_{R_{ref}}$, $y_{R_{ref}}$ and $z_{R_{ref}}$ being the position of a reference receiver $\mathbf{x}_{R_{ref}} \in \mathbb{R}^3$.

$$\mathbf{x}_{R_{ref}} = \begin{bmatrix} x_{R_{ref}} & y_{R_{ref}} & z_{R_{ref}} \end{bmatrix}^T \tag{3.8}$$

Most fish tags do not measure and transmit target velocity data and as a result are elements $\frac{\delta h}{\delta \hat{u}}$, $\frac{\delta h}{\delta \hat{v}}$ and $\frac{\delta h}{\delta \hat{w}}$ equal to zero for $\mathbf{h}_j$ ... $\mathbf{h}_N$.

The depth measurement is originally derived from a pressure measurement by the relationship presented in equation 3.9,

$$\begin{aligned} p_m &= \rho g z_m + p_{atm} \\ \Rightarrow z_m &= \frac{1}{\rho g}(p_m - p_{atm}) \end{aligned} \tag{3.9}$$

where p_m is the absolute measured pressure, ρ is the density of seawater and p_{atm} is the atmospheric pressure.

Assuming that the direct depth measurement is provided by the emitted fish tag signal as a negative value and that the WGS 84 frame coincides with the local sea level, is it possible to estimate the value of the height above ellipsoid directly, and a reduced observation matrix $\mathbf{H}_k$ can now be obtained as presented in equation 3.10,

$$\mathbf{H}_k = \begin{bmatrix} \frac{\delta h_j}{\delta \hat{x}} & \frac{\delta h_j}{\delta \hat{y}} & \frac{\delta h_j}{\delta \hat{z}} & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\delta h_N}{\delta \hat{x}} & \frac{\delta h_N}{\delta \hat{y}} & \frac{\delta h_N}{\delta \hat{z}} & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{3.10}$$

where the elements $\frac{\delta h_j}{\delta \hat{x}}$, $\frac{\delta h_j}{\delta \hat{y}}$ and $\frac{\delta h_j}{\delta \hat{z}}$ are given as in equation 3.11,

$$\begin{bmatrix} \frac{\delta h_j}{\delta \hat{x}} \\ \frac{\delta h_j}{\delta \hat{y}} \\ \frac{\delta h_j}{\delta \hat{z}} \end{bmatrix}^T = \begin{bmatrix} \frac{\sqrt{2}(x_{R_{ref}} - \hat{x})}{\sqrt{(x_{R_{ref}} - \hat{x})^2 + (y_{R_{ref}} - \hat{y})^2 + (z_{R_{ref}} - \hat{z})^2}} - \frac{\sqrt{2}(x_{R_j} - \hat{x})}{\sqrt{(x_{R_j} - \hat{x})^2 + (y_{R_j} - \hat{y})^2 + (z_{R_j} - \hat{z})^2}} \\ \frac{\sqrt{2}(y_{R_{ref}} - \hat{y})}{\sqrt{(x_{R_{ref}} - \hat{x})^2 + (y_{R_{ref}} - \hat{y})^2 + (z_{R_{ref}} - \hat{z})^2}} - \frac{\sqrt{2}(y_{R_j} - \hat{y})}{\sqrt{(x_{R_j} - \hat{x})^2 + (y_{R_j} - \hat{y})^2 + (z_{R_j} - \hat{z})^2}} \\ \frac{\sqrt{2}(z_{R_{ref}} - \hat{z})}{(\sqrt{((x_{R_{ref}} - \hat{x})^2 + (y_{R_{ref}} - \hat{y})^2 + (z_{R_{ref}} - \hat{z})^2}} - \frac{\sqrt{2}(x_{R_j} - \hat{z})}{\sqrt{(x_{R_j} - \hat{x})^2 + (y_{R_j} - \hat{y})^2 + (z_{R_j} - \hat{z})^2}} \end{bmatrix}^T \tag{3.11}$$

for $j \in \mathbb{J}$.

Measurement vector $\mathbf{y}$ consists of TDOA measurements and a depth measurement as presented in 3.12,

$$\mathbf{y} = M \begin{Bmatrix} \begin{bmatrix} \text{c } (\text{TOA}_j - \text{TOA}_{ref}) \\ \vdots \\ \text{c } (\text{TOA}_N - \text{TOA}_{ref}) \end{bmatrix} \\ \text{hae} \end{Bmatrix} \tag{3.12}$$

for $j \in \mathbb{J}$. Where M is the number from equation 3.6, c is the propagation speed of the acoustic signal, TOA_j is the time of arrival at receiver j, and TOA_{ref} is the time of arrival at the reference reciever. The measurement vector is naturally available only when receivers have received a signal.

The estimated measurements vector $\mathbf{h}(\hat{\mathbf{x}}_{k+1|k})$ is presented in equation 3.13,

$$\mathbf{h}(\hat{\mathbf{x}}_{k+1|k}) = M \begin{Bmatrix} \begin{bmatrix} \mathbf{h}_j(\hat{x}_{k+1|k}) \\ \vdots \\ \mathbf{h}_N(\hat{x}_{k+1|k}) \\ \hat{\text{hae}}_{k+1|k} \end{bmatrix} \end{Bmatrix} \tag{3.13}$$

for $j \in \mathbb{J}$, where $\mathbf{h}_j$ is the estimated range difference between receiver j and the reference receiver given by the measurement function from equation 3.7, and $\hat{\text{hae}}_{k+1|k}$ is the predicted depth (or height above ellipsoid) of the target provided by the Kalman filter. The predicted depth is created using the estimated target position in ECEF coordinates.

$$\hat{\text{hae}}_{k+1|k} = \frac{\hat{x}^2 + \hat{y}^2}{\cos(\hat{\mu}) - \text{N}} \tag{3.14}$$

where hae is the height above the ellipsoid assumed to coincide with the local depth, and N denotes the estimated radius of curvature in the prime vertical and estimated latitude $\hat{\mu}$ given as:

$$\text{N} = \frac{{r_e}^2}{{r_e}^2 \cos^2(\hat{\mu}) + {r_p}^2 \sin^2(\hat{\mu})} \tag{3.15}$$

$$\tan(\hat{\mu}) = \frac{\hat{z}}{\hat{x}^2 + \hat{y}^2}\left(1 - e^2 \frac{\text{N}}{\text{N} + \hat{\text{hae}}}\right)^{-1} \tag{3.16}$$

which must be solved iteratively using the method presented in equation 2.14

3.7.3 Additional Kalman Filter Functionality

The following sections emphasize upon some additional functionality which can be implemented to further enhance the performance of a Kalman filter.

Filter Algorithm
A simple filter algorithm can be implemented to prevent the Kalman filter from using malfunctioning TDOA measurements which should not be possible to obtain from a physical point of view. The distance between two receivers, i.e a pseudorange, can be estimated using data messages provided by client units. The quotient of the euclidean distance (dividend) between two receivers, divided by the speed of the signal (divisor), serves as a upper limit value for any valid TDOA measurement. A filter algorithm prevents the Kalman filter from using a measurement in which this inequality does not hold. Simply put is there an upper bound to what the value of a TDOA measurement can be, and this upper bound can be calculated using GNSS data.

Kalman Filter Tuning
Kalman filter tuning is known as the problem of adaptive filter tuning, estimating the measurement and state process noise covariance matrices $\mathbf{R}_{MxM}$ and $\mathbf{Q}_{NxN}$. M and N are the number of measurements and states in the system, respectively. Diagonal elements

in $\mathbf{R}_{MxM}$ and $\mathbf{Q}_{NxN}$ amount to the confidence associated with a measurement and uncertainty associated with a state in the process model, respectively. Kalman filters are generally tuned offline using simulated data and later used to process real data online or in real-time. In general are initial values for both $\mathbf{R}_{MxM}$ and $\mathbf{Q}_{NxN}$ set, and later adaptively updated and estimated. Tuning of $\mathbf{Q}$ values can be seen as a trade-off between retaining learning potential from measurements and valuing current filter estimates.

Data from USV data messages can be used by the Kalman filter to tune the noise spectral density matrix $\mathbf{R}_{NxN}$ in real-time. The horizontal dilution of precision (HDOP) and number of satellites provided by NMEA 0183 data are indicators of the GPS measurement quality. The SNR (signal to noise) included in TBR 700 data messages are indicators of the quality of the acoustic measurement. The Kalman filter can use this data to tune the noise spectral density Matrix R in real-time.

An analytic/practical approach can be used to determine initial values of $\mathbf{R}_{MxM}$ and $\mathbf{Q}_{NxN}$. By looking into error sources associated with the measurements and using sensor performance specifications provided by manufacturers is it possible to make some adjudicate assumptions regarding measurement accuracy and tune matrices based upon these assumptions.

Depth measurements: weighted average
Depth measurements which originate from the same transmitter ping should in theory not deviate from receiver to receiver. However, can they in practice deviate due to disturbances in the transmission channel affecting the DPPM acoustic signals. To implement a weighted average model of the depth measurements where the SNR associated with each measurement is valued is a rigours method to cope with this ambiguity.

Adaptive Filter
The Kalman filter can be designed in an adaptive way which enables the filter to cope with situations in which only a subset of the measurements are available. A rigours way of creating the Jacobian matrix $\mathbf{H}$ in situations where one or several TOA measurements are missing is presented in algorithm 1. Let the available measurements vector be denoted $\mathbf{V}$, and indicate the available measurements during a Kalman filter iteration, containing solely ones and zeros. An adaptive filter makes adjustments to the content and dimensions of the Kalman filter matrix $\mathbf{H}$ and $\hat{\mathbf{y}}$ vector, with respect to V. In addition changes the measurement noise spectral density matrix $\mathbf{R}_{NxN}$, where N is equal to the number of available measurements; or the sum of vector elements in $\mathbf{V}$. Changes in H and R inevitably changes the Kalman gain matrix $\mathbf{K}$, which is derived using this matrices. The vector containing the

estimated measurements, $\hat{\mathbf{y}}$, must match the measurements in the measurement vector $\mathbf{y}$.

Data: $\mathbf{V}$: Available Measurements Vector
$\mathbf{J}$: Adaptive Measurement Vector
$\hat{\mathbf{x}}$: Current Estimated Target Position
$\hat{\mathbf{x}}_{R_j}$: for $j \in$[1,2,3,4] Estimated Receiver Positions
Result: $\mathbf{H}$: Jacobian Matrix
createJacobian($\hat{\mathbf{x}}_{R_j}$, $\hat{\mathbf{x}}$, $\mathbf{V}$)

for $k \leftarrow 1$ **to** size($\mathbf{V}$) **do**

if $(\mathbf{V} == 1)$ **then**

$\mathbf{J} = \begin{bmatrix}\mathbf{J} & k\end{bmatrix}^T$

end

$\mathbf{H} = \text{zeros}(\text{size}(\mathbf{V}) - 1, 6)$

$\begin{bmatrix}x_{ref} & y_{ref} & z_{ref}\end{bmatrix} = \begin{bmatrix}\hat{\mathbf{x}}_{R_{J(1)}}(1) & \hat{\mathbf{x}}_{R_{J(1)}}(2) & \hat{\mathbf{x}}_{R_{J(1)}}(3)\end{bmatrix}$

for $j \leftarrow 1$ **to** size($\mathbf{V}$) $- 1$ **do**

$\begin{bmatrix}x_{R_j} & y_{R_j} & z_{R_j}\end{bmatrix} = \begin{bmatrix}\hat{\mathbf{x}}_{R_{\mathbf{J}(j+1)}}(1) & \hat{\mathbf{x}}_{R_{\mathbf{J}(j+1)}}(2) & \hat{\mathbf{x}}_{R_{\mathbf{J}(j+1)}}(3)\end{bmatrix}$

H(j,1:3) = $\begin{bmatrix}\frac{\delta h_j}{\delta \hat{x}} & \frac{\delta h_j}{\delta \hat{y}} & \frac{\delta h_j}{\delta \hat{z}}\end{bmatrix}$

$$= \begin{bmatrix} \frac{\sqrt{2}(x_{R_{ref}}-\hat{x})}{\sqrt{(x_{R_{ref}}-\hat{x})^2+(y_{R_{ref}}-\hat{y})^2+(z_{R_{ref}}-\hat{z})^2}} - \frac{\sqrt{2}(x_{R_j}-\hat{x})}{\sqrt{(x_{R_j}-\hat{x})^2+(y_{R_j}-\hat{y})^2+(z_{R_j}-\hat{z})^2}} \\ \frac{\sqrt{2}(y_{R_{ref}}-\hat{y})}{\sqrt{(x_{R_{ref}}-\hat{x})^2+(y_{R_{ref}}-\hat{y})^2+(z_{R_{ref}}-\hat{z})^2}} - \frac{\sqrt{2}(y_{R_j}-\hat{y})}{\sqrt{(x_{R_j}-\hat{x})^2+(y_{R_j}-\hat{y})^2+(z_{R_j}-\hat{z})^2}} \\ \frac{\sqrt{2}(z_{R_{ref}}-\hat{z})}{(\sqrt{((x_{R_{ref}}-\hat{x})^2+(y_{R_{ref}}-\hat{y})^2+(z_{R_{ref}}-\hat{z})^2}} - \frac{\sqrt{2}(x_{R_j}-\hat{z})}{\sqrt{(x_{R_j}-\hat{x})^2+(y_{R_j}-\hat{y})^2+(z_{R_j}-\hat{z})^2}} \end{bmatrix}^T$$

end

end

Algorithm 1: Adaptive Kalman Filter

3.8 Supervisory Control System

The Supervisory control system can be a custom-created module in Neptus. This module would support the operators during the different phases of a typical mission life cycle: planning, simulation, execution and post-mission analysis. The user interface provided by Neptus is especially important during phases of the operation in which something goes wrong and is not detected by the autonomous system. In these critical situations should the operators be able to shut down the operation and safeguard the vessels and equipment, including people involved in the operation.

3.9 Formation Control

The proposed formation control system presented in this thesis can be determined as a Single-Master Multi-Slave teleoperation of heterogeneous robots for single target tracking, which is a subset of operations within the field of Multi-agent formation control. Multi-agent formation control is a research area of which extensive work have been published in resent years, with many applications in unmanned operations. The system com-

puter which runs the formation controller and supervisory control software is the selected leader node of the system, and is autonomously coordinating and teleoperating follower robots or nodes, referred to as client nodes in this thesis, in a formation pursuing the target.

The objective of the formation control algorithm is to calculate USV waypoints which fulfills the system objectives of the system as stated in section 3.5. This section elaborates on how a formation control which compromises in an attempt to satisfy and unify these constraints in the best releasable way can be designed.

System objective 1: Keep USVs within acoustic transmission range
The first objective of the formation controller is to keep USVs within acoustic transmission range of target fish. This is an important system objective as there is no other way of localizing the object without the TBR 700 data messages. The transmission range is set to a constant approximated value in the simulations provided in this thesis, but in reality the transmission range will vary as it is dependent on the receiver, transmitter and transmission channel. The transmission channel constantly changes in time and space. An estimator should ideally estimate the transmission range throughout the operation in real-time. The estimated transmission range is an important parameter, and the system strives to keep the Euclidian distance between each receiver and the target smaller than this value. Equation 3.17 formulates the first system objective mathematically,

$$||\hat{\mathbf{x}}^E - \hat{\mathbf{x}}^E_{R_j}|| < \hat{\rho}_t \tag{3.17}$$

where $\mathbf{x}^E$ and $\mathbf{x}^E_{R_j}$ denote the position of the target and a USV in ECEF coordinates, respectively, and ρ_t denotes the estimated transmission range.

System objective 2: Minimize geometric dilution of precision
The optimal constellation with respect to geometric dilution of precision, i.e the constellation which fulfills the second system objective, is the one which minimizes the diagonal values of the matrix $\min_{\mathbf{x_R} \in \mathbb{R}^3} (\text{Tr}(\mathbf{H}^T\mathbf{H})^{-1})$. The unmanned surface vessels are restricted to the surface plane and should preferably be positioned equally distributed on a circle surrounding projection of target in the horizontal plane.

A formation reference point (FRP), as defined in a PhD thesis by Skjetne [35], is a reference point needed in order to set up a formation. The horizontal surface plane projection point of the most recent target estimate serves as a natural FRP candidate in fish tracking systems.

For the purpose of this thesis is the FPR as a moving point in $\mathbb{R}^3$ given in ECEF coordinates, indicating the centre of the desired USV formation. The FRP is defined as the horizontal projection point of the target fish estimate. In order to fulfill the second system objective, i.e form a square formation surrounding the target's projection in the surface plane, can desired position offsets for each USV be defined relatively to the FRP. How often the FRP should be updated and changed is a design parameter determining a trade-off between system tracking ability and energy-use. Let the USV formation geometry be

determined by a 4x3 offset matrix $\mathbf{O}$ in the NED frame.

$$\mathbf{O}^N = \begin{bmatrix} O_{1,1} & O_{1,2} & O_{1,3} \\ O_{2,1} & O_{2,2} & O_{2,3} \\ O_{3,1} & O_{3,2} & O_{3,3} \\ O_{4,1} & O_{4,2} & O_{4,3} \end{bmatrix} \tag{3.18}$$

The matrix values correspond to desired position offsets in directions north, east and down. The third column solely consist of zeros as the USVs are bound to the surface plane.

Combining system objective 1 & 2:
This section present a mathematical condition of which the formation controller can be designed to optimize, satisfying the two first system objectives.

Let the estimated depth be defined as a positive value and assume that the surface of the ocean coincides with the WGS84 ellipsoid. Further, allow the assumption of flat earth navigation to hold. Equation 3.19 calculates desired waypoints in the NED frame,

$$\begin{aligned} \bar{\mathbf{x}}^N_{\mathbf{R}_j} &= \hat{\mathbf{x}}^N_{,\,proj} + \mathbf{O}^N(j)^T \\ &= \hat{\mathbf{x}}^N + \begin{bmatrix} 0 \\ 0 \\ \hat{hae} \end{bmatrix} + \mathbf{O}^N(j)^T \end{aligned} \tag{3.19}$$

where $\mathbf{R}_j$ denotes the position of an Otter USV with number j = 1,2,3,4, $\bar{\mathbf{x}}^N_{\mathbf{R}_j}$ is the desired position (waypoint) of USV number j in NED frame, $\mathbf{x}^N_{,\,proj}$ is the projection of the target fish position in the horizontal plane (linearized ocean surface), $\mathbf{x}^N$ is the target fish position in NED frame and $\hat{hae}$ is the estimated depth of the target fish in NED frame (height above WGS84 ellipsoid) and $\mathbf{O}^N(j)^T = \begin{bmatrix} O_{j,1} & O_{j,2} & O_{j,3} \end{bmatrix}^T$ is row number j = 1,2,3,4 of the USV offset matrix in NED frame.

The desired USV positions, given in equation 3.19, must be transformed into ECEF coordinates and transmitted to each USV node of the system. By replacing terms $\bar{\mathbf{x}}^N_{\mathbf{R}_j}$ and $\mathbf{x}^N$ and with their respective relationships according to the flat earth equation 2.7 can equations 3.20 and 3.21 be obtained,

$$\bar{\mathbf{x}}^N_{\mathbf{R}_j} = \mathbf{R}^T(\bar{\mathbf{x}}^E_{R_j} - \mathbf{x}^E_{ref}) \tag{3.20}$$

$$\hat{\mathbf{x}}^N = \mathbf{R}^T(\hat{\mathbf{x}}^E - \mathbf{x}^E_{ref}) \tag{3.21}$$

where $\mathbf{R}$ is short for the rotation matrix $\mathbf{R}^E_N(l, \mu)$ from NED to ECEF frame as given in equation 2.8 and l and μ are estimated longitude and latitude coordinates, respectively. Let equations 3.20 and 3.21 be inserted into equation 3.19 as presented in equation 3.22.

$$\bar{\mathbf{x}}^N_{\mathbf{R}_j} = \hat{\mathbf{x}}^N + \begin{bmatrix} 0 \\ 0 \\ \hat{hae} \end{bmatrix} + \mathbf{O}^N(j)^T \implies \mathbf{R}^T(\bar{\mathbf{x}}^E_{\mathbf{R}_j} - \mathbf{x}^E_{ref}) = \mathbf{R}^T(\hat{\mathbf{x}}^E - \mathbf{x}^E_{ref}) + \begin{bmatrix} 0 \\ 0 \\ \hat{hae} \end{bmatrix} + \mathbf{O}^N(j)^T \tag{3.22}$$

The reference position $\mathbf{x}_{ref}^{E}$, i.e the centre of the flat earth linearized plane, disappears as it appears on both sides of the equation. By multiplying both sides of the equation with the rotation matrix and recalling that the matrix $\mathbf{R}$ is part of a special orthogonal group of rotation matrices denoted **SO(3)** where the following mathematical condition holds $\mathbf{R}^T\mathbf{R} = \mathbf{R}\mathbf{R}^T = 1$, can equation 3.23 be obtained.

$$\bar{\mathbf{x}}_{\mathbf{R}_j}^{E} = \hat{\mathbf{x}}^{E} + \mathbf{R}_{N}^{E}(l,\mu)\left(\begin{bmatrix}0\\0\\\hat{hae}\end{bmatrix} + \mathbf{O}^{N}(j)^{T}\right) \tag{3.23}$$

This equation calculates desired USV positions in the ECEF coordinate system. However, to satisfy the first system objective, must the equation also satisfy the mathematical inequality presented in equation 3.17. By combining equation 3.23 and 3.17 is the following obtained;

$$\left(\begin{bmatrix}0\\0\\\hat{hae}\end{bmatrix} + \mathbf{O}^{N}(j)^{T}\right) = \mathbf{R}_{N}^{E}(l,\mu)^{T}(\bar{\mathbf{x}}_{\mathbf{R}_j}^{E} - \hat{\mathbf{x}}^{E}) \tag{3.24}$$

$$\begin{bmatrix}O_{j1}\\O_{j2}\\\hat{hae}\end{bmatrix} = \mathbf{R}_{N}^{E}(l,\mu)^{T}(\bar{\mathbf{x}}_{\mathbf{R}_j}^{E} - \hat{\mathbf{x}}^{E}) \tag{3.25}$$

$$\left|\begin{bmatrix}O_{j1}\\O_{j2}\\\hat{hae}\end{bmatrix}\right| = \det(\mathbf{R}_{N}^{E}(l,\mu)^{T})(|\bar{\mathbf{x}}_{\mathbf{R}_j}^{E} - \hat{\mathbf{x}}^{E}|) \tag{3.26}$$

and by recalling that $\det(\mathbf{R}_{N}^{E}(l,\mu)^{T})$ = 1 and that the transmission inequality constraint $||\hat{\mathbf{x}}^{E} - \hat{\mathbf{x}}_{R_j}^{E}|| < \hat{\rho}_t$ must hold, can the inequality in equation 3.27 be obtained.

$$\sqrt{{O_{j1}}^2 + {O_{j2}}^2 + \hat{hae}^2} \leq \hat{\rho} \tag{3.27}$$

Let the formation controller radius in the horizontal plane be defined as in equation 3.28,

$$r_j = {O_{j1}}^2 + {O_{j2}}^2 \tag{3.28}$$

then the radius must satisfy the constraint presented in 3.29 to stay within estimated transmission range of the target fish.

$$r_j \leq \sqrt{\hat{\rho}^2 - \hat{\text{hae}}^2} \tag{3.29}$$

The finalized mathematical conditions for the formation controller to solve and optimize

are therefore:

$$\bar{\mathbf{x}}^E_{\mathbf{R}_j} = \hat{\mathbf{x}}^E + \mathbf{R}^E_N(l,\mu)\left(\begin{bmatrix}0\\0\\\hat{hae}\end{bmatrix} + \mathbf{O}^N(j)^T\right) \tag{3.30}$$

$$r_j \leq \sqrt{\hat{\rho}^2 - \hat{hae}^2} \tag{3.31}$$

for j = 1, 2, 3, 4, when the receiver is assumed to coincide with the position of the vessel in a single point.

System objective 3: Minimize distance travelled by USVs
This thesis presents three main concepts in an attempt of minimizing distance travelled by USVs: guidance radius of acceptance target, radius of acceptance and function for picking the shortest paths. The first is presented in the guidance section, while the second and third are presented followingly.

A rigorous way to limit the distance travelled by the USVs is to weight the distance between current receiver positions and desired constellation positions. Some vectors and matrices are defined followingly.

- $\bar{\mathbf{x}}$ - A matrix containing four desired constellation positions, i.e four positions in $\mathbb{R}^3$.
- $\bar{\mathbf{x}}_{R_j}$ - A matrix assigning four desired constellation positions to each USV for j = 1,2,3,4, i.e a matrix containing waypoints for each USV vessel.
- A minimize function denoted by $\bar{\mathbf{x}}_{R_j} = \min(\bar{\mathbf{x}}, \hat{\mathbf{x}}_{R_j})$ which picks distinct paths (or pseudoranges) ρ_{ij} which minimize the sum of the four paths, i.e minimizes the total distance travelled by four USVs.

USVs must be assigned distinct waypoints i = 1,2,3,4, which can be mathematically described as $i1 \neq i2 \neq i3 \neq i4$ where every i is distinct, and j = 1,2,3,4 denote numbers of four separate USVs. This constraint narrows down the possible solutions to the faculty of four (4!), i.e 24 valid solutions as i = 1,2,3,4 can be oriented in 24 different ways. The sum function is described by equation 3.32,

$$\min_{i1\neq i2\neq i3\neq i4} (\text{sum}(\rho_{i1} + \rho_{i2} + \rho_{i3} + \rho_{i4})) \tag{3.32}$$

where ρ_{ij} denotes elements in the pseudorange matrix consisting of 24 pseudoranges,

$$\rho = \begin{bmatrix} \rho_{1,1} & \rho_{1,2} & \rho_{1,3} & \rho_{1,4} \\ \rho_{2,1} & \rho_{2,2} & \rho_{2,3} & \rho_{2,3} \\ \rho_{3,1} & \rho_{3,2} & \rho_{3,3} & \rho_{3,4} \\ \rho_{4,1} & \rho_{4,2} & \rho_{4,3} & \rho_{4,4} \end{bmatrix} \tag{3.33}$$

with every element being the solution to equation 3.34.

$$\rho_{\text{ij}} = |\bar{\mathbf{x}}_i - \hat{\mathbf{x}}_{R_j}| = \sqrt{(\hat{x}_{R_j} - \bar{x}_i)^2 + (\hat{y}_{R_j} - \bar{y}_i)^2 + (\hat{z}_{R_j} - \bar{z})^2} \tag{3.34}$$

Data: $\hat{\mathbf{x}}_{R_j}$ and $\bar{\mathbf{x}}_i$
Result: $\bar{\mathbf{x}}_{R_j}$
for $i \leftarrow 1$ **to** 4 **do**
  for $j \leftarrow 1$ **to** 4 **do**
    $\rho_{ij} = \text{euclidean\_distance}(\bar{\mathbf{x}}_i, \hat{\mathbf{x}}_{R_j})$
  end
end
$\bar{\mathbf{x}}_{R_j} = \min_{i1 \neq i2 \neq i3 \neq i4} (\text{sum}(\rho_{i1} + \rho_{i2} + \rho_{i3} + \rho_{i4}))$

Algorithm 2: Function for picking shortest paths

The path picking function is presented as pseudocode in algorithm 2.

In an attempt to limit the distance travelled by the USVs can a formation radius of acceptance constraint be implemented. The formation radius of acceptance denotes a circular area of acceptance surrounding the horizontal projection of the estimated fish position in the surface plane. If the estimated target position projection moves out of this area should the formation controller assign a new FRP and new waypoints for all vessels. The radius of acceptance is a design parameter and determining its value is a trade-off between energy consumption and tracking ability of the system.

System objective 4: Prevent USVs from operating in restricted areas
Situations in which the Atlantic salmon, i.e target fish, swims along the shoreline or in water areas with shallow water or restrictions, can occur. A situation awareness model which alerts the system and switches the mode of the formation controller is needed to determine whether this is the case. This model can use USV position data and sea maps of the operation area.

Combining system objective 1, 2, 3 & 4:
The most desirable receiver constellation a circle formation surrounding the target might not be feasible as the position of the vessel is limited by the shoreline or restricted area. A formation algorithm which attempts to minimize the associated dilution of precision while being subject to limitations in positioning is presented in this section.

While being restricted to a certain area of operation can the FRP be defined as the point closest to the estimated fish position within non restricted water areas. A rigorous positioning method is to evenly distribute the vessels on a circle arc surrounding the FRP. The end points of the circle arc should preferably coincide with the intersection of a circle surrounding the FRP and the curve defining the border to the restricted area.

Final formation control algorithm

A formation controller which attempts to satisfy all system objectives, including navigation in restricted areas, is presented in this section. Firstly, some properties are defined, secondly, is pseudocode for the formation control presented in algorithm 3.

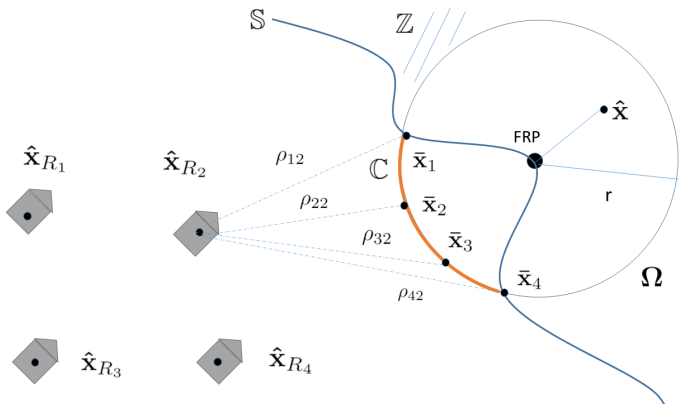

Figure 3.13: Formation control along restricted area

Properties used in the final formation controller, also indicated in figure 3.13, are defined followingly.

- Let $\mathbb{Z}$ denote an area of restricted waters in the horizontal plane in $\mathbb{R}^2$.
- Let $\mathbb{S}$ denote a curve in $\mathbb{R}^2$, in the horizontal surface plane, which defines a border between restricted and open waters.
- Let $\hat{\mathbf{x}}$ be the horizontal projection of the estimated target position in the surface plane, currently within a restricted water area.
- Let the FRP be defined as the intersection point of the orthogonal projection in $\mathbb{R}^2$ from $\hat{\mathbf{x}}$ to the curve $\mathbb{S}$.
- Let $\mathbf{r}$ denote the radius of the formation, determined by estimated transmission range and target depth.
- Let $\boldsymbol{\Omega}$ denote the horizontal circle defined in FRP with radius r.
- Let $\mathbb{C}$ be defined as the circle arc, the arc of $\boldsymbol{\Omega}$ within unrestricted waters, defined by the intersection points between $\boldsymbol{\Omega}$ and $\mathbb{S}$.
- Let ρ_j denote estimated ranges between the vessel positions and waypoints. ρ denotes a 4x4 Matrix, which corresponds to 16 pseudoranges between current vessel positions and waypoints.

The final formation controller algorithm presented in algorithm 3 assigns USVs waypoints depending on whether target position is within an restricted area $\mathbb{Z}$ or not, forming either circular or circle-arc receiver constellations to minimize GDOP.

Data: $\hat{\rho}$, $\hat{\text{hae}}$, $\mathbb{Z}$, $\mathbb{S}$, $\hat{\mathbf{x}}_{R_j}$ and $\hat{\mathbf{x}}$
Result: $\bar{\mathbf{x}}_{R_j}$
$r \leq \sqrt{\hat{\rho}^2 - \hat{\text{hae}}^2}$
if $\hat{\mathbf{x}} \in \mathbb{Z}$ **then**
  FRP $=$ OrthonogalProjection$(\hat{\mathbf{x}}, \mathbb{S})$
  $\mathbf{\Omega}$ $=$ DefineCircle$(r,$ FRP$)$
  $\mathbb{C}$ $=$ GetCircleArc$(\Omega, \mathbb{Z})$
  $\bar{\mathbf{x}}$ $=$ EvenDistrubution$(\mathbb{C})$
else
  $$\bar{\mathbf{x}} = \hat{\mathbf{x}}^E + \mathbf{R}_N^E(l,\mu)\left(\begin{bmatrix} 0 \\ 0 \\ \hat{hae} \end{bmatrix} + \mathbf{O}^N(j)^T\right);$$
end
for $i \leftarrow 1$ **to** 4 **do**
  for $j \leftarrow 1$ **to** 4 **do**
    ρ_{ij} $=$ euclidean_distance$(\bar{\mathbf{x}}_i, \hat{\mathbf{x}}_{R_j})$
  end
end
$\bar{\mathbf{x}}_{R_j} = \min_{i1 \neq i2 \neq i3 \neq i4} (\text{sum}(\rho_{i1} + \rho_{i2} + \rho_{i3} + \rho_{i4}))$

Algorithm 3: Final Formation Control Algorithm

System objective 5: Minimize radio communication

The necessary data flow in the proposed control system is quite small including only waypoints and potential abort messages from the centralized unit to clients, and TBR 700 and GNSS data in return. Communication between the centralized unit and client units is reduced by allowing the clients to operate with four independent guidance system. Proposed guidance systems are briefly introduced in the next chapter.

3.10 Guidance System

The purpose of a guidance system is to guide the path an Otter USVs towards a given waypoint, i.e assigning motion control objectives during transient motion. This section presents a guidance system which aims to satisfy system objectives 3 and 5; minimize distance traveled by vessels and communication demand. The orientation of the vessel, i.e heading angle ψ, is not affecting acoustic receiver performance due to their omnidirectional nature. An argument can therefore be made that orientation of vessels whilst not in transit between waypoints are not of interest in this fish-tracking system. However, a counterargument can be made that the receivers should be oriented towards the direction they are most likely to travel next, especially in cases of large formation radii of acceptance.

By employing the Otter USVs independent guidance systems is the need of communication between the centralized and client nodes significantly reduced. A single waypoint is the only input such a guidance system needs and no further feedback from the central-

ized node is needed until the vessel is assigned a new waypoint.

Guidance radius of acceptance
A guidance radius of acceptance, a circular area in the surface plane surrounding the waypoint of a USV, can be defined to limit energy consumption of the USVs. Whilst a formation controller radius of acceptance is taken into account by the centralized node is the guidance radius of acceptance associated with client nodes. Some operations, e.g for Floating Production, Storage and Offloading ships, demand very accurate positioning. An argument can be made that a fish-tracking system is not dependent on extremely accuracy positioning. As long as a receiver is within transmission range, and the formation of 4 USVs fairly represents a circle surrounding the target fish, deviations in dimension of several meters from the waypoints should be acceptable. Deviations in the order of 10 percent of the formation-radius can be assumed to be acceptable without compromising dilution of precision values to any significant degree. The formation radius can be assumed to be approximately 200 meters for most open ocean operations, resulting in a radius of acceptance of approximately 20 meters following this rule of thumb. A lower guidance radius of acceptance should, however, be assigned to vessels operating close to a restricted boarder.

Path parameterization and maneuvering problem
New waypoints from the formation controller are dispatched to the receivers with arbitrary intervals, and since the target position is not known apriori must the path of each Otter USV be parameterized in real-time. Two possible approaches exist for straight line parameterization to a new waypoint $\bar{\mathbf{x}}_{R_j}$:

- Parametrize from current position of the USV: $\hat{\mathbf{x}}_{Rj}$ to $\bar{\mathbf{x}}_{R_j}$
- Parametrize from the previous waypoint $\bar{\mathbf{x}}_{R_j}^{\text{prev}}$ to $\bar{\mathbf{x}}_{R_j}$

Depending on whether the first or second approach is chosen does the resulting total path consist of piece-wise linear paths and piece-wise continuous linear paths, respectively. Situations in which USVs are assigned new waypoints before having reached its previous waypoint $\bar{\mathbf{x}}^{\text{prev}}$ are likely to occur and the first approach is therefore the preferable choice for the fish tracking system through limiting the distance traveled by USVs.

The maneuvering problem for an Otter USV can be defined as the task of converging to and follow piece-wise parameterized paths until its position is within the guidance radius of acceptance. Let equation 3.35 define the geometric task, such as given in Skjetne [35], forcing the position of an Otter USV denoted $y(t)$ to converge to a desired path $y_d(\theta(t))$:

$$\lim_{t\to\infty} |y(t) - y_d(\theta(t))| = 0 \tag{3.35}$$

Straight Line Parameterization
Let $\bar{\mathbf{x}} = \begin{bmatrix} E_1 & N_1 \end{bmatrix}^T$ denote a newly assigned waypoint and $\hat{\mathbf{x}}_{R_j}^{\text{prev}} = \begin{bmatrix} E_0 & N_0 \end{bmatrix}^T$ denote the estimated position USV in the time instant the waypoint is received by the client node. The angle α_k of a straight line parameterization between these two points in a horizontal

surface plane is given in 3.36

$$\alpha_k = \arctan\left(\frac{N_1 - N_0}{E_1 - E_0}\right) \tag{3.36}$$

3.10.1 Lookahead-Based Steering

A lookahead-based steering law for 2-D horizontal plane motions as presented in Fossen [18] can be to ensure proper path following from the current position to a new waypoint. The control objective for straight-line path following is defined in equation 3.37,

$$\lim_{t\to\infty} e(t) = 0 \tag{3.37}$$

where e is the cross-track error which is the shortest distance from the current position to the path.

Let $\hat{\mathbf{x}}_{Rj}$ and $\bar{\mathbf{x}}_{Rj}$ be the latest position estimate of the Otter USV and the newly assigned waypoint given by the formation controller, respectively. A straight line in $\mathbb{R}^2$ defines a desired path in the surface plane of which the Otter USV should propagate along with. A figure showing the principle of Line of Sight (LOS) guidance is presented in figure 3.14. Δ is the along-track distance. **R** is the LOS circle of acceptance and the geomet-

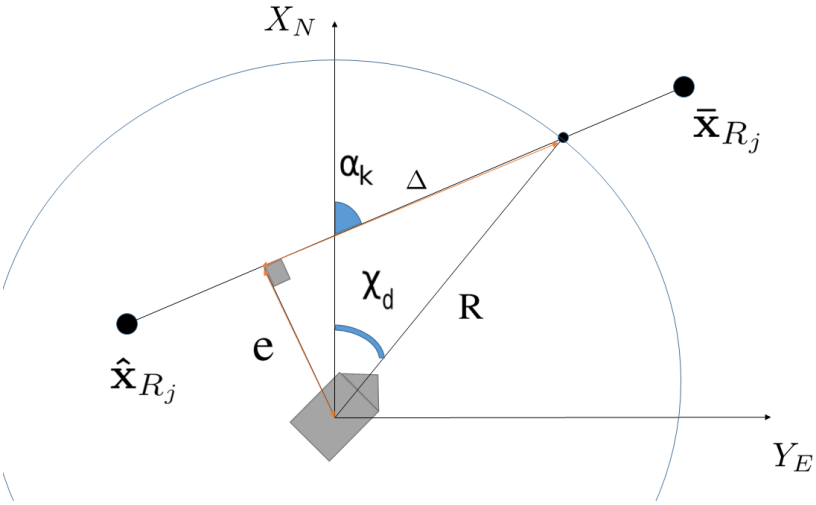

Figure 3.14: Line of Sight Guidance

ric relationship $e(t)^2 + \Delta^2 = R^2$ holds throughout the entire operation. A steering law which ensures asymptotically convergence of the cross-track error towards zero as time goes towards infinity such as indicated in equation 3.37, is presented in equation 3.38 and 3.39.

$$\chi_d(e) = \chi_p + \chi_e(e) \tag{3.38}$$

where $\chi_p = \alpha_k$ and $\chi_e(e)$ as given in the following equation.

$$\chi_e(e) = \arctan(-K_p e - K_i \int_0^t e(\tau) d\tau) \tag{3.39}$$

This steering law includes integral action which is useful for under-actuated crafts such as the Otter USVs while being subject to ocean currents and a nonzero sideslip angle β. The integral gain should however, be set with care as a high value might lead to windup effects and overshoot. According to Fossen [18] should the integral term only be used when a steady-state off-track is detected. A function for steady-state off-track detection is therefore needed if this control law were to be used in the real system.

The lookahead-based steering law is chosen because of the benefits associated with both convergence and computational intensity for this method. Lookahead-Based Steering does not demand $R \geq |e(t)|$, while other methods such as enclosed-based strategy do.

3.11 Vessel Modeling and Heading Control

Under the hypothesis that the Otter USVs are passively stable in roll ϕ, pitch θ and heave is no actuators needed nor exist to create forces in these degrees of freedom. The working space of the presented model therefore solely consists of surge, sway and yaw. If we assume that the Otter vessels have homogeneous mass distribution and xz-plane symmetry (centerline-vertical-plane) will the associated inertia matrices be equal to zero $\mathbf{I}_{xy} = \mathbf{I}_{yz}$ = 0, and surge is therefore decoupled from sway and yaw is due to symmetry. A 3 DOF nonlinear maneuvering model in the form presented in Fossen [18] is presented in equation 3.40.

$$\begin{aligned} \dot{\eta} &= R(\psi)\nu \\ M\dot{\nu} + D\nu &= \tau \end{aligned} \tag{3.40}$$

1. $\eta = \begin{bmatrix} N & E & \psi \end{bmatrix}^T$ - motions in north, east and yaw. These variables are defined in the global frame.
2. $\nu = \begin{bmatrix} u & v & r \end{bmatrix}^T$ - velocity in surge, sway and angular velocity in yaw. These variables are defined in the body frame.
3. $\tau = \begin{bmatrix} X & Y & N \end{bmatrix}^T$ - surge and sway forces and momentum in yaw. These variables are defined in the body frame. Because the Otter only have two fixed thrusters pointing in surge direction can $\tau = \begin{bmatrix} \tau_1 & 0 & \tau_3 \end{bmatrix}^T$ be obtained.
4. M - mass matrix (rigid body and added mass)
5. D - linear damping matrix
6. $R(\psi)$ - rotation matrix

$$R(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.41}$$

$$M = \begin{bmatrix} m_{11} & 0 & 0 \\ 0 & m_{22} & m_{23} \\ 0 & m_{32} & m_{33} \end{bmatrix} \tag{3.42}$$

$$D = \begin{bmatrix} d_{11} & 0 & 0 \\ 0 & d_{22} & d_{23} \\ 0 & d_{32} & d_{33} \end{bmatrix} \tag{3.43}$$

Non-linear damping effects DNL are not taken account for in this model. The velocity-components ν of the USV are assumed to be low and it is therefore fair to neglect the contribution from the Coriolis effect terms C(ν)ν = $C_{RB}(\nu)\nu$ + $C_A(\nu)\nu$.

Other models can be used to represent the dynamics of the Otters USVs, e.g the dynamic positioning (DP) model given in Fossen [18]. The DP model is valid for station-keeping and low-speed maneuvering vessels up to approximately 2 m/s which is the march speed of the Otter vessel.

3.11.1 Thrust Allocation

The principle of thrust allocation is to translate the required forces in DOF to force requirements and subsequently to setpoints for the individual thrusters. The thrust allocation is mapping $\tau \rightarrow u$.

$$\tau = \begin{bmatrix} X \\ N \end{bmatrix} \tag{3.44}$$

X is force in surge direction and N is momentum in ψ (yaw).

The Otter USV is equipped with two fixed thruster propellers in surge direction. Let $\mathbf{u} = \begin{bmatrix} u_1 & u_2 \end{bmatrix}^T$ denote the thrust force vector where u_1 and u_2 are the apparent thruster force on port and starboard side, respectively. u_1 and u_2 are equally spaced from the center line of the vessel with a $\pm$ 0.397875 meters deflection, denoted by d. 3 The Otter USV is not equipped with any actuators to produce thrust in sway direction Y and momentum in ψ direction is possible only through assigning a thrust forces difference $u_1 \neq u_2$. The resulting relationship between force in surge, momentum and thruster forces is given by equation 3.45,

$$\begin{aligned} \tau &= \begin{bmatrix} X \\ N \end{bmatrix} \\ &= \begin{bmatrix} u_1 + u_2 \\ (u_1 - u_2)d \end{bmatrix} \\ &= \begin{bmatrix} 1 & 1 \\ d & -d \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \\ &= \mathbf{T}u \end{aligned} \tag{3.45}$$

where $\mathbf{T}$ is the thrust allocation matrix. By inverting the thrust allocation matrix can the desired thrust be found, which is followingly fed to the control system.

$$\begin{aligned}
&\mathbf{T}u = \tau \\
&\Rightarrow u = \mathbf{T}^{-1}\tau \\
&\quad = \left(\begin{bmatrix} 1 & 1 \\ d & -d \end{bmatrix}\right)^{-1} \tau \\
&\quad = \left(\frac{1}{(-d)-(-d)}\right)\begin{bmatrix} -d & -1 \\ -d & 1 \end{bmatrix}\begin{bmatrix} X \\ N \end{bmatrix} \\
&\quad = \begin{bmatrix} \frac{1}{2} & \frac{1}{2d} \\ \frac{1}{2} & \frac{-1}{2d} \end{bmatrix}\begin{bmatrix} X \\ N \end{bmatrix}
\end{aligned} \tag{3.46}$$

3.11.2 PID Controller

A PID controller heading autopilot is presented in equation 3.47,

$$\tau = -K_p\tilde{\psi} - K_d\tilde{\psi} - K_i\int_0^t \tilde{\psi}(\tau)d\tau) \tag{3.47}$$

where $\tilde{\psi} = \psi - \psi_d$ and K_p, K_d and K_i the proportional, derivative and integral gain, respectively, are all larger than zero. A speed assignment in addition to a heading controller are needed to navigate a vessel, however, are speed models not examined in this thesis.

3.12 Acoustic Receiver Position Modelling

The section examines how the acoustic receiver can be positioned relative to the body coordinate system while being subject to hydrodynamic forces. The first section examines forces acting on the acoustic receiver while being submerged. The two following sections presents models for receiver positioning depending on whether the acoustic receiver is cable-connected or fixed to some non-deflecting structure.

3.12.1 Force Analysis

This section examines the forces acting on an submerged acoustic receiver in a current and wave field.

Let the acoustic receiver presented in figure 3.15 (TBR 700) be subject to a hydrodynamic forces inducing a drag force D. Let G, B and S be gravity, buoyancy and string forces, respectively. Lets assume that the receiver is not accelerating, then Newton's second law demands the sum of forces to be equal to zero in x and y direction, and relationships presented in equation 3.48 can be obtained.

$$S_y + B = G \tag{3.48}$$
$$S_x = D \tag{3.49}$$

By utilizing the relationships $S^2 = S_x^2 + S_y^2$ and $\tan(\alpha) = \frac{S_x}{S_y}$ is it possible to obtain the result presented in equation 3.12.1.

$$\alpha = \tan^{-1}\left(\frac{D}{G-B}\right) \tag{3.50}$$

By assuming a perfect cylindrical shape of the acoustic receiver can results in equation 3.51 be obtained for buoyancy and gravity force,

$$G = mg \tag{3.51}$$
$$B = \rho_w V g \tag{3.52}$$

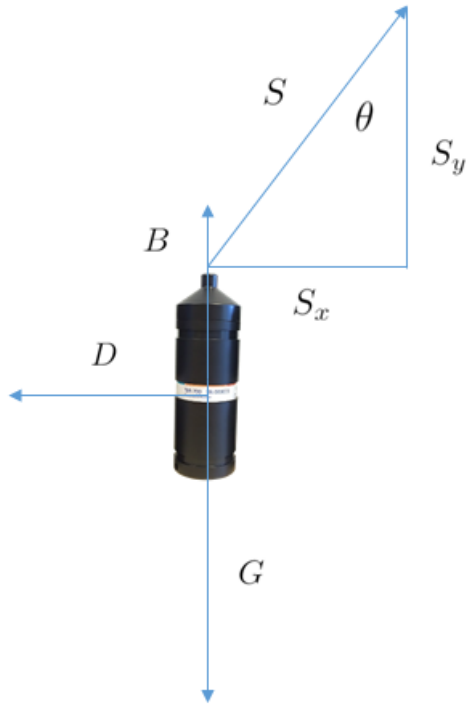

Figure 3.15: Forces acting on receiver [21]

where g is the gravity of the Earth, m is the mass of the acoustic receiver, ρ_w is the density of seawater and V is the displaced volume.

Drag Force Modeling

To estimate the hydrodynamic forces is complicated and a study field of its own. The acoustic receiver can be modeled as vertical rigid fixed circular cylinder in a incident wave field using Morison's equation, as presented in the book Sea Loads on Ships and Offshore Structures by Faltinsen [36]. Morison's equation is a semi-empirical formula used to calculate forces in the in-line direction, along the wave-direction. Elongated body, strip theory and long-wave approximation is applied in this approach. The drag force acting on a cylinder segment dz is given by equation 3.53,

$$dF = \rho\frac{\pi d^2}{4}C_m a_1 dz - \rho\frac{\pi d^2}{4}(C_M - 1)\ddot{\eta}dz + \frac{\rho}{2}C_D d|u - \dot{\eta}|(u - \dot{\eta})dz \tag{3.53}$$

where a_1 and u are the horizontal incident-wave acceleration and velocity, d is the diameter of the receiver and C_D and C_M are mass and drag coefficients, which must be estimated empirically. Typical values for C_D and C_M in transcritical flow past a smooth cylinder are 0.7 and 1.8 as given in Faltinsen [36], and is used for approximation in this example.

Transcritical flow regime corresponds to a Reynolds number larger than 3x10^3, with the Reynolds number (denoted R_n) being a measure of the importance of inertial forces relative to viscous forces. $R_n = \frac{UL}{\nu}$ where L is the characteristic body length of the receiver, ν is the kinematic viscosity and U the flow velocity. $\dot{\eta}$ and $\ddot{\eta}$ denote the velocity and acceleration in surge direction of the acoustic receiver. As equation 3.53 indicate is the drag-force contribution connected with the relative velocity between incident waves and acoustic receiver.

Assume no incident waves, i.e $a_1 = 0$, constant surge speed u of the cylinder, i.e $\ddot{\eta} = 0$ and $\dot{\eta} = u$, and a constant current β in negative surge direction, i.e u = β. Equation 3.54 presents an expression for the drag force under these assumptions,

$$dF = \frac{\rho}{2} C_D d|\beta - u|(\beta - u)dz \tag{3.54}$$

where $\beta - u$ is the relative speed between surge speed u and current speed β.

Approximate Inclination Angle
This example aims to approximate the displacement angle of an acoustic receiver cable-connected to an Otter USV propagating at speed u. A speed dependent model for the inclination angle α is presented.

Lets assume no current, transcritical flow and that the assumptions resulting in equation 3.54 hold. Let the acoustic receiver be connected to a cable of length l and assume that no drag forces are induced by the cable itself. The drag force acting on a cylinder segment dz is then given by equation 3.55.

$$dF = -\frac{\rho}{2} C_D du|u|dz \tag{3.55}$$

The diameter, height and weight of the receiver in freshwater is 0.075 m, 0.23 m and 0.26 kg. Weight in water G_w multiplied by the gravity of the Earth corresponds to the difference in weight and buoyancy force $G - B$, given in equation 3.12.1. Density of freshwater is 1000 kg/m^3. Let C_D = 0.7 be the proposed drag coefficient for smooth cylinders as given in Faltinsen [36]. By integrating the dz force over the height of the cylinder in equation 3.56 can the total drag force F be found.

$$F = \int_0^h dF dh \tag{3.56}$$

$$= \int_0^h -\frac{\rho}{2} d C_D u|u|dh \tag{3.57}$$

$$= |u|u \int_0^h -\frac{\rho}{2} d C_D dh \tag{3.58}$$

$$= -6.0375|u|u \tag{3.59}$$

Lets assume the receiver connection-cable to be straight throughout the entire operation with a length l. Further assume that the relationship from equation holds. The inclination angle of the receiver cable can then be found by equation 3.60.

$$\alpha = \tan^{-1}\left(\frac{F}{G-B}\right) \tag{3.60}$$

$$= \tan^{-1}\left(\frac{F}{G_w g}\right) \tag{3.61}$$

$$= \tan^{-1}\left(\frac{-6.0375N|u|u}{0.26kg9.81\frac{m}{s^2}}\right) \tag{3.62}$$

$$\tag{3.63}$$

An empirical model for alpha is given in equation 3.64,

$$\alpha = \tan^{-1}\left(-2.3671\left[\frac{s^2}{m^2}\right]u|u|\right) \tag{3.64}$$

where u is the relative flow speed between the receiver and current, and $|u|$ is its absolute magnitude.

3.12.2 Cable-connected Positioning Model

A position model of a submerged cable-connected acoustic receiver such as indicated in figure 3.5 is presented in this section. The model assumes that the cable is straight and that momentum is not transferred through the cable, only axial forces in the z body frame direction. The deflection of the receiver is given as a point relative to the centre of flotation of the vessel. The flotation point is positioned using relative positioning with respect to a GNSS aboard the vessel. The GNSS has a fixed relative position to the CF and CO, with the latter serving as the centre of the body coordinate system.

Positioning Model

- Let a point $\mathbf{x}_c^b = \begin{bmatrix} x_c & y_c & z_c \end{bmatrix}^T$ be a fixed point in the body frame on an Otter USV in which the cable connecting the acoustic receiver is attached to. From now on is this point referred to as the connection point. x_c, y_c and z_c represents constant displacements in x,y and z direction parallel to the body axis' of the USV.
- Let the horizontal plane which intersects the connection point $\mathbf{x}_c^b$ in z direction be referred to as the connection plane.
- Let a line which is parallel to the cable-connector and intersects the connection point $\mathbf{x}_c^b$ be referred to as the connection line.
- Let the projection of the acoustic receiver's position in the connection plane be referred to as the connection projection point.

- Let a line between the connection point and the connection projection point be referred to as the projection line.
- Let α denote the smallest angle between the horizontal plane and the connection line. The vertex of this vertical angle is the connection point $\mathbf{x}_c^b$.
- Let ψ_{rel} denote the angle between the longitudinal body axis of the USV (aft to fore) and the projection line.

A proposed model of the displacements the the origin of an acoustic receiver denoted $\mathbf{x}_a^b = \begin{bmatrix} x_a & y_a & z_a \end{bmatrix}^T$, relative to the flotation point CF, is presented in equation 3.65. The model is on the form: $\mathbf{x}_a^b$ = CF + MODEL.

$$\mathbf{x}_a^b = \text{CF} + \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ x_c\theta + y_c\phi \end{bmatrix} + \begin{bmatrix} \cos(\psi_{rel})\cos(\alpha)(l + z_c) \\ \sin(\psi_{rel})\cos(\alpha)(l + z_c) \\ \sin(\alpha)(l + z_c) \end{bmatrix} \tag{3.65}$$

l is the length of the cable and θ and ϕ are the pitch and roll angles of the vessel, respectively. The flotation point CF is the point of which the vessel pitch and roll around, indicated in figure 2.1. Assuming that the cable is straight with a length l, and $\mathbf{x}_c^b$, ψ_{rel} and α are known exact can the cable-connected receiver hypothetically be positioned relative to the USV body without errors using equation 3.65.

Position Model with no current

In the case of no current can a simplified speed model be obtained. With no current is the ψ_{rel} expected to be 0 or 180 degrees depending on whether the USV is propagating in positive or negative surge direction. This is due to the fact that the drag force points in opposite direction of the surge direction when there is no current. Lets assume the USV only propagate in positive surge direction. When these assumptions hold can a simplified speed model based upon the original model in equation 3.65 be obtained:

$$\mathbf{x}_a^b = \text{CF} + \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ x_c\theta + y_c\phi \end{bmatrix} + \begin{bmatrix} \cos(\alpha)(l + z_c) \\ 0 \\ \sin(\alpha)(l + z_c) \end{bmatrix} \tag{3.66}$$

Reduced Positioning Model

With the current sensor configuration aboard the Otter USVs is there no way of measuring the horizontal or vertical cable angles ψ_{rel} and α, neither the vessel orientation angles θ and ϕ. A reduced version of the model presented in equation 3.65 is presented in equation 3.67.

$$\mathbf{x}_a^b = \text{CF} + \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} + \begin{bmatrix} \cos(\alpha)(l + z_c) \\ 0 \\ \sin(\alpha)(l + z_c) \end{bmatrix} \tag{3.67}$$

The proposed model can use the alpha angle model presented in equation 3.64 with $\alpha = \tan^{-1}(-2.3671u|u|)$.

Conversion to ECEF coordinates

The target observer needs position estimates of acoustic receivers in ECEF coordinates in order to estimate target position. The position of a body-fixed GNSS receiver aboard the vessel is denoted $\mathbf{x}^b_{\text{GPS}}$The position an acoustic receiver relative to the CO is given by equation 3.68,

$$\mathbf{x}^b_a = \text{CO} + \text{LCF} + \text{MODEL} \tag{3.68}$$

where LCF is a vector pointing from the CO to CF point. The position of the acoustic receiver is given by the equation 3.69.

$$\mathbf{x}^b_a = -\mathbf{x}^b_{\text{GPS}} + \text{LCF} + \text{MODEL} \tag{3.69}$$

The position transformation of an object from body to NED and NED to ECEF are given by equations 2.5 and 2.9, and the position of the acoustic receiver in ECEF coordinates is given by equation 3.70.

$$\mathbf{x}^E_a = \mathbf{x}^E_{\text{GPS}} + \mathbf{R}^E_N(l,\mu)\mathbf{x}^N_a \tag{3.70}$$

$$= \mathbf{x}^E_{\text{GPS}} + \mathbf{R}^E_N(l,\mu)\mathbf{R}^N_b(\boldsymbol{\Theta})(-\mathbf{x}^b_{\text{GPS}} + \text{LCF} + \text{MODEL}) \tag{3.71}$$

As the current configuration of the Otters have no way of measuring pitch, roll, or current can a simplified model can be used,

$$\mathbf{x}^E_a = \mathbf{x}^E_{\text{GPS}} + \mathbf{R}^E_N(l,\mu)\mathbf{R}^N_b(\hat{\psi})(-\mathbf{x}^b_{\text{GPS}} + \text{LCF} + \text{MODEL}) \tag{3.72}$$

where MODEL = Cable Model + Receiver Model, as the dynamic forces will act on both the cable and receiver body. However for the purpose of this thesis lets assume that the cable dynamics can be neglected, and that the dynamics of the receiver can be described by the simplified straight cable with vertical axial forces - model proposed in equation 3.67.

$$\text{MODEL} = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} + \begin{bmatrix} \cos(\alpha)(l+z_c) \\ 0 \\ \sin(\alpha)(l+z_c) \end{bmatrix} \tag{3.73}$$

By employing the estimated α model from equation 3.64 is a proposed cable-connected receiver model presented in equation 3.74,

$$\hat{\mathbf{x}}^E_a = \mathbf{x}^E_{\text{GPS}} + \mathbf{R}^E_N(l,\mu)\mathbf{R}^N_b(\hat{\psi}) \left(-\mathbf{x}^b_{\text{GPS}} + \text{LCF} + \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} + \begin{bmatrix} \cos(\tan^{-1}(-2.3671\hat{u}|\hat{u}|))(l+z_c) \\ 0 \\ \sin(\tan^{-1}(-2.3671\hat{u}|\hat{u}|))(l+z_c) \end{bmatrix} \right) \tag{3.74}$$

where $\hat{u}$ and $\hat{\psi}$ is the estimated surge speed and heading, respectively, $\mathbf{R}^N_b(\phi,\theta,\psi)$ is the body to NED rotation matrix given in equation 2.6, $\mathbf{R}^E_N(l,\mu)$ is the NED to ECEF rotation matrix given in equation 2.8 and $\mathbf{x}^E_{\text{GPS}}$ is the direct measurement provided by Garmin 18x-5Hz GPS aboard each Otter USV.

Inclination angle at cruising-speed

At cruising speed of the Otter USVs 2 m/s is an inclination angle of -1.466 radians expected, corresponding -83.971 degrees, using equation 3.64 and the relationship rad =

$\angle^{\circ}\frac{\pi}{180}$. This result indicates that drag induced forces on the receiver can create a relative cable-angle α between negative surge direction and depth of approximately 90 degrees. At angles close 90 degrees α is the receiver barley submerged, which is undesirable due to a variety of reasons. The acoustic receivers or cables can intertwine with the rotating Torqeedo propellers which is naturally unacceptable as equipment can be damaged. Concerns related to high noise levels close to the surface, induced by propeller-cavitation or breaking waves, further stresses the importance of connecting the receiver to the USV in a way which ensures sufficient submergence throughout the entire operation. Connecting additional weight to either the receiver or the cable is a possible solution. Additional weight will however effect the speed performance of the Otter USVs negatively. This problem can also be solved through employing a fixed receiver configuration.

3.12.3 Fixed Positioning Model

Imagine a construction fixing the acoustic receiver to the Otter. The construction can be created a stainless steel material or other appropriate material for seawater interaction. In the following models is its material assumed to be non deflecting, i.e the hydrodynamic forces acting on the construction and receiver does not bend the material.

Let $\mathbf{x}_f^b = \begin{bmatrix} x_f & y_f & z_f \end{bmatrix}^T$ be the point of which the centre of the acoustic receiver is fixed to relative to the centre of flotation in the body frame. This point is defined relative to the flotation point CF, x_c, y_c and z_c represents constant displacements in x,y and z direction parallel to the body axis' of the USV, given in equation 3.79.

$$\mathbf{x}_a^b = \text{CO} + \text{LCF} + \mathbf{x}_f^b \tag{3.75}$$

Conversion to ECEF coordinates
The real position of the acoustic receiver is given by equation 3.76,

$$\mathbf{x}_a^E = \text{CO}^E + \mathbf{R}_N^E(l,\mu)\mathbf{R}_b^N(\phi,\theta,\psi)\left(\text{LCF} + \mathbf{x}_f^b\right) \tag{3.76}$$

where the position of the receiver is decomposed in principle rotations around the body axis' of the vessel; θ, ϕ and ψ and CO^E is the origin of the body coordinate system in ECEF coordinates (as presented in Fossen [18]). Because the craft pitch and roll around the CF (flotation point) can $\mathbf{R}_b^N(\phi,\theta,\psi)$ be simplified to $\mathbf{R}_b^N(\psi)$ for the LCF term,

$$\mathbf{x}_a^E = \text{CO}^E + \mathbf{R}_N^E(l,\mu)\mathbf{R}_b^N(\psi)\text{LCF} + \mathbf{R}_N^E(l,\mu)\mathbf{R}_b^N(\phi,\theta,\psi)\mathbf{x}_f^b \tag{3.77}$$

θ and ψ angles are not measurable by the current sensor configuration and only rotations around the z principle axis are therefore considered. For a catamaran vessel types such as the Otter USVs which have a large metacentric height are angular displacements in roll significantly reduced. If roll and pitch deflections are neglected can the model in 3.78 be obtained.

$$\mathbf{x}_a^E = \text{CO}^E + \mathbf{R}_N^E(l,\mu)\mathbf{R}_b^N(\psi)(\text{LCF} + \mathbf{x}_f^b) \tag{3.78}$$

The location of CO must be positioned relative to a GPS receiver aboard the vessel. Equation 3.79 positions an acoustic receiver relative to the a GPS, CO and CF.

$$\mathbf{x}_a^b = -\mathbf{x}_{\text{GPS}}^b + \text{LCF} + \mathbf{x}_f^b \tag{3.79}$$

A fair assumption, is that the receiver is fixed to the xz plane (body coordinate system), i.e positioned along the center-line of the vessel, $y_f^b = 0$. Other configurations where $y_f^b \neq 0$ are undesirable as hydrodynamic forces acting on the receiver and construction will induce a momentum in yaw direction ψ, i.e corrupting vessel straight-line stability. A final fixed-configuration position model is presented in equation 3.80.

$$\hat{\mathbf{x}}_a^E = \mathbf{x}_{\text{GPS}}^E + \mathbf{R}_N^E(l,\mu)\mathbf{R}_b^N(\hat{\psi}) \left(-\mathbf{x}_{\text{GPS}}^b + \text{LCF} + \begin{bmatrix} x_f^b \\ 0 \\ z_f^b \end{bmatrix} \right) \tag{3.80}$$

Chapter 4

Implementations

The practical implementations presented in this thesis involves the integration of a GPS receiver (Garmin 18x 5Hz), CAN controller (PiCAN2), motor controller (Torqeedo interface board) and Torqeedo thrusters with a Raspberry Pi 3 single-board computer. The delivery time of the Otter USVs unfortunately was delayed due to a fire in the manufacturer warehouse. The vessels are expected to be delivered during the summer of 2018, i.e prior to the master's thesis project of this author. Work and tests on the vessels, e.g payload installation and ocean testing, are therefore not subject of this thesis. Because the acoustic receivers TBR 700 have proven successful in other NTNU projects (e.g [14], [15] and [9]) is testing and calibration of the receiver not a subject of this thesis.Time synchronization of the receivers is the main theme of the master's thesis by Efteland [14], and therefore neither examined significantly in this thesis.

4.1 Hardware

4.1.1 GPS

A Garmin GPS 18x-5Hz is integrated with a Raspberry Pi 3 single-board computer using its GPIO (general-purpose input/output) interface, enabling the devices to communicate using UART. A test run revealed successful real-time positioning at NTNU Gløshaugen, Trondheim, Norway. The Garmin 18x-5Hz GPS cannot be connected directly to the GPIO pins on the Raspberry Pi and receive meaningful data as the devices do not operate on similar voltage levels. A Mikroelektronika MAX3232 RS-232 converter board can be used as voltage converter device. Solder tin is a fusible metal alloy which creates a permanent bond between metal work pieces and wires. A soldering iron is used to create the necessary electrical connections for the GPS, with more practical instructions including wiring and communication schematics included in appendix section B.3.

A variety of software alterations in the core of the Raspbian (Linux distribution) operative system running on the Raspberry Pi 3 are necessary to enable communication with the

Garmin GPS, and detailed instructions are included in the appendix section B.3. Real-time GPS data acquisition in DUNE is enabled using the DUNE task [Sensors.GPS], which decodes NMEA0183 data and dispatches GPS data messages to the IMC bus.The baud-rate of the Garmin GPS is specified to its correct value of 19200 Baud in its initialization file, and more details regarding GPS data acquisition are included in appendix section B.3. RS-232 converter board connections are indicated in figures 4.3 and 4.4.

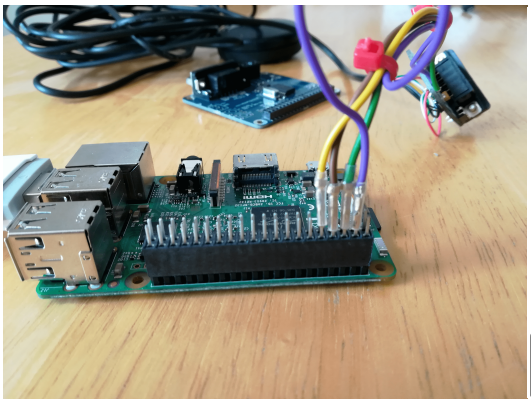

Figure 4.1: GPS Experiment setup

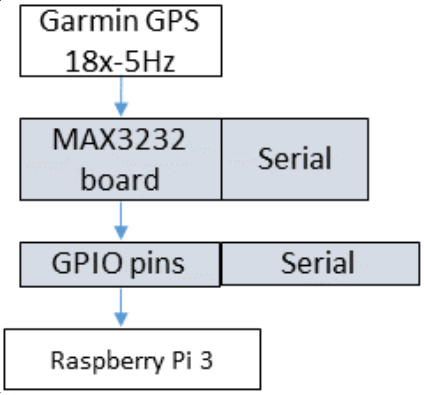

Figure 4.2: GPS setup schematics

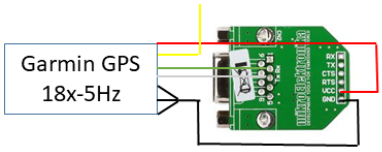

Figure 4.3: GPS Connection I [25]

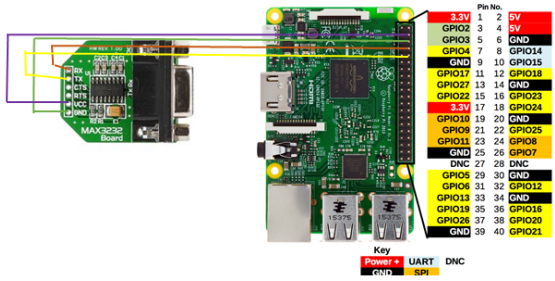

Figure 4.4: GPS Connection II [25] [37]

4.1.2 Controller Area Network

This section presents a possible hardware configuration which enables the autonomous vessel to control its thrusters by adjusting the pulse-width modulated voltage. A Raspberry Pi 3 single-board computers communicates with a motor controller board, i.e Torqeedo interface board, using Controller Area Network (CAN) bus communication. A CAN controller, i.e PiCAN2, provides CAN-bus capability to Raspberry Pi 3.

A variety of software and hardware implementations are necessary to enable thruster communication and are explained in further detail in appendix section B.4. First, a CAN interface in the single-board computer is set up to enable CAN communication. Secondly, a 120Ω Terminator solder jumper is inserted terminating both ends of the PiCAN2. Fi-

nally, the PiCAN2 is connected to the GPIO pins of the single-board computer and by a CAN cable to the Torqeedo interface board.

Can messages are sent to the Torqeedo board on its hexadecimal address 0xAB, and are sequentially ordered in a little-endian format in which the most significant byte, i.e containing the lowest address value, is sent first. Seven various messages types from the board exist whereas only two types of messages can be sent to the board. Messages from the board contains information of various system components such as current, voltage, uptime, power, temperature, battery capacity and error messages among others. Messages to the board are sent to initialize and set throttle value for both motors.

Thrust is set as an integer value in the interval -1000 and 1000 indicating maximum thrust in forward (surge) and backwards (-surge) direction, respectively. A single CAN message sets desired thrust for both propellers in the following manner:
00 D0 FE AB C8 00 C8 00
where 00 D0 corresponds to the ID of thrust messages (D = 13) and AB referrers to the address of the board. The first out of two C8 00 messages corresponds to 200, i.e one fifth of maximum thrust, for the first thruster, and similarly for the second. Full thrust is given by 03 E8 (1000). By setting these two values to distinct values can a momentum in the ψ (yaw) direction be created, and hence adjust the heading of the vessel.

4.1.3 Thruster test

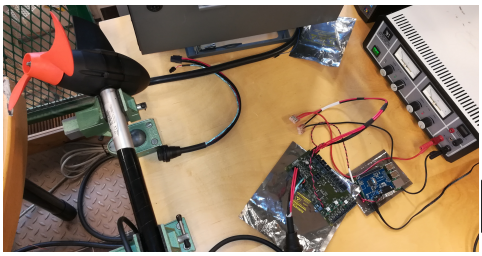

Figure 4.5: CAN interface testing with thruster

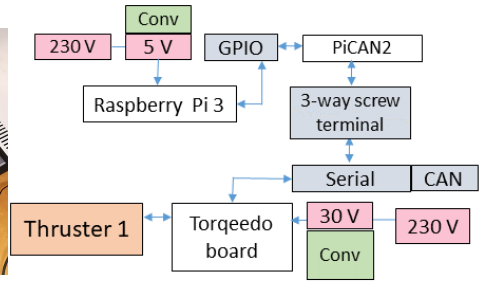

Figure 4.6: Lab setup schematics

A Torqeedo thruster with supplementary hardware setup is tested in ”forsøkshallen” experiment lab, elektrobygg D at NTNU Glshaugen, Trondheim, Norway. Thrust commands is set in Raspberry Pi 3 with propeller response, i.e Torqeedo thruster and hardware system behaves satisfactory. A picture and schematics of the lab setup is included in figure 4.5 and 4.6, consisting of four main components; Raspberry Pi 3, PiCAN2, Torqeedo interface board and a Torqeedo thruster.

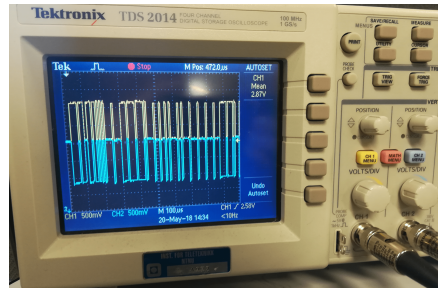

Figure 4.7: Digital Oscilloscope Analysis: CAN signal from PiCAN2 to Torqeedo board

A EA-PS 7032 100 power supply (0-32 Volts, 0-10 Ampere) is powering the Torqueedo board, and normal 5 Volt cellphone charger powers the Raspberry Pi and PiCAN2 shield. The thruster is powered directly by the Torqueedo interface board. Please note that the thruster is equipped with a tilt sensor which prohibits propeller rotation for tilt angles larger than about 20 to 30 degrees. Thrust can therefore not be given in a variety if configurations, e.g the initial configuration presented in 4.5, and a lot of time was therefore spent error searching prior to knowing this fact in the lab.

Messages sent from the Torqueedo board are analyzed using the candump function in Raspberry Pi terminal. All messages from the board, i.e messages with ID 3, 6, 10, 12, 14, 15, and 16 are successfully received by the Raspberry Pi. A digital oscilloscope, i.e Tektronix TDS 2014, is used to analyze and verify high and low CAN signals provided by the PiCAN2 shield. Figure 4.7 displays the differential signaling nature of CAN bus signals. An analysis of the data from the terminal dump and digital oscilloscope shows that signals from and to the board are successfully transmitted and received with correct voltage levels for both high and low CAN signals. A simple python script provided by the manufacturer is ran to set throttle vales for the thruster with satisfactory thruster response.

4.2 Software

Modem
The Huawei E3372 4G LTE modem is successfully tested to be directly compliant with the Raspberry Pi 3, and can therefore be used for inter-vehicle communication in client nodes. SD and SIM cards are however needed to get internet capabilities.

Structs
Various C++ structs are defined to enable data storage in DUNE. In C++ is a struct a group of data elements stored under a common name. Data structures are created for the target fish, USVs, TOA, TDOA and USV data messages. A struct containing Kalman filter data using the predefined Matrix-Class in DUNE, i.e in src/DUNE/Math/, is created.

Functions
Some of the most important functions used by the TOA data generator and Kalman Filter tasks are explained briefly in this section. A list of the most important functions follows:

- Functions for importing CVS (.txt) data.
- Functions for transforming vectors from LLH and ECEF coordinates, based upon the DUNE task WGS84.
- Data string parsing of client node messages containing position estimates of the vessel and TBR 700 data
- Kalman filter functions creating the Jacobian matrix **H** and estimated measurement vector $\hat{\mathbf{y}}$.

DUNE Tasks

Data Message Creator (TBR 700)
This DUNE task calculates transmission times of an acoustic signal to multiple receiver nodes, and dispatches messages to the IMC-bus in real-time. This task converts comma-separated values (CSV) files, i.e comma-delimited text (.txt) files, into TBR 700 data messages. Messages are dispatched to and available on the IMC bus with realistic timing, i.e as they would in a real system. This task can be used in simulation phases to analyze performance and tracking ability of an unmanned tracking system using TBR 700 receivers. The task is tested and verified in a simulation presented in chapter 5. The task simulation speed is an adjustable variable which allows the user to have results created and debug messages printed to the console (terminal) in any speed desirable. 1 indicates normal speed whereas 100 would speed up the time of the simulation by a factor of 100.

The CVS files must contain position data of receivers and transmitter and be stored in a specific data format. This data format is described in the appendix in section D.2. Position data must be in ECEF or LLH coordinates. Developers can also change the data format specifications by manipulating the data storage functions presented in 4.2.

Let a data message containing TBR 700 acoustic data and the associated ECEF position be referred to as a client node data message. The DUNE task dispatches client node data messages to the IMC bus on the form;

$TBR02, 1521394325, 253, S256, 2, -19.999420, 50, USV2, 2811417.889201, 515365.680904, 5682720.780954

where the comma-separated elements correspond to: acoustic receiver ID, timestamp in seconds (unixtime) and milliseconds, code type, tag ID, pressure data (depth), signal to noise ratio, USV receiver ID and x,y and z in ECEF coordinates, respectively. Please note that the timestamp is simplified to a single double value containing both seconds an milliseconds in the simulation presented in chapter 5 and corresponding code in appendix

section D.1.

Real-time Extended Kalman Filter
A DUNE task which estimates the position of a target fish in real-time by consuming client node data messages is implemented. Position estimates of the target are dispatched to the IMC bus in ECEF or LLH coordinates and other navigation tasks in DUNE, e.g a formation controller, can use the data as input. The performance of the Extended Kalman filter is verified and tested with fake-data in chapter 5. The filter can hypothetically be used in a real fish-tracking experiment.

Chapter 5

Simulation

Goal

The following simulation aims to verify and test the performance of the real-time Kalman filter implemented in DUNE using generated TBR 700 data.

Time and place

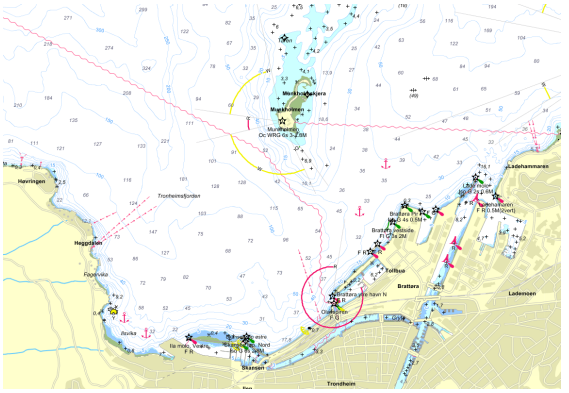

Figure 5.1: Trondheim Location ([38])

Figure 5.2: Trondheimsfjord Sea Map ([39])

A hypothetical tracking simulation is set to be located in the Trondheimsfjord, indicated by the map in figure 5.2. The initial position of a target fish is set to 10.382645 °E and 63.450705 °N in longitude and latitude coordinates. Four USVs initially form a quadrilateral shape surrounding the targets projection on the surface plane with an offset matrix $\mathbf{O}_{Rj}$ as indicated in equation 5.1,

$$\mathbf{O}_{Rj} = \begin{bmatrix} -250 & 250 & -350 & 250 \\ 350 & -250 & -250 & 250 \end{bmatrix}^T \tag{5.1}$$

where matrix elements corresponds to offsets in North and East direction in the unit of meters. The initial time of the simulation is set to Wed, 04 Apr 2018 04:04:04 (+0000), or 04/04/2018 @ 4:04am universal time (UTC), which corresponds to Unix timestamp 1521394325. Unix time is equivalent to the number of seconds since +0000 first of January 1970 (UTC).

5.1 Data

The fake position data used in this simulation has its origin from a fish tracking simulation in the project thesis paper *Examining use of various vehicles in a single fish tracking system* [13] written by this author in December 2017. The original data set consists of position values in xyz coordinates (three-space) for a target fish and four USVs. The duration of the simulation is 6000 seconds with a time-step of 0.5 seconds, resulting in 12000 data points for each degree of freedom, i.e 36000 data points per vessel.

The four USVs are tracking the target using a formation controller and straight-line guidance. The final fish trajectory is created by piece-wise sinusoidal trajectories and a slowly varying bias model. The target fish moves along the trajectory with varying speed propagating mainly in one direction relatively close to the surface. The bias model is a 1st order Markov process in $\mathbb{R}^3$. This method creates a trajectory which seemingly represent "the natural behavior of a fish" with some randomness introduced. A slowly varying Markov process is created by allowing white noise to travel through a low-pass filter as presented in equation 5.2,

$$\dot{\mathbf{x}}(t) = \frac{-1}{T}\mathbf{x}(t) + \mathbf{w}(t) \tag{5.2}$$

where $\dot{\mathbf{x}} \in \mathbb{R}^3$ is the velocity of the target, $\mathbf{x} \in \mathbb{R}^3$ is the position of the target, T is the Markov time constant and $\mathbf{w} \in \mathbb{R}^3$ is Gaussian white noise.

The data is manipulated from its original format and stored in comma-separated values (CVS) text-files (.txt) enabling testing of the developed C++/DUNE software presented in this thesis. Data is firstly transformed into the NED frame from xyz coordinates by modifications where x corresponds to North, $-$ y to East and $-$ z to Down in the NED-frame. The data is converted from the NED frame to both ECEF and LLH coordinates utilizing the relationships presented in section 2 in a MATLAB script. Position data in ECEF and LLH coordinates are saved in CVS text-files (.txt): usv_pos_ecef.txt, usv_pos_llh.txt, target_ecef.txt and target_llh.txt.

5.2 Assumptions

Four USVs are assumed to host acoustic receivers with positions coinciding with vessel point-mass in $\mathbb{R}^3$. USV position and orientation estimates are assumed to be perfect, and velocity and heading directly manipulative $u = \frac{dx}{dt}$, $v = \frac{dy}{dt}$ and $w = \frac{dz}{dt}$. Acoustic receiver clocks are assumed to be perfectly synchronized.

A fish tag tagged to a target fish is transmitting a signal every 0.5 seconds, i.e with a frequency of 2 Hertz, propagating at the speed of sound in seawater set to a constant 1484 meters per second. Depth measurements are transmitted with the acoustic signal and assumed to be available without any delays except the transmission time of the signal itself. The WGS-84 ellipsoid is assumed to coincide with the ocean surface and depth measurements are therefore direct measurements of the height above ellipsoid (hae), with depth defined as a negative property.

Radio communication links between client nodes (computer aboard USVs) and the centralized computer hosting the target observer are assumed to be stable and instantaneous.

5.3 Software

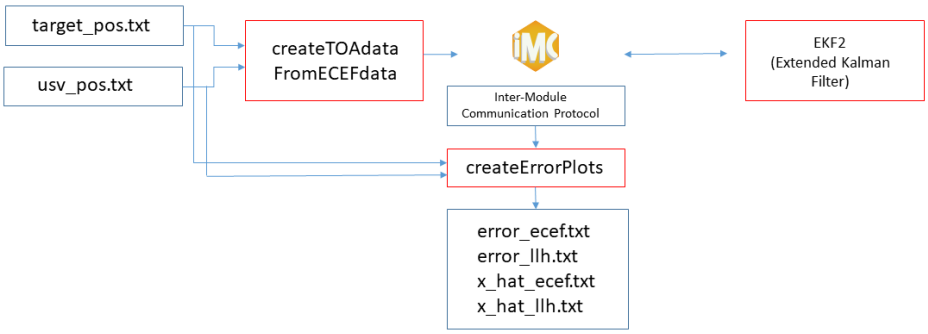

Figure 5.3: DUNE fish tracking simulation: data flow

Figure 5.3 indicates the implemented software and data-flow during simulation. DUNE tasks are indicated by red boxes and communicate using the IMC protocol.

Two DUNE tasks createTOAdataFromECEFdata and EKF2 operate like producer and consumer tasks, respectively. A third DUNE task *createErrorPlots* creates CVS error data using estimated and true target position values prior to a simulation. Estimated target positions and errors in ECEF and LLH coordinates are stored in text files. DUNE tasks use various C++ functions for calculations which are defined in the header and source file *funkytions.h* and *funkytions.h*, respectively. The data format, DUNE tasks, INI files and C++ functions used in the simulations are described in further detail in the appendix sections D.2, D.3, D.4 and D.5.

The acoustic data obtained by a client node is combined with the latest receiver position estimate provided by a Garmin 18x-5Hz GPS and receiver positioning model, and sent as a single IMC message (IMC::DevDataText) on the form presented followingly.

$TBR0x, 1446716612, 123, S256, 2, 233, 50, USVx, 2811674.127, 515158.467, 5682591.148

where x is the receiver and USV number x = 1,2,3,4. The first part is acoustic TOA data similar to the form of TBR 700 data messages as presented in 3.3.1, whereas the last part are the estimated receiver position in ECEF coordinates. Spacing between commas are not included in the real messages.

The Kalman Filter is tuned and designed in a way where depth measurements are valued with a higher priority than the TDOA measurements. More specifically is the associated depth measurement value in the noise spectral density matrix R lower than the values associated with the TDOA measurements.

5.4 Results

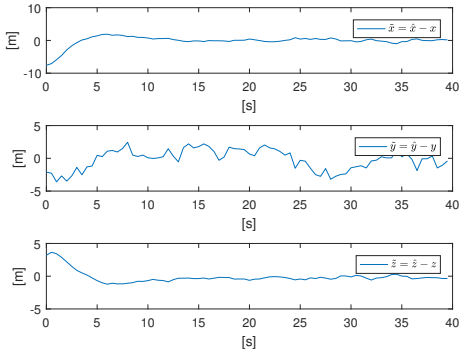

Figure 5.4: ECEF Error Plots

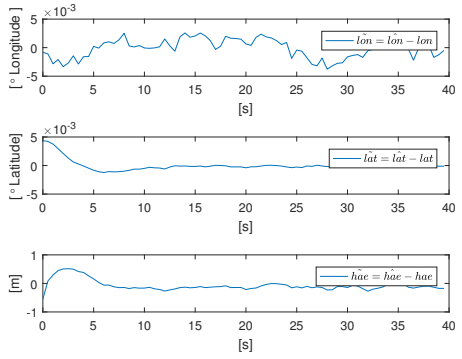

Figure 5.5: LLH Error Plots

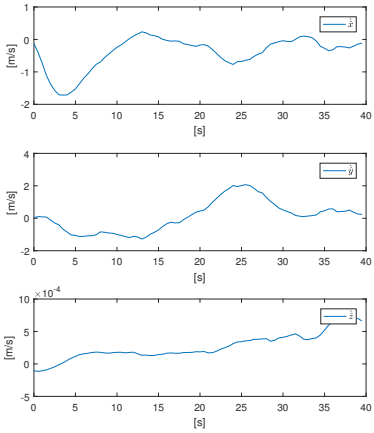

Figure 5.6: ECEF Velocity Estimates

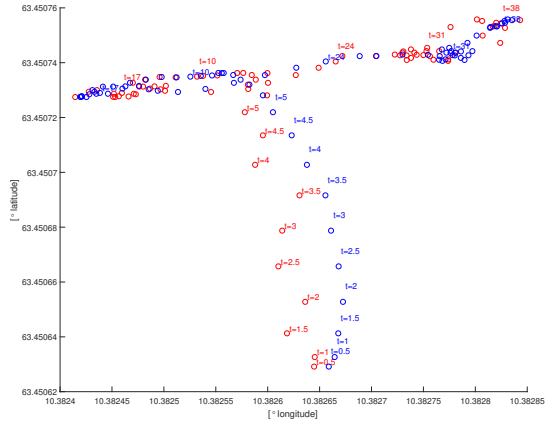

Figure 5.7: North-East Plot

The true errors in ECEF coordinates are plotted in figure 5.4. The initial errors are approximately 10, 2, and 5 meters in x,y and z direction, respectively. The plots indicate that the Extended Kalman filter is converging, i.e the total error is decreasing with time. The error in y direction is however somewhat fluctuating (noisy).

The true errors in LLH coordinates, longitude and latitude degrees and height above WGS 84 ellipsoid (meters) are presented in figure 5.5. The errors in hae direction and latitude are quickly converging towards zero from their initial offset of approximately 0.5 meters and 5x10^{-3} degrees latitude. The Kalman filter consumes depth measurements which enables accurate positioning in the hae direction, and the results shows that the error in hae direction is small throughout the entire simulation.

Estimated velocities in x,y and z direction in ECEF coordinates, i.e $\hat{u}$, $\hat{v}$ and $\hat{w}$, are presented in figure 5.6. Error plots are not included because the original data-set only includes position values. The velocity estimates indicates a target fish which initially propagates in $-$x direction and mainly in +y direction towards the end of the simulation. Angular-velocities in longitude and latitude, and velocities in hae can be derived using ECEF velocity estimates and the relationships presented in equations 2.11, 2.12, and 2.13.

Figure 5.7 presents 2D scatter plots of the true and estimated target trajectories in a North - East coordinate system, where red and blue color indicate real and estimated position, respectively. The axis' indicate longitude and latitude degrees, and scatters are plotted with additional text indicating the time associated with the specific positions. Time-plots indicate time past since the beginning of the simulation.

Chapter 6

Discussion

The research questions presented in section 1.5 are discussed and answered in this chapter. The chapter also includes an evaluation of the validity of underlying assumptions in the developed models. Finally, suggestions for further research is presented.

6.1 First Research Question

How can important control system components of an optimal fish tracking system be developed?

A proposed control system unifies five system objectives in a single formation control algorithm, and is designed in an adaptable way in which receiver are positioned in a tactical manner depending on whether the target is located close to the surface, at great depth, close to the shoreline or in an open ocean environment. Both the proposed guidance system and formation controller contribute to create a more reliable control system by allowing vessels to operate close to a restricted boarder and be less dependent on inter-vehicle communication, respectively. A target estimator and receiver positioning models are specifically designed to increase the accuracy of the target estimates.

The following paragraphs elaborate upon why the system objectives are chosen and how a formation controller attempts to unify them, as presented in algorithm 3. First, a recap on what the system objectives are is presented:

1. Keep USVs within acoustic transmission range of target
2. Struggle to maintain vessels in formations which minimizes associated geometric dilution of precision for the receiver array
3. Minimize distance travelled by USVs

4. Prevent USVs from grounding and operating within restricted areas
5. Minimize radio communication

The first objective is chosen based upon the fact that there is no way of using the developed TDOA estimator with all receivers being outside transmission range. A dead-reckoning estimator can be implemented in an attempt to reallocate the target in such situations. Dead-reckoning is implemented in this thesis through allowing the developed Kalman filter to simply integrate the latest velocity estimates of the target fish. However, dead-reckoning navigation is assumed to be somewhat unlikely of reallocating a target fish in an open sea environment, which emphasizes the importance of keeping receivers within transmission range. The second system objective is satisfied by positioning USVs in circular and circle-arc shaped formations depending on whether the target is located within a restricted area or in the open ocean, surrounding the horizontal target projection point FRP. In this way are receivers obtaining configurations which minimizes geometric dilution of precision and maximizes quality and resolution of target geospatial data. Receiver constellations which have a large formation radius are closer to realizing the optimal receiver array arrangement.

System objectives one and two are partly contradictory through demanding a small and big formation radius, respectively, and combining them in a unified formation control algorithm is therefore a trade-off. Maximum formation radius is dependent on estimated transmission range and estimated target depth, as given in equation 3.17. Hypothetically, as a target fish dives to great depth or transmission range decreases USVs are forced to move significantly closer towards the formation reference/centre point (FPR), i.e decrease formation radius to stay within transmission range. In contrast, increased formation radius is allowed whilst the target is located close to the surface.

The third system objective is to minimize distance traveled by vessels, which includes preventing USVs from performing unnecessary station-keeping. This system objective is particularly important for long-endurance tracking missions as the batteries aboard the Otter USVs have limited power. These objectives can be satisfied through implementing concepts presented in this thesis: target & guidance radius of acceptance and a path picking function. The first and second allow estimated target and USV positions to deviate from their respective FPR and waypoint localizations within a limited area without assigning new waypoints or demanding station-keeping, respectively. Tuning the radii of acceptance is a trade-off between energy-use, GDOP and tracking ability of the system. Positioning within guidance radii of acceptance is assured locally by client nodes. Ideally, the centralized system should be able to tune the guidance radii of client nodes independently through inter-vehicle communication. In general, the guidance radii of acceptance should be smaller in situations where the vessels are close to one another or close to restricted areas. The path picking function consequently assigns USVs paths which minimize the distance sum traveled by vessels by weighing the distance from their current positions.

The fourth objective is to prevent USVs from operating in restricted areas, e.g shoreline or harbour areas. The objective is important for obvious reasons; prevent the vessels and surrounding environment from taking damage from collisions, and allow tracking near restricted areas. This can be implemented using the concept presented in figure 3.13 and

algorithm 3, using sea maps of the sea-bottom and surrounding environment in areas of operation. Straight-line guidance is often not feasible whilst operating in restricted areas and methods for curved path-following and terrain avoidance must be implemented. This will eventually allow vessels to navigate parallel along restricted area boarders to their waypoints provided by algorithm 3.

The fifth system objective is to minimize the use of radio communications. Radio communication can be costly, difficult to implement and the reception of mobile communications is often limited in ocean areas. Vulnerability to slow communication links can be reduced for a fish tracking system by assigning the suggested guidance system presented in this thesis. The need for inter-vehicle communication is significantly reduced by allowing client nodes (Otter USVs) to operate with independent guidance systems. Communication from a client node to the centralized node is solely necessary when the client node has received an acoustic measurement. The acoustic data is decoded and combined with the latest receiver position estimate, provided by a Garmin 18x-5Hz GPS and receiver positioning model, and sent as a single IMC message on the form as presented in 5.3. Communication from the centralized node to client nodes is only necessary when the system picks a new FRP assigning new vessel waypoints. Accurate formation tracking cannot be expected for this system as vessels are likely to be oriented with different heading angles whilst performing station-keeping, and expected to be subject to independent disturbances during transit. Client nodes are therefore not expected be aligned with their respective paths nor arrive at new waypoints simultaneously. A more complicated formation control algorithm, e.g assigning trajectory tracking and speed control objectives, can be implemented to ensure more accurate formation tracking. This, however, introduces a higher demand for feedback from the centralized computer. To minimize the demand for inter-vehicle communication and achieve accurate formation tracking are therefore to some extent contradictory objectives.

The Kalman filter implemented in DUNE converges and delivers satisfactory position estimates of a target fish using generated TBR 700 data in real-time, and can therefore be used in a real experiment. Results presented in section 5.4 and figures 5.4, 5.5, 5.6 and 5.7 indicate convergence and successful real-time localization in a simulation located in the Trondheimsfjord. The errors are very small throughout the entire simulation, with errors in estimated height above WGS 84 ellipsoid (hae) in the order of decimeters. The developed DUNE software can be used to simulate tracking scenarios, test system performance, and tune estimator and formation control parameters prior to a real experiment. The proposed additional estimator functionality presented in section 3.7.3, i.e adaptiveness, filtering algorithm, filter tuning and weighted average depth method, can be implemented to further enhance the filter's ability to deliver robust and accurate position estimates.

Two models for a body fixed and cable-connected TBR 700 receiver are presented in this thesis, and can be used in a real system depending on the final connection configuration. This thesis argues that better positioning, but more noise, can be expected from a fixed receiver configuration compared to a cable-connected alternative. This argument is mainly based upon the assumption that an acoustic receiver mounted by a stiff construction (e.g

metal rod) fixed to the Otter USV is significantly more affected by relative motion with water, due to deflections in sync with vessel roll and pitch periods. A cable-connected configuration can be beneficial in sea states in which large roll and pitch deflections are expected. These aspects are further elaborated upon in the following section.

6.2 Second Research Question

What are the error sources associated with the proposed system?

This chapter presents some of the expected error sources for the proposed fish tracking system presented in 3. Error sources are in this context defined as all phenomena which prohibits the localization estimator from accurately determining the true position of its target. Two types of measurements are used in positioning of a target fish: TDOA and depth measurements. The following sections presents the error sources associated with both types of measurements and to some extent also their significance. The expected performance of various system components and sensors and their implications on the estimator are discussed in the following section.

A variety of error sources are expected to limit the accuracy of TDOA and depth measurements. The sources of error can be divided into three main groups; **1) environmental-, 2) acoustic receiver-, 3) transmitter - related errors**.

1) Environmental Error Sources
Environmental error sources can be divided into subgroups **i) transmission path -** and **ii) noise** - related errors.

1) i) Transmission Path
Transmission path errors are mainly associated with varying signal speed and multipath phenomena. The position algorithm (EKF) assumes uniform transmission paths with a constant signal speed and errors in the TDOA measurements are created by differentiating TOA measurements associated with unmodeled dynamics. In reality transmission paths are unique as acoustic signals pursue independent trajectories through the water column before reaching receiver destinations. Transmission path variations in both time and space are expected. The speed of an acoustic signal in water varies non-linearly with properties such as salinity, pressure and temperature, with Del Grosso's formula being a sufficient approximation displaying this relationship. Large variations in water properties are expected in operations located close to a river estuary, e.g in a salmon tracking operation, with temperature and salinity differences between freshwater outlets and seawater expected to be high.

Transmitted signals can differ with various acoustic receivers due to reflection patterns from surrounding objects and the sea bed causing multi-path interference. timestamping by first wave-front is a method for guaranteeing accuracy, or a consistent margin of error, in acoustic communication. This is because straight paths are shorter than any transmission paths associated with multi-path.

1) ii) Noise
The TBR 700 hydrophones use advanced digital signal processing and dynamically adjust threshold levels to optimize reception and maximize noise reduction. Noise in the environment is expected to limit the performance of acoustic receivers and potentially jeopardize their ability in decoding DPPM signals and timestamping correctly. In acoustic communications literature, e.g the book Autonomous Underwater Vehicles by Burrows and Khan [40], environmental noise in an acoustic environment is normally divided into three groups: ambient or background noise of the ocean, self-noise of the vehicle and intermittent noise (including biological noises). Self-noise and intermittent noise levels are expected to be high as a result of phenomena such as breaking waves and rain.

In a study performed at NTNU the transmission range of the acoustic signal was lower than expected and previously observed, with the reason assumed to be noise from an intensive rain shower. [9] Results showed that the reception of an acoustic receiver on a USV platform is significantly influenced by the depth of the submerged hydrophone, indicating better performance with increased depth. The study showed that a system performed satisfactory if the USV travelled at speeds below 3 knots, despite noise generated by the propeller and travelling hull being one of the major concerns ahead of the field trail. By employing tracking missions on days with calm weather and no rain associated with sea states with small significant wave heights, intermittent noise from breaking waves and rain, and self noise levels induced by hull slamming are expected to be low.

A book about the fundamentals of acoustics from 1982 [41] states that self-noise from cavitation effects on the propeller is significant for higher frequencies. In this context, high frequencies are noted as acoustic signals with a frequency higher than 10 kHz. As most acoustic transmitter signals have a mean frequency of about 70-80 kHz, self noise from propellers are expected to have an impact on the performance of the acoustic receivers. Cavitation levels and associated induced noise is expected to be high for the propellers aboard the Otter USVs, as they operate closely to the surface. Cavitation, defined as the process of air bubble generation on the propeller blades, is a pressure related phenomena which decays with increasing depth. This stresses the importance of prohibiting the receivers from operating close to the propellers.

Water moving past an acoustic receiver is assumed to induce noise with a magnitude (decibel) increasing with the relative speed difference between the receiver and surrounding water. This stresses the importance of choosing a connection configuration which reduces displacements of the acoustic receiver and relative speed difference with water.

A cable-connected acoustic receiver can be assumed to be less subject to noise than a fixed configuration through being less affected by rapid water-interaction induced by roll and pitch rotations of the vessel. Deflections in heave direction (z) due to surface waves are instantaneous in upwards direction (pull force) in a cable-connected configuration. Deflection in downwards direction (USV moving from wave crest to wave trough) are damped due to a ”sink delay” or inertia phenomena as pressure forces (push) are not converted

through a sufficiently elastic cable. The sinking latency phenomena can be expected when the cable is not entirely straight (stretched). Sinking latency can be positive as the relative speed of water passing over the receiver surface is damped, compared to a fixed configuration. However irregular bursts of activity, by fits and starts, due to sudden cable stretch from wave-induced heave motion can induce high noise-levels. A cable-connected acoustic receiver is not affected by roll and pitch rotations of the vessel if momentum is assumed to not be transferred through the cable. Pitch and roll rotations could however induce a pull force in upward direction unless the cable is connected in the flotation point of the vessel, which is a preferable connection point reducing noise.

2) Acoustic Receiver Errors
Acoustic receiver - related errors mitigating the accuracy measurements can be divided into subgroups of **i) positioning, ii) timestamping** and **iii) time-synchronization**.

2) i) Acoustic receiver positioning
Position estimates of the acoustic receivers are directly used by the extended Kalman filter to estimate the target fish position, and positioning errors are therefore directly influencing its performance. The position of the acoustic receiver is defined relative to the body of the USV vessel body coordinate system. The error in acoustic receiver positioning can be considered as the sum of GNSS positioning errors and errors in positioning the receiver relative to the GPS.

The UERE(User Equivalent Range Error) of the Garmin 18x-5Hz GPS is less than 15 meters 95% of the time according to the manufacturer. [24] Other enhancement techniques for more accurate GNSS positioning, e.g RTK, could be employed as the current GPS configuration is one of the most significant error sources in the system.

The error in positioning of a cable-connected receiver relative to the connection point in the body coordinate system is only limited by the length of the cable itself. In configurations in which the system has no possibility of measuring current direction or amplitude, which is the case for the current sensor configuration aboard the Otter USVs, the errors in positioning can grow significantly large. In tracking scenarios in which there is no current, e.g tracking in a lake, a surge-model can give accurate estimates of the receiver position if the hydrodynamic forces are modeled accurately. Hydrodynamic forces, are however, difficult to model as they are dependent on many variables which are difficult to measure, such as the Reynolds number. Other phenomena, such as vortex induced vibrations, are also expected to affect the movement of the receiver. The error in positioning of a cable connected acoustic receiver can be reduced if the receiver is connected to a heavy device, or to a heavy cable, as the range of possible cable-angles α are reduced due to increased gravity.

Good positioning precision is expected for a fixed receiver as it is not expected to be significantly deflected by hydrodynamic forces. The apparent deflection is determined by the construction's ability to resist deformation (stiffness) and magnitude of the applied hydrodynamic forces. Deflections in heave direction (z), e.g due to surface waves, are

instantaneous and in sync with the movement of the craft. An acoustic receiver connected to a fixed structure is subject to deflections in the x and y direction in the body frame induced by roll and pitch rotations of vessel, and are proportional with the length (or depth) of the fixed construction. A deeply submerged receiver therefore comes with the cost of larger deflections in the horizontal plane due to roll and pitch. By applying sensors for measuring roll and angles the deflections can be compensated for using equation 3.76, however the current configuration of the Otter USV payload does not include such sensors.

2) ii) timestamping
As addressed and stated in Efteland [14] the reception resolution of the TBR 700 receivers is about 1 ms, due to a cycle time of 1 ms while in listening mode. timestamps are found to be within 1 ms 90 % of the time. A 1 ms error margin corresponds to about 1.5 meters error in positioning with a signal speed of 1500 meters per second.

2) iii) Time-synchronization
As presented in a NTNU master thesis by Efteland [14] TBR 700 receivers can be time-synchronized using GPS technology through integrating a GPS module with the TBR board. GPS technology can provide time synchronization down to 60 nanoseconds, which is sufficient with respect to the TBR 700 receiver resolution of 1 ms. [14] The Garmin 18x-5Hz GPS outputs time and date in coordinated universal time (UTC) data and could hypothetically be used for time-synchronization. According to the manufacturer the timing precision is limited to a tenth-of-a-second. [24] The accuracy of Garmin 18x-5Hz time measurements is therefore clearly insufficient for the purpose fish tracking, as an acoustic signal can travel up to 150 meters within the time-interval of a-tenth-of-a-second. Other solutions and sensor configurations must therefore be considered. The use of a surface support module (SSM), GPS/Tinymesh expansion circuit board and a Raspberry Pi 3 single-board computer, as presented in Efteland [14], can be considered.

3) Transmitter Error Sources
The total errors associated with depth measurements can be considered as the sum of the depth measurement offset and depth signal resolution. According to the manufacturer of ADT-16 the fish tag should be sensitive to depth changes down to as little 1 cm, with a maximum depth measurement offset of $\pm$ 50 cm. [23] Measured depth is continuously encoded as the time delay between two acoustic pulses (DPPS) by a piezoelectric transducter, and the resolution of the transmitted signals are 10 cm corresponding to 10 ms. A delay of 1000 ms corresponds to 0 m increases with 100 ms per meter, e.g if the target is located at a depth of 100 meters will DPPM signals be transmitted with a delay of 1000 ms + 100ms * depth = 11 seconds. The transmitted DPPM will be affected by changes in acoustic signal speed due to transmission path variations for two subsequent waves reaching the same receiver after a time delay. Transmission variations in the relevant time spans are however expected to be small and should not limit the accuracy of the depth measurements in any significant way.

6.3 Validity of Underlying Assumptions in Receiver Positioning Models

This section aim to evaluate the validity of underlying assumptions leading to the acoustic receiver positioning models presented in section 3.12. Firstly, the hydrodynamic force analysis presented in section 3.12.1 is discussed. Secondly, additional assumptions resulting in the cable-connected positioning model are presented.

Force analysis assumptions
In general the force analysis presented in this thesis is very simplified, and a more detailed study should be conducted if more accurate results are needed. Models are, however, based upon a force analysis using data from the manufacturer, and estimates of the buoyancy force should therefore be quite accurate. The receiver weight in water (buoyancy subtracted by gravity force) provided by the manufacturer is given in freshwater, and small adjustments (in order of ≈ 2.5 %) are therefore expected in seawater due to changes in water density. The induced drag forces, however, are severely simplified by a variety of assumptions which are stated followingly:

1. Long-wave approximation
2. Receiver fixed with respect to incoming current field
3. Receiver perfectly cylindrical with suffciently smooth surface
4. Transcritical flow regime
5. No forces from vortex-induced vibration phenomena acting on receiver

The validity of the assumptions is expected to be high for 1,2 and 3, and low for 4 and 5, based upon arguments presented followingly. Long-wave approximation can be used when the incident waves tend to be unaffected by the interaction with a structure. Faltinsen [36] states that long-wave approximation can be used when $\lambda \geq 5D$, i.e the wavelength of the incident waves are five times longer than the diameter of the acoustic receiver. This corresponds to a wavelength of 0.375 meters in the case of TBR 700 RT. This holds for the majority of sea states expected during operation as most sea states are associated with wave-lengths larger than 0.375 meters. The force analysis assumes that the acoustic receiver is fixed and not tilted with respect to the incoming current-field. Small tilt-angles are not expected for a fixed receiver-configuration unless pitch and roll angles of the vessel are very large, whereas tilt-angles for a cable-connected configuration is expected to be significantly larger due to receiver and cable deflection. Faltinsen, however, states that the semi-empirical formula Morison's equation can be used also in situations where the cylinder is tilted [36], which further validates the assumption for both configurations. Receiver positioning models assume that the receiver has a perfect cylindrical shape with a sufficiently smooth surface, such as presented in Faltinsen, and that the associated drag-coefficient (also presented in Faltinsen) can be applied. [36] The real receiver is quite close to having a perfect cylindrical shape, and is expected to have a relatively smooth surface. The model assumes a transcritical flow regime, whereas in a real experiment the

flow regime surrounding the receiver is expected to be weather dependant and constantly changing. The validity of this assumption is therefore low. The vortex-induced vibration (VIV) phenomena is not taken into account when developing the receiver positioning models.

Cable-connected model assumptions
Additional assumptions are necessary for obtaining the cable-connected positioning model presented in equation 3.74. The model neglects the effects of weight and drag induced forces on the cable connecting the receiver to the USV. However, cable induced drag forces can grow severely large with increasing diameter of the cable, and should be modeled for better positioning. The model assumes the cable to be straight, whilst it in real life is expected to bend with increasing depth in a current field. The model assumes that both pull and push axial forces are conducted through the cable line, and that no momentum is transferred through the cable. In real life only stretch forces are expected to be conducted through the cable.

6.4 Suggestions For Further Research

- Implement the remaining DUNE tasks towards finalizing the proposed control system in section 3.6, this involves all control system components except the target estimator.
- A variety of practical tasks remain towards finalizing the system and can be conducted as soon as the Otter USVs are delivered. This involves payload implementation in the vessels and deciding upon a receiver configuration. In determining whether a fixed or a cable-connected configuration is more beneficial for the Otter USVs some of the arguments presented in this thesis can be taken into account.
- Examine and develop anti-collision for the Otter USVs in DUNE. A function which aims to guarantee non-intersecting USV paths with a sufficient margin distance between vessels should be implemented to further prevent the likelihood of collisions. This function can incrementally evaluate the feasibility of USV path-combinations from shortest to longest, using the presented path picking function in equation 3.32, and assign the first combination which satisfies inter-vehicle distance requirements. Such a function would have to consider the speed performance and orientation of each vessel in anticipating their paths.
- Convert the script provided by maritime robotics giving thrust commands to the Torqeedo thruster from python to C++ and implement it in DUNE.
- Examine methods for station-keeping of the Otter USVs. A loitering-like station-keeping technique can perhaps be used to keep USVs within guidance radius of acceptance whilst being subject to current. Implementing a bow-thruster in sway-direction on the Otter USVs will allow the vessel to be less under-actuated, decoupling momentum in yaw and surge force, and dynamical positioning with higher precision can be expected. In this configuration will Otters USVs be able to rotate without creating large horizontal circles.

- Increase system integrity by creating user interface error messages in Neptus based upon data returned by the Torqeedo interface board. The Torqeedo board returns a variety of messages, e.g status, power levels, temperature, safety and error flags for thrusters and batteries, and the client nodes could notify the system operators by forwarding important error messages through 4G communication.

- Study feasibility of various guidance methods close to the shoreline and terrain, and implement a suitable method in DUNE which allows maneuvering along a restricted between waypoints provided by algorithm 3.

- Develop a model for estimating acoustic transmission range in real-time. The output of this model would be used directly used by the proposed formation controller presented in this thesis. This model can use data such as pseudorange estimates, available measurement vector (i.e indicating receivers which have received acoustic signal) and SNR-ratio.

- Examine benefits and possibility of using a hybrid localization estimator, based upon both the principle of TDOA and signal strength localization, as presented in 2.2.4. In general there is more uncertainty involved with using a signal strength based technique in comparison with TDOA localization, with only the latter being implemented in this thesis. Combinations of utilizing both techniques could potentially increase the robustness of an estimator and result in enhanced tracking performance during critical phases of operation in which few receivers are within transmission range. Pseudorange estimation based upon signal strength is difficult in marine environments due to a varying transmission channel and background noise. Through having obtained TDOA measurements for some time, however, a model of measured signal strength (or SNR) as a function of range can be derived. Such a model and its parameters can be tuned in real-time.

- In this thesis, dead-reckoning navigation is implemented through allowing the Kalman filter to simply integrate the latest velocity estimates of the target. However, dead-reckoning can be implemented in a more advanced manner, e.g a tactical search pattern for moving targets or based upon some apriori knowledge (biological model) of where the fish is likely to be located. However, apriori knowledge about the target fish is often very limited and to some extent also what the system is supposed to deliver. A more detailed study examining the feasibility of already-existing search methods within other fields, e.g avalanche search rescue, should be conducted as they possibly could be employed in fish tracking.

Chapter 7

Conclusions

This thesis contributes to the development of an autonomous fish tracking system for Atlantic Salmon through presenting various hardware and software implementations and a proposed control system. However, a variety of tasks remain to be solved towards finalizing the proposed fish tracking system, due to the limited time of this master's thesis and vast number of tasks which have to be conducted. The developed models presented in this thesis can be used as inspiration for other peers working on the NTNU fish tracking project, e.g future master's and PhD students. The final system can be used to monitor both hatchery farmed and wild Atlantic salmon in an open ocean environment, and scientists can use the data to develop a better understanding of the species and the way it interacts with its environment. Scientist can further use this data to examine the exposure of the aquaculture industry on wild stocks, and tactically narrow down the recapturing search area in the event of fish farm escapes.

A target positioning algorithm based upon the principle of time difference of arrival positioning (multilateration), i.e an extended Kalman filter using data from four TBR 700 acoustic receivers, is implemented and tested in a unified navigation environment (DUNE). The results reveal filter convergence and successful real-time localization of a target fish in a simulation located in the Trondheimsfjord. The developed DUNE software can be used to simulate tracking scenarios, test system performance in critical phases of operations and tune estimator and formation control parameters, prior to a real experiment. Proposed additional functionality, i.e adaptiveness, filtering algorithm, filter tuning and weighted average depth method, can be implemented to further enhance the filter's ability to deliver robust position estimates. Positioning models for a cable-connected and fixed acoustic receiver configuration are presented with the intention of accurately position the acoustic receiver relative to a GPS aboard the vessels, whilst being subject to hydrodynamic forces. The position estimates of receivers are directly used by the target estimator, hence the proposed models can limit the errors which results in more accurate geospatial data.

Hardware implementations and software solutions integrating affordable components are

presented, hence this thesis is contributing towards finalizing the physical system. This involves the integrating a GPS receiver (Garmin 18x-5Hz), CAN controller (PiCAN2), motor controller (Torqeedo interface board) and Torqeedo thrusters with a Raspberry Pi 3 single-board computer. Thrust commands and thruster response is successfully tested in ”forsøkshallen” experiment lab, elektrobygg D at NTNU Gløshaugen, Trondheim, Norway, with all system components behaving satisfactory.

Important components of a fish tracking control system are developed and optimized towards 5 objectives; keeping receivers within transmission range, enabling USVs to operate in restricted areas, minimizing geometric dilution of precision (GDOP), distance traveled by system nodes and need for inter-vehicle communication. A proposed formation controller unifies the system objectives in a single algorithm positioning USVs in circular and circle-arc shaped formations surrounding the target fish, whilst in non-restricted and restricted waters, respectively. Distance traveled by vessels is minimized through applying three concepts; target & guidance radius of acceptance and a path picking function. The need for inter-vehicle communication is significantly reduced by allowing client nodes (Otter USVs) to operate with independent guidance systems. Target & guidance radius of acceptance allow estimated target and receiver positions to deviate from their respective FPR and waypoint localizations within a limited area without assigning new waypoints or demanding station-keeping, respectively. Tuning of radii of acceptance is a trade-off between energy-use, GDOP and tracking ability of the system. The path picking function is a function which consequently picks paths that minimize the distance sum traveled by vessels, weighting the distance from their current positions.

The most significant error sources associated with positioning of a target fish for the proposed fish tracking system are presented in this thesis. This involves environmental-, acoustic receiver- and transmitter- related errors, with subgroups transmission path-, noise-, receiver positioning-, timestamping- and time-synchronization- related errors. Errors in timestamping and time-synchronization are scaled with the speed of the acoustic signal itself. Errors associated with the current GPS configuration aboard vessels are expected to be significant and other positioning methods should be assessed to assure better target tracking.

Bibliography

[1] Food and Agriculture Organization of the United Nations (FAO). The State of World Fisheries and Aquaculture 2016. Rome: 2016.

[2] SSB - Statistics Norway (Statistisk sentralbyrå). Aquaculture, 2016, preliminary figures [Internet]. Oslo: SSB; 2017 May 29 [cited 2018 June 01]. Available from: https://www.ssb.no/jord-skog-jakt-og-fiskeri/statistikker/fiskeoppdrett/aar-forelopige/2017-05-29content

[3] Isaksen JR, Andreassen O, Robertsen R. Kommunenes holdning til økt oppdrettsvirksomhet. Nofima; 2012 April. 18/2012. Availble from: https://www.nofima.no/filearchive/kommunenes-holdning-til-okt-oppdrettsvirksomhet.pdf

[4] Liu y, Diserud OH, Hindar K, Skonhoft A. An ecologicaleconomic model on the effects of interactions between escaped farmed and wild salmon (Salmo salar). Fish and Fisheries; 2013;14(2):158-173.

[5] Nekouei O, Vanderstichel R, Thakur K, Arriagada G, Patanasatienkul T, Whittaker P, et al. Association between sea lice (Lepeophtheirus salmonis) infestation on Atlantic salmon farms and wild Pacific salmon in Muchalat Inlet, Canada. Sci Rep. 2018;8(1):4023-4023.

[6] Ray Hilborn. Salmon-farming impacts on wild salmon. PNAS. 2016;103(42):15277.

[7] Communications and Data Systems Division, National Aeronautics and Space Administration (NASA). Telemetry Summary Of Concept And Rationale. Frascati, Italy: CCSDS Secretariat Communications and Data Systems Division; November 1986 [cited 2018 May 12]. 6.

[8] Hussey NE, Kessel ST, Aarestrup K, Cooke SJ, Cowley PD, Fisk AT, et. al. ECOLOGY. Aquatic animal telemetry: A panoramic window into the underwater world. Science. 2015;348(6240):125564.

[9] Zolich AP, Johansen TA, Alfredsen JA, Kuttenkeuler J, Erstrop E. A Formation of Unmanned Vehicles for Tracking of an Acoustic Fish-Tag. IEEE OCEANS 2017. Anchorage, Alaska: Institute of Electrical and Electronics Engineers (IEEE); 2017. pp. 1-6.

[10] Urke HA, Kristensen T, Ulvund JB, Alfredsen JA. Riverine and fjord migration of wild and hatchery-reared Atlantic salmon smolts. Fisheries Management and Ecology. 2013;20(6):544-552.

[11] Forney C, Manii E, Farris M, Moline MA, Lowe CG, Clark CM. Tracking of a tagged leopard shark with an AUV: Sensor calibration and state estimation. 2012 IEEE International Conference on Robotics and Automation. Saint Paul, MN: 2012. p. 5315-5321.

[12] Sousa LL, Lpez-Castejn F, Gilabert J, Relvas P, Couto A, Queiroz N, et al. Integrated Monitoring of Mola mola Behaviour in Space and Time. Porto, Portugal: Public Library of Science; 2016.

[13] Ekanger A. Examining Use of Various Unmanned Vehicles in Single Fish Tracking [unpublished]. (Specialization Project in Marine Control Systems). Trondheim: 2017 December.

[14] Efteland JØ. Underwater Acoustic Positioning System for Real-time Fish Tracking [Master's Thesis]. Trondheim: NTNU; 2016.

[15] Løvskar SS. Positioning of periodic acoustic emitters using an omnidirectional hydrophone on an AUV platform [Master's Thesis]. Trondheim: NTNU; 2017.

[16] Norgren P, Ludvigsen M, Ingebretsen T, Hovstein VE. Tracking and remote monitoring of an autonomous underwater vehicle using an unmanned surface vehicle in the Trondheim fjord. Washington, DC, USA: IEEE OCEANS 15; 2015.

[17] Solem Ø, Hedger R, Urke H, Kristensen T, F, Ulvan E, et. al. Movements and dispersal of farmed Atlantic salmon following a simulated-escape event. Environmental Biology of Fishes. 2013; 96(8):927939.

[18] Fossen TI. Handbook of Marine Craft Hydrodynamics and Motion Control. 1st Edition. Chichester, UK: John Wiley Sons, Ltd; 2013.

[19] Maritime Robotics AS. Otter USV [Internet]. Trondheim: Maritime Robotics AS; unknown date [cited 2018 May 01]. Available from: https://maritimerobotics.com/mariner-usv/otter/

[20] Vik B. Integrated Satellite and Inertial Navigation Systems. Trondheim: NTNU. 2014.

[21] Thelma Biotel AS. TBR 700 ACOUSTIC RECEIVER [Internet]. Trondheim: Thelma Biotel AS; unknown date [cited 2018 February 06]. Available from: http://www.thelmabiotel.com/tbr-700/

[22] Thelma Biotel AS. TBR 700 RT [Internet]. Trondheim: Thelma Biotel AS; unknown date [cited 2018 February 07]. Available from: http://www.thelmabiotel.com/tbr-700-real-time/

[23] Thelma Biotel AS. Pressure/Depth [Internet]. Trondheim: Thelma Biotel AS; unknown date [cited 2018 February 07]. Available from: http://www.thelmabiotel.com/depth-tags/

[24] Garmin International, Inc. GPS 18x Technical Specifications. Olathe, Kansas, USA: Garmin International, Inc; 2011.

[25] MikroElektronika. MAX3232 Board [Internet]. [unknown location]: MikroElektronika; unknown date [cited 2018 February 02]. Available from: https://www.mikroe.com/max3232-board

[26] SK Pang electronics. PiCAN2 CAN-Bus Board for Raspberry Pi 2/3 [Internet]. [unknown location]: SK Pang electronics; unknown date [cited 2018 February 12]. Available from: http://skpang.co.uk/catalog/pican2-canbus-board-for-raspberry-pi-23-p-1475.html

[27] Raspberry Pi Foundation. RASPBERRY PI 3 MODEL B [Internet]. [unknown location]: Raspberry Pi Foundation; unknown date [cited 2018 February 24]. Available from: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[28] Maritime Robotics AS. Torqeedo interface board [not published].

[29] Huawei. E3372 [Internet]. Shenzhen, Guangdong: Huawei; unknown date [cited 2018 January 17]. Available from: https://consumer.huawei.com/en/mobile-broadband/e3372/specs/

[30] Pinto J, Dias PS, Martins R, Fortuna J, Marques E, Sousa J. The LSTS toolchain for networked vehicle systems. OCEANS - Bergen, 2013 MTS/IEEE. Bergen, Norway: IEEE; 2013.

[31] LSTS Toolchain. Our Toolchain [Internet]. Porto, Portugal: LSTS toolchain; unknown date [cited 2018 February 17]. Available from: https://lsts.fe.up.pt/toolchain

[32] LSTS Toolchain. Dune Tasks interaction [Internet]. Porto, Portugal: LSTS toolchain; unknown date [cited 2018 February 17]. Available from: https://www.researchgate.net/figure/Message-passing-concept-behind-DUNE-tasks-implementation_fig1_259678296?_sg=7INMMHQYRj9wvN_fGmB7OJxTeZ6Zn8LlPO14565CjWStwJLuxD9K7b9IYbJpgaEeNF8crUMeoplHWne4Ri6CjA

[33] Kalman RE. A new approach to linear filtering prediction problems. ASME Journal of Basic Engineering. 1960;82(1):35.

[34] Becerra VM, Roberts PD, Griffiths GW. Applying the extended Kalman filter to systems described by nonlinear differential-algebraic equations. Control Engineering Practice. 2001;9(3),267-281.

[35] Skjetne R. The Maneuvering Problem [PhD]. Trondheim: NTNU; 2005.

[36] Faltinsen OM. Sea loads on ships and offshore structures. Cambridge: Cambridge University Press; 1990.

[37] Big Mess 'o Wires. Raspberry Pi GPIO Programming in C. Unknown: Big Mess 'o Wires; unknown date [cited 2018 June 04]. Available from: https://www.bigmessowires.com/2018/05/26/raspberry-pi-gpio-programming-in-c/

[38] Google. Google Maps [Internet]. Mountain View, California, USA: unknown date [cited 2018 May 06]. Available from: https://www.google.no/maps/

[39] Kartverket. Sjøkart [Internet]. Hønefoss, Norway: unknown date [cited 2018 May 04]. Available from: https://www.kartverket.no/kart/sjokart/

[40] Burrowes G, JY. Khan. Chapter 8 Short-Range Underwater Acoustic Communication Networks. Autonomous Underwater Vehicles. Croatia: InTech Europe; 2011. p. 173-198.

[41] Kinsler, LE. Fundamentals of acoustics. 3rd ed. New York: Wiley; 1982.

Appendix

A Communication protocols

Some of the serial communication protocols used by the proposed system are presented followingly.

NMEA 0183 protocol
The NMEA 0183 is a electrical and data specification which is widely used in marine systems for communication with different electronic components, e.g sonars, compasses and GPS receivers. NMEA 0183 uses a simple ASCII encoding for serial communication which defines how the data is transmitted. The data is transmitted in ”data-sentences”, with a special start delimiter character, often set to the dollar sign: $. The messages have a maximum length of 82 ASCII characters, ranging from (space) to (~), which corresponds to 0x7e to 0x20 in hexadecimals. All data fields are separated by commas (comma-delimited). The typical baud-rate for NMEA devices is 4800 baud with 8 bits (byte) of data, but varies from different type of sensors and electrical devices. The method of parity is not used in the NMEA 0183 protocol, and messages comes with one stop bit.

Serial Peripheral Interface
Serial Peripheral Interface is a synchronous serial communication. Connections can be made with a D-subminiature (DB9) or a 3 way screw terminal.

American Standard Code for Information Interchange
American Standard Code for Information Interchange (ASCII) is a character encoding standard for serial communication. ASCII code is often embedded in a byte, a eight bit

field with seven information bits and a parity bit. Using the whole byte for information is known as extended ASCII.

Universal Asynchronous Receiver-Transmitter communication
Universal Asynchronous Receiver-Transmitter (UART) communication is a computer hardware device for asynchronous serial communication. The protocol is byte-oriented.

Universal Serial Bus
Universal Serial Bus (USB) is an industry standard for both asynchronous serial communication and power supply.

Recommended Standard 232
Recommended Standard 232 (RS-232) is a serial communication standard which was introduced in 1960. In RS-323 is the data sent as time-seris of bits, and both synchronous and asynchronous data transmission is supported by the standard. The standard have voltage levels which corresponds to logical one and zero in the range from +- 3 to 15 volts, with respect to a ”common ground” (GND) pin. A voltage within the range in between -3 to 3 volts is therefore not a valid RS-232 voltage. Two data transmission lines are included, transmit (TxD) and receive (RxD). Some RS-232 devices come with a DE-9 socket outlet.

Recommended Standard 485
RS-485 is an electrical interface which is mainly used in asynchronous serial communication. RS-485 support multidrop communication (multidrop bus). The TBR700 communication is based on half-duplex RS-485, a differential bus which is more robust than single-ended serial protocols.

Torqeedo communication protocol

Description	PCB label	Max current	Voltage	Rail	Channel
Motor 0	H_MOT0	60 A	24-33,2 V	0	0
Motor 1	H_MOT1	60 A		1	0
Aux 0	H_AUX0	10 A	24-33,2 V	2	0
Aux 1	H_AUX1	10 A			1
12V out 0	H_12V0	3 A (total)	12 V	3	0
12V out 1	H_12V1				1
12V out 2	H_12V2				2
Vreg. out 0	H_VR0	3 A (total)	Adjustable 5-15 V		3
Vreg. out 1	H_VR1				4
5 V output	H_5V	3 A	5 V		5

Name	Rail	Fuse
Motor 0	0	60 A. Part of LT1910 driver, current set by shunt resistor. Reaction time is a few hundred nanoseconds.
Motor 1	1	Same as above
Aux	2	10 A. Uses INA226 current alert to trigger interrupt. Reaction time is a few hundred microseconds.
Other	3	None, but regulators are current limited to 3 A.

Name	ID	Direction	Description
MSG_TQ_MOTOR_SET	13	To board	Set throttle value for both motors. Motor power must be enabled with MSG_OUTPUT_SET first!
MSG_OUTPUT_SET	9	To board	Change output state (enable/disable). If fuse is triggered, use this message to reset the fuse.
MSG_RAIL	3	From board	Current, voltage and more for each rail (0-3).
MSG_TQ_MOTOR_DRIVE	12	From board	Power, temperature and RPM, per motor.
MSG_TQ_MOTOR_STATUS_BITS	16	From board	Status and error flags, per motor.
MSG_TQ_BAT_STATUS	14	From board	Voltage, current, SoC, temperature and error code, per battery.
MSG_TQ_BATCTL	15	From board	Most recent error code from battery, kept sticky by the battery controller in case of power loss (small black box plugged into the battery).
MSG_ID	6	From board	Identifies the board type (MR Torqeedo Interface Board).
MSG_UPTIME	10	From board	Uptime since boot, and cause of last reboot.

Figure 1: Torqeedo interface board communication protocol ([28])

B Hardware & Software Guidelines

This is a brief guide on how to replicate some of the work presented in this thesis.

B.1 Raspberry Pi 3

Some general instructions follows.

Raspberry Pi 3 power supply
During operation, i.e while the Raspberry Pi 3 is operating aboard the Otter USV, the micro-controller must be powered by the Torqeedo interface board. A connector from the Torqeedo board to RP3 is therefore needed. The Raspberry Pi 3 can be powered through a micro USB connector, the GPIO pins and a regular USB. A connector from the PCB (Torqeedo board) 5V outlet "H_5V" is needed. The Molex Nano-Fit 2-ways receptacle connector can be used for this purpose. A regular phone charger is sufficient for testing purposes.

Micro SD Memory Card
The Pi is not delivered with a micro SD card, so the user has to buy a separate card of minimum class 10. Class 10 card are able to transmit data with 10 MB/s. Make sure the card has enough memory to hold both the Raspbian operating system (version Jessie full $\approx$ 5 MB), and dune.

Reformat SD card and get NOOBS image
Use a SD card reader (either USB reader or computer with this functionality) to both reformat and image NOOBS on to the SD card. NOOBS is an easy operating system installer which contains Raspbian. When NOOBS is imaged on the SD card, the user can insert the SD card on the bottom side of the Raspberry Pi 3.

First boot
Connect the Raspberry Pi 3 to a monitor or a tv through a HDMI cable, and connect a mouse and a keyboard through the USB ports. The Pi boots automatically when a regular micro usb is providing the Pi with 5V. Install Debian operating system on the Pi as guided by NOOBS.

Setting correct keyboard language
After booting the Raspberry Pi 3 for the first time, the user might want to change the keyboard language into his preferable language. This can be done through the terminal by the command:
$ sudo raspi-config
Then pick option 4 Localization Options, and then option I3 Change Keyboard Layout.

Freeze
The Raspberry Pi 3 occasionally freezes. Simply disconnecting the 5V power supply and reconnecting the cable makes the Pi reboot.

SSH connection
There exist a variety of different ways to connect to the Raspberry Pi 3 externally from another computer. One option is to use PuTTY software on a Windows computer. In order to do this, the user must know the host name or IP adress of the Pi, which can be done by writing the command ”ip config” in the Raspberry Pi terminal. In this way the user have access to the terminal in the Raspberry Pi externally, and is able to use the computer without projecting a window through the HDMI cable.

Power sign / low power
If the power sign is shown, it means that the Pi is almost using all the power of which it is supplied with. A way to provide the Pi with additional power is to connect it with a USB cable in addition to the micro USB. In general, make sure to only run one power demanding operation on the PI at once. For example, do not browse in a web browser while running the make task or installing some software, as this may cause the Pi to overuse its electricity or freeze.

Connect to Eudoram (NTNU internet)
$ pi@raspberrypi: /Desktop $ cd
$ pi@raspberrypi: $ cd Downloads
$ pi@raspberrypi: /Downloads $ ./eduroam-linux-Ntu-N.sh
By using the commands above and typing in your username and password can the user connect to the Eudoroam internet using the RP3.

B.2 DUNE

This is a simple guide to how Unified Navigation Environment (DUNE) can be installed on a Raspberry Pi 3 single-board computer.

Access to dune source code in Github and Gitlab
A variety of different DUNE versions exist. However, in many cases the user needs access to Github or Gitlab repository governing the source code. To clone a Github repository is very simple through running the clone command directly in terminal. To clone a Gitlab repository is somewhat harder, as the user might have to create an SSH key on their Raspberry Pi. This is done in the terminal by running the code line $ ssh-keygen -t rsa. When the key pair is created, the user can simply copy the public part of the SSH key by using the terminal code line $ cat /.ssh/id_rsa.pub and manually copying the terminal output. This output aka public part of the user’s SSH key, should then be added to the ”SSH keys” in the user’s Gitlab in a web browser. The ”SSH keys” menu is somewhat hidden and could be hard to find the first time. By simply navigating to the top right part of the Gitlab window, and click on the scroll down menu next to the user’s Gitlab picture and click ”settings”, the ”SSH keys” menu should be visible on the left side of the screen.

Installation
The following command lines are written in RP3 terminal to install the official or NTNU version of DUNE. $ mkdir /DUNE
$ cd /DUNE
$ git clone https://github.com/LSTS/dune.git
For NTNU version (demands SSH key access):
$ git clone git@uavlab.itk.ntnu.no:uavlab/dune.git
$ mkdir build
/DUNE$ cd build
/DUNE/build$ sudo apt-get install cmake
/DUNE/build$ sudo cmake -G "Eclipse CDT4 - Unix Makefiles"../dune
/DUNE/build$ cmake ../dune
/DUNE/build$ cmake ../dune
/DUNE/build$ make -j3

Creating a DUNE task
/DUNE$ python programs/scripts/dune-create-task.py DUNE_PATH AuthorName NameOfTheTask

Rebuild cache
In DUNE, the rebuild cache line must be run every time changes is made to a DUNE task or a new INI file is created. If changes are made to an existing INI file, neither rebuild cache nor make is needed. The following code line indicates how to rebuild cache and then make:
/DUNE/build$ make rebuild_cache
/DUNE/build$ make -j3
A quicker way to do this is the command:
/DUNE/build$ make rebuild_cache make -j3
Run dune task
/DUNE/build$ ./dune -c ../etc/development/CONFIGURATIONFILE.ini

make
"Make" is the function which creates the DUNE tasks, and must be run every time changes in any DUNE tasks are made, or new instances of INI files are created. It is not necessary to run "make" if changes to already existing INI files are made. The make command is only using one core, and could be enhanced by the commands: make -j2, make -j3 and make -j4, where 2,3 and 4 core are being used, respectively. Make -j4 is the fastest one, but I would however recommend to use the make -j3 command (for RP3 Model B) as the make -j4 quite frequently causes the Raspberry Pi to freeze.

Multimeter power tests

Through using a regular multimeter can the user test the voltage level provided to the RP3 to ensure that the micro USB power is sufficient. A similar test can be performed to test the power outlet in the GPIO pins.

Overall power test

Set a multimeter to 20 Vdc (direct current) indicated by a V with straight line and a dotted line on its right side, and let the red cable be connected to the Volt-omega-mA inlet, and the black cable be connected to the COM outlet. Simply flip the raspberry pi. Touch and connect the black cable pin to the ground indicated with PP5, and the red cable to either the PP1 or PP2 connector. Read the power off the multimeter. If you somehow switched up the cables or pinned the with opposite color pins, you will simply get a reading of the negative voltage, in this case -5 volts. In this figure, PP1 and PP2 are 5V input from USB. PP35 is 5V after the polyfuse. PP7 is 5V after the input circuit PP3 through PP6 are GND.

GPIO power test:

Set the multimeter to 20 Vdc. Orient the raspberry pi with the front side up, and touch and connect the black cable pin to the ground GPIO ground pin 06, and touch and connect the red cable pin to the GPIO 5V DC power pin 02 or 04. If the user somehow switches up the cables or measures the with opposite color pins, no major danger occurs but the voltage reading will simply be the negative voltage, in this case -5 volts. This test was mainly performed to ensure that the input voltage of the GPS 18x 5Hz is 4.4 5.5 Vdc, which is provided by the GPIO pin 2, was within the operation range given in the [24].

Testing Garmin 5Hz GPS voltages

Set the multimeter to 20 Vdc. Orient the raspberry pi with the front side up, and touch and connect the black cable pin to any of the GPIO ground pins, and touch and connect the red cable pin to green receive line while it is not connected to the GPIO port. If the user want to test the transmitted voltage, connect the red cable pin to the white transmit line instead. The GPS Wire Pinout is given in table 4. This test was successfully done by this author measuring 5 volts in the green receive wire, and 3.3 volt in the white transmit wire. These measures match the CMOS serial output levels specifications stated in the product specifications [24].

B.3 Garmin 5Hz-18x GPS

This is a guide on software and hardware alteratioins which are necessary in order to connect a Garmin 5Hz GPS to a Raspberry Pi 3 and gather GPS data through UART communication.

Connection

The Garmin 5Hz-18x GPS can be connected to the GPIO pins of the Raspberry Pi 3 using a Mikroelektronika MAX3232 power converter, as indicated in 2. A solder iron and some additional wires are necessary to perform the hardware alterations. The three ground wires from the Garmin GPS must be soldered together as one. Detailed information on each wire

is indicated in the figure 2, where the colored wires going from the gps are equivalent to the ones given by the manufacturer, as given in figure 4.

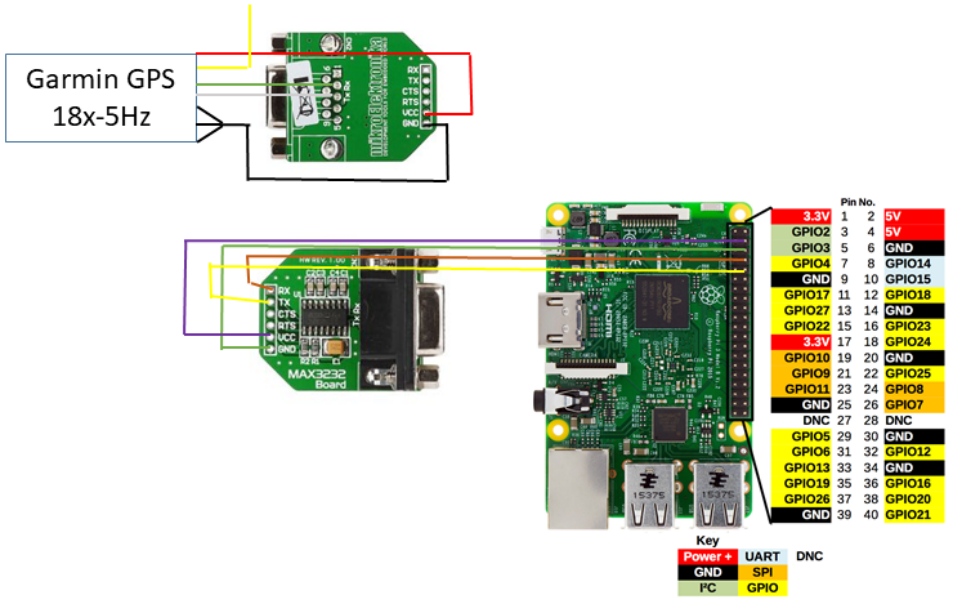

Figure 2: GPS connection schematics [25] [37]

Software alterations

Enabling UART
GPIO serial port is disabled by default in the Raspberry Pi 3 running a normal Jessie Raspbian operating system. This is solved by editing the file config.txt. Run the command:
$ sudo nano /boot/config.txt
in the RP3 Terminal window, and add the line ”enable_uart=1” at the bottom of the file. Save the file (press Ctrl-O, then enter) and exit (press Ctrl-X).

UART workaround
In earlier versions of Raspberry Pi and Linux in general, COM1 equivalent is found on pins 14 and 15 of the GPIO header and is called /dev/ttyAMA0. However, for Raspberry Pi 3 the new Bluetooth functionality have taken the /dev/ttyAMA0 from the GPIO header and an inferior second one has been substituted in its place. /dev/ttyAMA0 was a hardware serial port (uart) and high performance. The second serial port is referred to as mini uart in /dev/ttyS0. /dev/ttyS0 calculates the bit timing from the CPU cores frequency, but if the CPU is under heavy load, sometimes the serial communications get corrupted. A way to work around this is provided further, but unfortunately comes with a slight loss in performance, although normally not noticeable by the user. This work around switches the Bluetooth and GPIO serial port in the following manner: before switch/workaround:

/dev/ttyAMA0 → Bluetooth
/dev/ttyS0 → GPIO serial port

after switch/workaround:

/dev/ttyAMA0 → GPIO serial port
/dev/ttyS0 → Bluetooth

To see what configuration which is running on the Raspberry, simply run the command ”ls -l /dev”in Terminal, and see where the serial ports are pointing.

Disabling the Console
Because the serial port is used for anything other than the console, it must be disabled. The following changes must be performed:

$ sudo systemctl stop serial-getty@ttyS0.service
$ sudo systemctl disable serial-getty@ttyS0.service

and make changes to the cmdline.txt file. To edit the file write:
$ sudo nano /boot/cmdline.txt In the cmdline.txt the user will normally find something like this:
dwc_otg.lpm_enable=0 console=serial0,115200 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline fsck.repair=yes root wait

In order to not encounter a lag problem, the user must remove the line:
console=serial0,115200
then save and reboot for changes to take effect.

GPS data acquisition software

GPS 18x 5Hz has a default setting of Baudrate 19200 [24], databits 8, parity none, and 1 stopbit. A number of different software can be used for data acquisition and analysis. Minicom or microcom are two possible options for which can be used to analyze and read data. It is also possible to read data directly to the Terminal through by using any of the following commands:
$ screen /dev/ttyAMA0
$ Sudo screen /dev/serial0
$ screen /dev/ttyS0 ¡speed¿
$ screen /dev/ttyS0 ¡19200¿

However, for real-time data acquisition in DUNE, a very simple INI file code segment must be added:

```
[Sensors.GPS]
Enabled = Hardware
Debug Level      = Spew
Entity Label = GPS
Serial Port - Device = /dev/ttyAMA0
Serial Port - Baud Rate = 19200
Sentence Order = GPGGA, GPRMC
```

Please note that the baud rate must be correct in order to receive useful data.

Potential error messages and solutions

In the process of alternating the Raspbian cmdline.txt and config.txt, small errors might affect the Raspberry's ability to boot as normal. In the following section is some of the associated errors this author experienced.

ERROR MSG: ttyAMA0 is busy
This error can be solved by killing the task. $ screen -d -m /dev/ttyS0
$ screen -ls
There is a screen on: 5207..host (10/04/2011 10:16:50 AM) (Detached) 1 Socket in /var/run/screen/S-user.
$ screen -r 5207 -X kill
$ screen -ls
No Sockets found in /var/run/screen/S-user.
The name of the process is often name in front of "host..". In this author's case, the following code segment solved the problem:
$ screen r 884 X kill

Kernel panic - not syncing: VFS: Unable to mount root fs on unknown-block(179,2)
This problem was solved by removing the SD card from the raspberry pi 3 and opening the boot folder in windows, where the user have access to the altered files cmdline.txt config.txt. However, the problem was that the original cmdline.txt was changed into a format named cmdline.txt.orig, and standard windows have no capability of opening a file of this kind. A simple online converter was sufficient to change the message back into .txt, and by reverse the config.txt changes in windows, the problem was solved. After altering the changes back into its original state, the SD card was dismounted and put in to the RP3. The system then rebooted as normal.

Python: could not open port.
$ sudo chmod 666 /dev/ttyAMA0

B.4 CAN Interface

This section aims to provide some guidelines in how to setup the CAN interface on a Raspberry Pi 3 with a PiCAN2 (CAN controller), and integrate it with the Torqeedo interface board.

Extended GPIO pins and spacing
In the ”out of box” configuration of which the Raspberry Pi 3 and PiCAN2 micro-controller boards are delivered in, is no additional space on the GPIO connection left for other purposes as the CAN shield fully covers the pins. Extended GPIO pins are therefore needed to enable the Raspberry Pi to be connected to the Garmin 18x-5Hz GPS, PiCAN2 and a Molex Nano-Fit power supply connector from the PCB, simultaneously during tracking operations. It therefore suggested that an additional stacking header, e.g 40-pin GPIO extra long stacking header, is purchased and connected to the computer. To ensure a steady distance in-between the micro-controller boards can 11 mm standoff spacers be purchased and connected to the Raspberry Pi.

Bringing up the CAN interface
The definition of ”bringing up the interface” is the availability of a can0 port device on the Raspberry Pi 3 computer (”ifconfig can0” command in Linux). Please note that the PiCAN2 board must be connected when using this command in order for the interface to show up. When the interface is successfully setup, should this command give a input similar to the following:
pi@raspberrypi:$ ifconfig
$ can0: flags=193⟨UP,RUNNING,NOARP⟩ mtu 16
$ unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 10 (UNSPEC)
$ RX packets 866 bytes 6682 (6.5 KiB)
$ RX errors 0 dropped 866 overruns 0 frame 0
$ TX packets 0 bytes 0 (0.0 B)
$ TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

Installing CAN utilities
The first step is to install the CAN utilities and add some loadable Kernel modules to Linux by using the modprobe command. The following command prompt lines are necessary to enable the CAN interface:
$ sudo apt-get update
$ sudo apt-get install can-utils
$ sudo modprobe can
$ sudo modprobe vcan
$ sudo modprobe slcan

Changes to /boot/config.txt
$ sudo nano /boot/config.txt
Then add the following lines to the bottom of the file:

dtoverlay=mcp2515-can0,oscillator=16000000,interrupt=25
dtoverlay=spi-bcm2835
Save the file (press Ctrl-O, then enter) and exit (press Ctrl-X).

Changes to /etc/network/interfaces.txt $ sudo nano /etc/network/interfaces.txt
Then add the following lines to the bottom of the file:
auto can0
Iface can0 inet manual
Save the file (press Ctrl-O, then enter) and exit (press Ctrl-X).

Python thrust commands
Python-can can be used to create thrust-commands for the Torqeedo thrusters. The following commands installs pythons-can on RP3.
$ pi@raspberrypi: /Desktop $ sudo apt-get install crcmod
$ sudo easy_install -U pip
$ pi@raspberrypi: /Desktop $ sudo pip install -U crcmod
$ pi@raspberrypi: /Desktop $ sudo pip install python-can

Raspberry Pi 3 to PiCAN2 connection
The PiCAN2 board can be directly connected on-top of the Raspberry Pi 3 board GPIO pins.

PiCAN2 to Torqeedo board connection
The PiCAN2 board is connected to the Torqeedo board through a two-wire CAN-cable, for high and low CAN current. The connections are directly connected to the 4-way screw terminal on the PiCAN2 board from the Torqeedo board.

120Ω Terminator
A terminator solder, i.e a 2way header pin to JP3 on the PiCAN2 board, and a jumper is inserted on the board to terminate both ends of the PiCAN2 board. This must be done because the CAN bus is a voltage-differential dependent serial communication method.

Torqeedo board communication protocol

CAN bus messages consist of pair-wise hexadecimals, each able to express 256 (16^2) various symbols. Two important destinations of the CAN bus is the address of where the Torqeedo board broadcasts messages, and the address of the board itself. Additional information regarding the Torqeedo board and its communication protocol is found in Appendix table 1.

Messages from board
The board broadcasts seven various ID messages to the MRCAN_BC with the hexadecimal

address (0xFF), and are all explained in the following list.

- ID 3: 0x03 - Current, voltage and more for each rail
- ID 6: 0x06 - Identifies the board type (MR Torqeedo Interface Board).
- ID 10: 0x0A - Uptime since boot, and cause of last reboot.
- ID 12: 0x0C - Power, temperature and RPM, per motor.
- ID 14: 0x0E - Voltage, current, SoC, temperature and error code, per battery.
- ID 15: 0x0F - Most recent error code from battery, kept sticky by the battery controller in case of power loss.
- ID 16: 0x0F - Status and error flags, per motor.

Messages to board
Two types of messages are sent to the board at its board address 0xAB:

- ID 9: 0x09 - Change output state (enable/disable). If fuse is triggered, message is used to reset the fuse. Motor power must be enabled with this message first.
- ID 13: 0x0D - Set throttle value for both motors.

Endianness
CAN bus messages sent from and to the Torqeedo board are sequentially ordered in a little-endian format, in which the most siginificant byte, i.e the byte containing the lowest address value, is sent first, in a decreasing significance order. An example being that the byte FF (255) is sent after A1 (161).

Set thrust
The ID 13: 0x0D - message is of further study in the following section, as this is a very important message being used by the control system in DUNE to set thrust. Thrust is given as a value in the interval -1000 and 1000 for the Torqeedo thrusters aboard the Otters, indicating maximum thrust in forward (surge) and backwards (-surge) direction, respectively. A single CAN ID 13 message sets desired thrust for both propellers in the following manner: 00 D0 FE AB C8 00 C8 00
where 00 D0 corresponds to the ID of the message, and AB referrers to the address of the board. The first C8 00 corresponds to 200 thrust for motor 1, and the second C8 00 to 200 for motor 2, in little endian format. Full thrust is given by 03 E8 (1000), leaving a large variety of combinations unused as four hexadecimals have 65536 possible combinations (16^4).

Useful CAN functions
Some functions which can be used to both print and create CAN messages are candump and cansend, respectively. candump can be specified to only show messages with specific ID, i.e 0x123 or 0x7FF on vcan0, through the following command:
$ candump vcan0,0x123:0x7FF
cansend sends a single CAN frame onto the bus, with a specified device, identifier and

data bytes. An example is:
$ cansend can0 123223366
Where 0x123 is the identifier, and 22, 33, and 66 represents the data bytes 0x22, 0x33, and 0x66. Values are assumed to be hexidecimal.

The ”can” command only refers to interfaces that already exist, such as a physical can0, an example being a CAN-to-USB dongle. Virtual CAN interfaces are referred to through the command ”vcan”, and can be used to simulate a CAN bus without real hardware. A virtual CAN interface can be useful for simulation and testing, and Can-utils can be tested without having an actual CAN device. A vcan device is created by running the following lines:
$ sudo modprobe vcan
$ sudo ip link add dev vcan0 type vcan
Once the device is created, it can be used like any other CAN device.

C Schematics

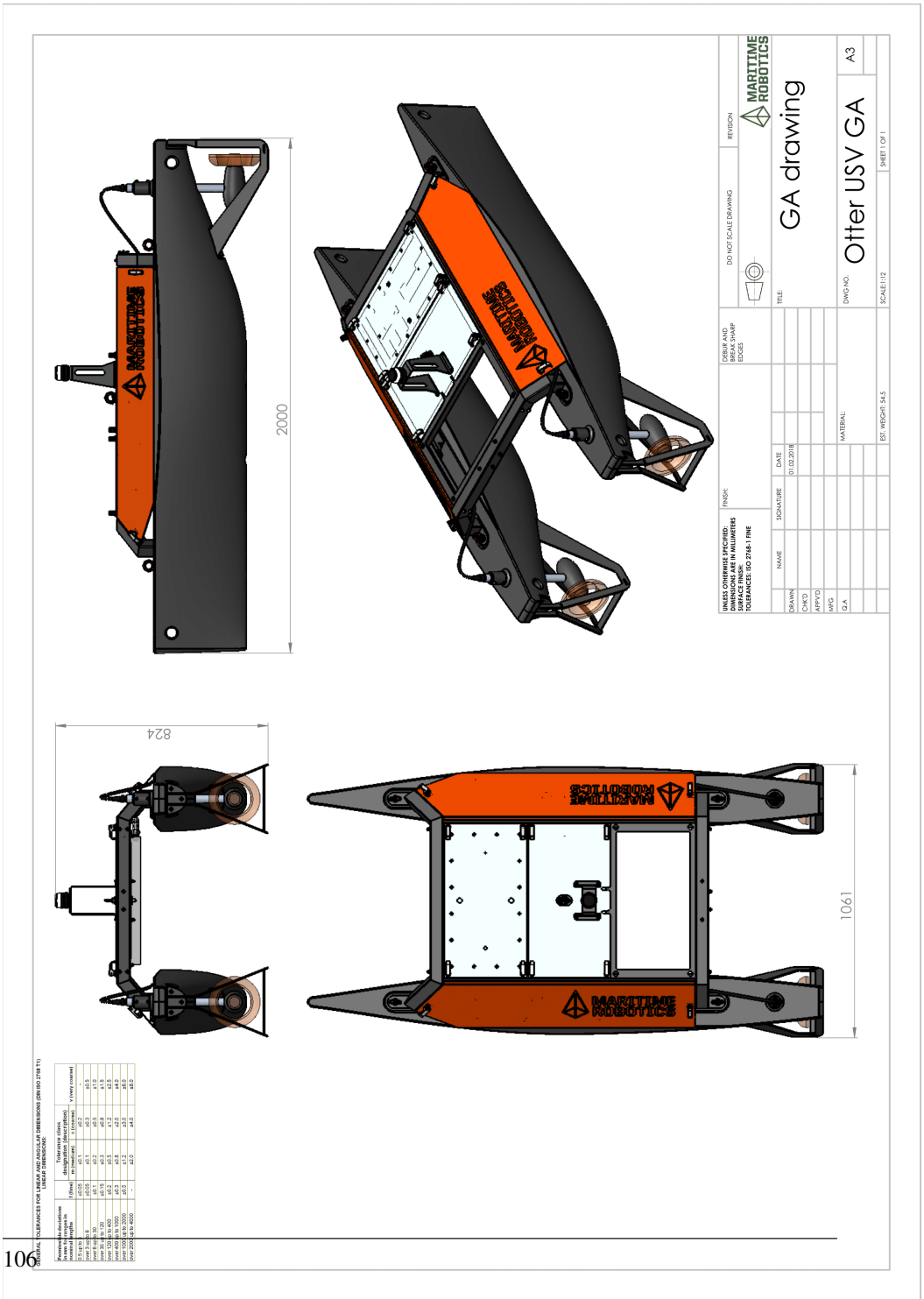

Figure 3: Otter USV schematics [19]

GPS 18x Pin #	Color	Signal Name	Wire Gauge
1	Yellow	Measurement Pulse Output	28
2	Red	Vin	26
3	Black	Ground	28
4	White	Transmit Data	28
5	Black	Ground	26
6	Green	Receive Data	28

Figure 4: GPS 18x-5Hz Wire Pinout [24]

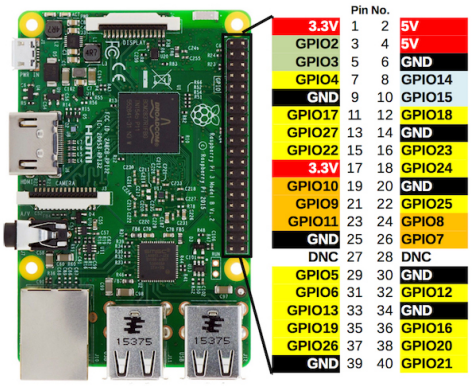

Figure 5: Raspberry Pi 3 GPIO chart [37]

D DUNE software

This section include information about DUNE and software as presented in simulation chapter 5. Firstly is an example of how DUNE tasks and ini-files interact presented. Then follows an explination about how the CVS files are built, and code of the developed INI files and DUNE tasks. Finally are aid functions used by DUNE tasks presented in a header and source code file; funkytions.h and funkytions.cpp.

D.1 INI file & DUNE task example

An example of how INI files and DUNE tasks work together, and how task parameters can be manipulated in the INI files, follows. By including the following code snippets in the two hypothetical DUNE tasks; Sensors/TBR700/task.cpp & Control/HeadingAndSpeed/-Task.cpp, can the value of the ID of an acoustic receiver or a PID integral gain be tuned and set directly from a INI file:

Changes in Sensors/TBR700/task.cpp (onUpdateParameters(void)):
param(”Acoustic Receiver ID”, tbrID)
.description(”Name/ID of TBR700 Acoustic Receiver”)

.defaultValue("TBR02")
param("Integral Gain", K_i)

Changes in Control/HeadingAndSpeed/Task.cpp (onUpdateParameters(void)):
param("Integral gain", K_i)
.description("PID integral gain")
.defaultValue("0")
ottThe receiver ID and integral gain is set to "TBR02" and 0 by default, respectively.

Initializing Sensor/TBR700 Task and receiver ID in INI file:
[Sensors.TBR700]
Acoustic Receiver ID = TBR01
Enabled = Always
Entity Label = Acoustic receiver
Debug Level = Debug

Initializing Control/HeadingAndSpeed Task and integral gain in INI file:
[Control.HeadingAndSpeed]
Enabled = Always
Integral Gain = 1 Entity Label = Heading and Speed Controller
Debug Level = None

A DUNE task is initialized by initializing the tasks in brackets, and setting the Enabled value to Always (instead of Never). The Debug Level is a custom parameter which adjusts the amount of information which is printed to the terminal and can be set equal to the varibles None, Debug or Spew. tbrID and K_i refers to the variable names within the DUNE task, while the identifiers "Acoustic Receiver ID" and "Integral gain" are used in the INI file. If no values are set in the INI files are the default values used.

D.2 CVS data (.txt)

usv_pos_ecef.txt & usv_pos_llh.txt
The data format of CVS files used in the creation of the simulation presented in chapter 5 follows. Position data in ECEF and LLH coordinates for four USVs are saved in the Comma-separated values (CVS) text-files, target_ecef.txt and target_llh.txt. Both the data sets consist of 3 times 4 times 12000 data points, corresponding to x,y,z dimension, receiver 1,2,3,4, and total number of iterations, respectively.

In the file usv_pos_ecef.txt, are double values/elements within the CVS file given as x_j^i, y_j^i and z_j^i represents x,y,z coordinates in the ECEF frame for an USV/receiver with number $j \in [1, 2, 3, 4]$, and iteration i $j \in [0, 12000]$. The data is stored in the following manner:

$x_1^0, y_1^0, z_1^0, x_1^1, y_1^1, z_1^1, \quad ... \quad , x_1^{12000}, y_1^{12000}, z_1^{12000}$

$x_2^0, y_2^0, z_2^0, x_2^1, y_2^1, z_2^1, \quad ... \quad , x_2^{12000}, y_2^{12000}, z_2^{12000}$

$x_3^0, y_3^0, z_3^0, x_3^1, y_3^1, z_3^1, \quad ... \quad , x_3^{12000}, y_3^{12000}, z_3^{12000}$

$x_4^0, y_4^0, z_4^0, x_4^1, y_4^1, z_4^4, \quad ... \quad , x_4^{12000}, y_4^{12000}, z_4^{12000}$

In the file usv_pos_llh.txt are double values/elements within the CVS file given as lon_j^i, lat_j^i and hae_j^i represents longitude, latitude, and height above WGS84 ellipsoid (hae), the LLH frame, for an USV/receiver with number $j \in [1, 2, 3, 4]$, and iteration i $j \in [0, 12000]$. The data is stored in the following manner:

$lon_1^0, lat_1^0, hae_1^0, lon_1^1, lat_1^1, hae_1^1, \quad ... \quad , lon_1^{12000}, lat_1^{12000}, hae_1^{12000}$

$lon_2^0, lat_2^0, hae_2^0, lon_2^1, lat_2^1, hae_2^1, \quad ... \quad , lon_2^{12000}, lat_2^{12000}, hae_2^{12000}$

$lon_3^0, lat_3^0, hae_3^0, lon_3^1, lat_3^1, hae_3^1, \quad ... \quad , lon_3^{12000}, lat_3^{12000}, hae_3^{12000}$

$lon_4^0, lat_4^0, hae_4^0, lon_4^1, lat_4^1, hae_4^4, \quad ... \quad , lon_4^{12000}, lat_4^{12000}, hae_4^{12000}$

The target_ecef.txt & usv_llh.txt Position data in ecef and llh coordinates for four usvs are saved in the Comma-separated values (CVS) text-files , target_ecef.txt and target_llh.txt. Both the data sets consist of 3 times 4 times 12000 data points, corresponding to x,y,z dimension, receiver 1,2,3,4, and total number of iterations, respectively.

D.3 INI files

The simulation parameters and debug-settings are set in the .INI files using DUNE. Followingly are the INI files initializing the simulations presented in chapter 5. INI files are ran using the "./dune -c "configFileName" in the dune folder build: dune/build, i.e is the following file ran by the command: "./dune -c "fishpath".

fishpath.ini

```
[Include ../common/transports.ini]

[august.createTOAdataFromECEFdata]

Enabled = Always
Entity Label = Producer
Debug Level = Debug #Debug #None


[august.EKF2]
TBR700 receiver 1 ID = TBR01
```

```
TBR700 receiver 2 ID = TBR02
TBR700 receiver 3 ID = TBR03
TBR700 receiver 4 ID = TBR04
Enabled = Always
Entity Label = Consumer7
Debug Level = Spew #None


[Transports.Logging]
Enabled = Always
Entity Label = Logger
Transports = DevDataText
```

fishpath_createErrors.ini

```
[august.createErrorPlots]
Enabled = Always
Entity Label = Consumer7
Debug Level = Spew #None
```

D.4 DUNE Task: createTOAdataFromECEFdata

Code description:
Firstly, the task saves CVS data from .txt files in vector containers, with ”USVs” and ”fish” struct elements, in which is easier accessible. Secondly, is the position data of both the target and USVs used to calculate a TOA measurement for each receiver. The data is stored in a tupple vector-container, where each tupple contains the data indicated in the data message below. Finally, are the vector-containers sorted with respect to their TOA measurement, to enable real-time message dispatching to the bus as it would occur in a real system.

```
// Author: August

// Cpp headers
#include <cstdlib>
#include <cmath>
#include <cstring>
#include <string>
#include <vector>
//#include <iostream>
#include <fstream>
#include <cctype>
#include <algorithm>
#include <fstream>
#include <vector>
#include <iostream>
#include <algorithm>
#include <sstream>
```

```
18 #include <utility>
19 #include "funkytions.cpp"
20 #include <tuple>
21 // DUNE headers.
22 #include <DUNE/DUNE.hpp>
23
24
25 namespace
26 {
27 namespace august
28 {
29 namespace createTOAdataFromECEFdata
30 {
31 using DUNE_NAMESPACES;
32
33 double period;
34 double simspeed;
35 double iterationend;
36 std::ofstream outdata_realfish_LLH;
37 std::ofstream outdata_realfish_ECEF;
38
39 std::stringstream ss_target_ecef;
40 std::stringstream ss_target_llh;
41 std::string s_target_llh;
42 std::string s_target_ecef;
43
44 struct Task: public DUNE::Tasks::Task
45 {
46 //! Constructor.
47 //! @param[in] name task name.
48 //! @param[in] ctx context.
49
50
51 Task(const std::string& name, Tasks::Context& ctx):
52 DUNE::Tasks::Task(name, ctx)
53 {
54
55 }
56
57
58 //! Update internal state with new parameter values.
59 void
60 onUpdateParameters(void)
61 {
62 }
```

```
63
64  //! Reserve entity identifiers.
65  void
66  onEntityReservation(void)
67  {
68  }
69
70  //! Resolve entity names.
71  void
72  onEntityResolution(void)
73  {
74  }
75
76  //! Acquire resources.
77  void
78  onResourceAcquisition(void)
79  {
80  }
81
82  //! Initialize resources.
83  void
84  onResourceInitialization(void)
85  {
86  }
87
88  //! Release resources.
89  void
90  onResourceRelease(void)
91  {
92  }
93
94  //! Main loop.
95  void
96  onMain(void)
97  {
98
99  using namespace std;
100
101 // SIMULATION: create llh data covert 2 ecef then Kalman
        return target in llh
102 int tot_time_steps_target = 11999;
103 int tot_time_steps_usvs = 12000;
104 double time_step = 0.5;
105 double speedOfSound = 1484;
106 TOA_data TOAs;
```

```
107 TDOA_data TDOAs;
108 vector <TOA_data> toa_data_vector;
109 vector <double> depthvector;
110
111 //fetching MATLAB data
112 vector <double> target_data_ecef =
        createTarget_PosData_ECEF();
113 vector <double> target_data_llh = createTarget_PosData_LLH
        ();
114 vector <double> usv_GPSdata_ecef = createUSV_PosData_ECEF()
        ;
115
116 usvs receivers;
117 fish target_ecef;
118 fish target_llh;
119
120 tuple <int,double,double, double, double, double> tot_msg;
        // receiver_number, timestep, depth, x,y,z (ecef)
121 vector <tuple <int, double, double, double, double, double
        >> tot_msg_unsorted;
122
123 //  day: 04/04/2018 time: 04.04 (+00.00)
124 double unixtime = 1521394325;
125
126 //create data
127 int iterationend = 100;
128 for (int timestep = 0; timestep < iterationend; timestep++)
        { //timestep<1 when debug 13 when normal
129 usvspos2(usv_GPSdata_ecef, &receivers, timestep,
        tot_time_steps_usvs);
130 fishpos2(target_data_ecef, &target_ecef, timestep,
        tot_time_steps_target);
131 fishpos2(target_data_llh, &target_llh, timestep,
        tot_time_steps_target);
132 depthvector.push_back(target_llh.z);
133 getUsvsDist2Target(&target_ecef, &receivers);
134 TOA_data temp;
135 //createTOA(&temp, &receivers, timestep, speedOfSound);
136 temp.TOA_rec1 = receivers.rec1.dist2target / speedOfSound;
137 temp.TOA_rec2 = receivers.rec2.dist2target / speedOfSound;
138 temp.TOA_rec3 = receivers.rec3.dist2target / speedOfSound;
139 temp.TOA_rec4 = receivers.rec4.dist2target / speedOfSound;
140
141 temp.TOA_rec1 = temp.TOA_rec1 + 0.5* timestep + unixtime;
142 temp.TOA_rec2 = temp.TOA_rec2 + 0.5* timestep + unixtime;
```

```
143 temp.TOA_rec3 = temp.TOA_rec3 + 0.5* timestep + unixtime;
144 temp.TOA_rec4 = temp.TOA_rec4 + 0.5* timestep + unixtime;
145
146 tot_msg = make_tuple(1, temp.TOA_rec1, target_llh.z,
        receivers.rec1.x, receivers.rec1.y, receivers.rec1.z);
147 tot_msg_unsorted.push_back(tot_msg);
148 tot_msg = make_tuple(2, temp.TOA_rec2, target_llh.z,
        receivers.rec2.x, receivers.rec2.y, receivers.rec2.z);
149 tot_msg_unsorted.push_back(tot_msg);
150 tot_msg = make_tuple(3, temp.TOA_rec3, target_llh.z,
        receivers.rec3.x, receivers.rec3.y, receivers.rec3.z);
151 tot_msg_unsorted.push_back(tot_msg);
152 tot_msg = make_tuple(4, temp.TOA_rec4, target_llh.z,
        receivers.rec4.x, receivers.rec4.y, receivers.rec4.z);
153 tot_msg_unsorted.push_back(tot_msg);
154
155 }
156
157 //sort with respect to TOA for real time dispatchment
158 vector<tuple<int, double, double, double, double, double
        >> tot_msg_sorted = tot_msg_unsorted;
159 sort(tot_msg_sorted.begin(), tot_msg_sorted.end(),
        compare_tupple);
160
161 //create string messages
162 vector<string> TOAmessagesOut;
163 std::stringstream ss;
164 std::string s;
165 for (int i = 0; i < iterationend - 10; i ++) {
166 for (int j = 0; j < 4; j++) {
167 ss << "$TBR0" << to_string((get<0>(tot_msg_sorted[4 * i + j
        ]))) << ", " << to_string(get<1>(tot_msg_sorted[4 * i +
        j])) << ", " << 0 << ", S256, 2, " << to_string(get<2>(
        tot_msg_sorted[4 * i + j])) << ", " << to_string(50) <<
        ", " << "USV" << to_string((get<0>(tot_msg_sorted[4 * i
        + j]))) << ", " << to_string(get<3>(tot_msg_sorted[4 * i
         + j])) << ", " << to_string(get<4>(tot_msg_sorted[4 * i
         + j])) << ", " << to_string(get<5>(tot_msg_sorted[4 * i
         + j])) << endl;
168 s = ss.str();
169 TOAmessagesOut.push_back(s);
170 ss.str("");
171 }
172 }
173
```

```
174 //creating delay vector (aka wait vector)
175 vector<double> wait_vec;
176
177 vector <double> toa_vector;
178 toa_vector.push_back(unixtime);
179 vector <tuple <int, double, double, double, double, double
    >>::iterator x = tot_msg_sorted.begin();
180
181 //first wait
182 wait_vec.push_back(get<2>(*x)-unixtime);
183 x++;
184 //the rest
185 while (x!=tot_msg_sorted.end()){
186 wait_vec.push_back(get<2>(*(x+1))-get<2>(*x));
187 x++;
188 }
189
190 //saving messages
191 std::ofstream tbrmsgesout;
192 tbrmsgesout.open("/home/pi/dune/src/august/results/
    tbrmsgesout.txt", std::ios_base::app);
193 vector <string>::iterator q = TOAmessagesOut.begin();
194 while (q != TOAmessagesOut.end()) {
195 tbrmsgesout << *q ;
196 q++;
197 }
198 tbrmsgesout.close();
199 //dispatching msges to the IMC bus
200 vector <double>::iterator k = wait_vec.begin();
201 vector <string>::iterator i = TOAmessagesOut.begin();
202 while (!stopping())
203 {
204 int simspeed=1;
205 IMC::DevDataText msg;
206 msg.value = *i;
207 Delay::wait(*k / simspeed);
208 dispatch(msg);
209 debug(DTR("Dispatching receiver TOA message: %s"), msg.
    value.c_str());
210 //inf("Dispatching receiver TOA message: %s", msg->value.
    c_str());
211 i++;
212 k++;
213 // waitForMessages(1.0);
214 }
```

```
215 }
216 // vet ikke om denne skal v re her
217 };
218 }
219 }
220 }
221
222 DUNE_TASK
```

D.5 DUNE Task: EKF2

Code description
The main objective of the DUNE task EKF2 is the estimate the position of the target fish and dispatch the data to the local bus. The task produce solution estimates to the TDOA localization equations using an extended Kalman filter algorithm. The task consumes IMC messages of the type DevDataText (device data text) in real-time as they are dispatched to the IMC bus. The task differentiates between messages using IDs, where "TBR01", "TBR02", "TBR03" and "TBR04" corresponds to messages sent from USV 1,2,3 and 4, respectively. When a message is consumed, the data is saved, and the corresponding receiver data is set to be active. When four measurements are active, the Kalman filter runs after being initialized.

```
1
2  // Author: August
       *
3  // DUNE headers.
4  #include <DUNE/DUNE.hpp>
5
6  #include "../createTOAdataFromECEFdata/funkytions.h"
7  #include <string>
8  #include <sstream>
9
10 namespace
11 {
12 namespace august
13 {
14 namespace EKF2
15 {
16 using DUNE_NAMESPACES;
17
18
19 struct Task: public DUNE::Tasks::Task
20 {
```

```
21 //! Constructor.
22 //! @param[in] name task name.
23 //! @param[in] ctx context.
24
25 EKF matrixxes;
26 datamsges datastruct;
27 std::string TBR_ID;
28 std::string USV_ID;
29 std::vector<std::string> tempStringVec;
30 std::string temp_position;
31 std::stringstream ss;
32 std::string s;
33 std::string TBR_receiver1_ID;
34 std::string TBR_receiver2_ID;
35 std::string TBR_receiver3_ID;
36 std::string TBR_receiver4_ID;
37 IMC::DevDataText msg_out;
38 double speedOfSound = 1484;
39 Task(const std::string& name, Tasks::Context& ctx):
40 DUNE::Tasks::Task(name, ctx)
41 {
42 // PARAMETERS
43 param("TBR700 receiver 1 ID", TBR_receiver1_ID)
44 .description("TBR700 receiver 1 ID")
45 .defaultValue("TBR01");
46
47 param("TBR700 receiver 2 ID", TBR_receiver2_ID)
48 .description("TBR700 receiver 2 ID")
49 .defaultValue("TBR02");
50
51 param("TBR700 receiver 3 ID", TBR_receiver3_ID)
52 .description("TBR700 receiver 3 ID")
53 .defaultValue("TBR03");
54
55 param("TBR700 receiver 4 ID", TBR_receiver4_ID)
56 .description("TBR700 receiver 4 ID")
57 .defaultValue("TBR04");
58
59 bind<IMC::DevDataText>(this);
60 }
61
62 //! Update internal state with new parameter values.
63 void
64 onUpdateParameters(void)
65 {
```

```
66  }
67
68  //! Reserve entity identifiers.
69  void
70  onEntityReservation(void)
71  {
72  }
73
74  //! Resolve entity names.
75  void
76  onEntityResolution(void)
77  {
78  }
79
80  //! Acquire resources.
81  void
82  onResourceAcquisition(void)
83  {
84  }
85
86  //! Initialize resources.
87  void
88  onResourceInitialization(void)
89  {
90  }
91
92  //! Release resources.
93  void
94  onResourceRelease(void)
95  {
96  }
97  void  initEKF2(EKF * matrixxes, fish * x_hat_init) {
98  matrixxes->init = 1;
99
100 double x_hatty[] = { x_hat_init->x,
101 x_hat_init->y,
102 x_hat_init->z,
103 0,
104 0,
105 0 };
106 matrixxes->x_hat.fill(6, 1, x_hatty);
107 matrixxes->x_hat_pred = matrixxes->x_hat;
108
109 double time_step = 0.5;
110 double Fd[] = { 1, 0, 0, time_step, 0, 0,
```

```
111 0, 1, 0, 0, time_step, 0,
112 0, 0, 1, 0, 0, time_step,
113 0, 0, 0, 1, 0, 0,
114 0, 0, 0, 0, 1, 0,
115 0, 0, 0, 0, 0, 1 };
116 //Matrix F;
117 matrixxes->F.fill(6, 6, Fd);
118
119 double Dd[] = { time_step, 0, 0, 0, 0, 0,
120 0, time_step, 0, 0, 0, 0,
121 0, 0, time_step, 0, 0, 0,
122 0, 0, 0, time_step, 0, 0,
123 0, 0, 0, 0, time_step, 0,
124 0, 0, 0, 0, 0, time_step };
125 //Matrix D;
126 matrixxes->D.fill(6, 6, Dd);
127 double P_factor = 0.01;
128
129 double P_hatd[] = { P_factor, 0, 0, 0, 0, 0,
130 0, P_factor, 0, 0, 0, 0,
131 0, 0, P_factor, 0, 0, 0,
132 0, 0, 0, P_factor, 0, 0,
133 0, 0, 0, 0, P_factor, 0,
134 0, 0, 0, 0, 0, P_factor };
135 //Matrix P_hat;
136 matrixxes->P_hat.fill(6, 6, P_hatd);
137
138 double Qd[] = { P_factor, 0, 0, 0, 0, 0,
139 0, P_factor, 0, 0, 0, 0,
140 0, 0, 1000, 0, 0, 0,
141 0, 0, 0, P_factor, 0, 0,
142 0, 0, 0, 0, P_factor, 0,
143 0, 0, 0, 0, 0, P_factor }; // trust depth more.
144
145 double Rd[] = { 1, 0, 0, 0,
146 0, 1, 0, 0,
147 0, 0, 1, 0,
148 0, 0, 0, 0.001 }; //we trust fourth measurement alot
    because its the depth meassurement.
149 matrixxes->R.fill(4, 4, Rd);
150 //Matrix Q;
151 matrixxes->Q.fill(6, 6, Qd);
152 }
153
154 void runEKF(EKF * matrixxes, datamsges * datastruct, fish *
```

```
    x_hat_init) {
155
156 if (matrixxes->init != 1) {
157 initEKF2(matrixxes, x_hat_init);
158 debug(DTR("EKF is initialized: "));
159 }
160 usvs boats;
161 boats.rec1.x = datastruct->data1.lon;
162 boats.rec1.y = datastruct->data1.lat;
163 boats.rec1.z = datastruct->data1.hae;
164
165 boats.rec2.x = datastruct->data2.lon;
166 boats.rec2.y = datastruct->data2.lat;
167 boats.rec2.z = datastruct->data2.hae;
168
169 boats.rec3.x = datastruct->data3.lon;
170 boats.rec3.y = datastruct->data3.lat;
171 boats.rec3.z = datastruct->data3.hae;
172
173 boats.rec4.x = datastruct->data4.lon;
174 boats.rec4.y = datastruct->data4.lat;
175 boats.rec4.z = datastruct->data4.hae;
176
177 fish x_hatie;
178 x_hatie.x = matrixxes->x_hat(0, 0);
179 x_hatie.y = matrixxes->x_hat(1, 0);
180 x_hatie.z = matrixxes->x_hat(2, 0);
181
182 pseudoranges pseudos;
183 createPseudorange(&pseudos, &x_hatie, &boats);
184
185 double depth = datastruct->data1.depth;
186
187 double y_hat_d[] = { pseudos.rec2 - pseudos.rec1,
188 pseudos.rec3 - pseudos.rec1,
189 pseudos.rec4 - pseudos.rec1,
190 depth };
191
192 matrixxes->y_hat.fill(4, 1, y_hat_d);
193 createJacobian6(&matrixxes->H, &boats, &x_hatie);
194
195 double lat, lon, hae;
196 fromECEF(matrixxes->x_hat(0, 0), matrixxes->x_hat(1, 0),
    matrixxes->x_hat(2, 0), &lat, &lon, &hae);
197
```

```
198 double TDOA21 = datastruct->data2.seconds - datastruct->
        data1.seconds;
199 double TDOA31 = datastruct->data3.seconds - datastruct->
        data1.seconds;
200 double TDOA41 = datastruct->data4.seconds - datastruct->
        data1.seconds;
201
202 double speedOfSound = 1484;
203
204 double y_d[] = { speedOfSound* TDOA21,
205 speedOfSound*TDOA31,
206 speedOfSound*TDOA41,
207 hae };
208
209 matrixxes->y.fill(4, 1, y_d);
210 matrixxes->x_hat_pred = matrixxes->F*matrixxes->x_hat;
211 matrixxes->P_pred = matrixxes->F*matrixxes->P_hat*transpose
        (matrixxes->F) + matrixxes->D*matrixxes->Q*transpose(
        matrixxes->D);
212 matrixxes->K = matrixxes->P_pred*transpose(matrixxes->H)*
        inverse(matrixxes->H*matrixxes->P_pred*transpose(
        matrixxes->H) + matrixxes->R);
213 matrixxes->P_hat = matrixxes->P_pred - matrixxes->K*
        matrixxes->H*matrixxes->P_pred;
214 matrixxes->x_hat = matrixxes->x_hat_pred + matrixxes->K*(
        matrixxes->y_hat - matrixxes->y);
215
216
217 //// DISPATCHING & saving ECEF x_hat
218 ss << "$x_hat_ECEF, "<< setprecision(16)<< matrixxes->x_hat
        (0, 0) << ", " << matrixxes->x_hat(1, 0) << ", " <<
        matrixxes->x_hat(2, 0) <<", " << matrixxes->x_hat(3, 0)
         << ", " << matrixxes->x_hat(4, 0) << ", " << matrixxes
        ->x_hat(5, 0) ;
219 s = ss.str();
220 msg_out.value = s;
221 //saving ECEF messages
222 ofstream ecefmsgesout;
223 ecefmsgesout.open("/home/pi/dune/src/august/results/x_hat-
        ecef.txt", std::ios_base::app);
224 ecefmsgesout << s <<endl;
225 ecefmsgesout.close();
226 dispatch(msg_out);
227 debug(DTR("%s"), msg_out.value.c_str());
228 ss.str("");
```

```
229
230
231 // DISPATCHING LLH x_hat
232 fish temp;
233 temp.x = matrixxes->x_hat(0, 0);
234 temp.y = matrixxes->x_hat(1, 0);
235 temp.z = matrixxes->x_hat(2, 0);
236 ecef2llhFISH(&temp);
237 ss << "$x_hat_LLH, "<<setprecision(16)<< temp.x << ", "<<
    temp.y << ", "<< temp.z;
238 s = ss.str();
239 //saving LLH messages
240 ofstream llhmsgesout;
241 llhmsgesout.open("/home/pi/dune/src/august/results/x_hat-
    llh.txt", std::ios_base::app);
242 llhmsgesout << s <<endl;
243 llhmsgesout.close();
244 msg_out.value = s;
245 dispatch(msg_out);
246 debug(DTR("%s"), msg_out.value.c_str());
247 ss.str("");
248
249
250 //resetting EKF
251 datastruct->data1.active = 0;
252 datastruct->data2.active = 0;
253 datastruct->data3.active = 0;
254 datastruct->data4.active = 0;
255 }
256
257 void
258 consume(const IMC::DevDataText* msg){
259
260
261 fish target;
262 target.x = 2811674.127241466;
263 target.y = 515158.4675443817;
264 target.z = 5682591.148048251;
265
266 //creating offset
267 fish x_hat_init = target;
268 x_hat_init.x++; x_hat_init.x++; x_hat_init.x++; x_hat_init.
    y++; x_hat_init.x++; x_hat_init.z++;
269 x_hat_init.x++; x_hat_init.x++; x_hat_init.x++; x_hat_init.
    y++; x_hat_init.x++; x_hat_init.z++;
```

```
fish x_hat = x_hat_init;

tempStringVec = splitString(msg->value);

if (tempStringVec[0] == TBR_receiver1_ID){
datastruct.data1 = parseFakeFishTagData2(msg->value);
datastruct.data1.active = 1;
}
else if (tempStringVec[0] == TBR_receiver2_ID){
datastruct.data2 = parseFakeFishTagData2(msg->value);
datastruct.data2.active = 1;
}

else if (tempStringVec[0] == TBR_receiver3_ID){
datastruct.data3 = parseFakeFishTagData2(msg->value);
datastruct.data3.active = 1;
}

else if (tempStringVec[0] == TBR_receiver4_ID){
datastruct.data4 = parseFakeFishTagData2(msg->value);
datastruct.data4.active = 1;
}

if ((datastruct.data1.active ==1) && (datastruct.data2.
    active == 1) && (datastruct.data3.active == 1) && (
    datastruct.data4.active == 1)) {
runEKF(& matrixxes, & datastruct, &x_hat_init);
}
}

//! Main loop.
void
onMain(void)
{
while (!stopping())
{
waitForMessages(1.0);
}
}
};
}
}
}

DUNE_TASK
```

D.6 DUNE Task: createErrorPlots

```
1  // Author: August
       *
2
3  #include <DUNE/DUNE.hpp>
4  #include "../createTOAdataFromECEFdata/funkytions.h"
5  #include <string>
6  #include <sstream>
7  namespace august
8  {
9  namespace createErrorPlots
10 {
11 using DUNE_NAMESPACES;
12
13 struct Task: public DUNE::Tasks::Task
14 {
15 //! Constructor.
16 //! @param[in] name task name.
17 //! @param[in] ctx context.
18 Task(const std::string& name, Tasks::Context& ctx):
19 DUNE::Tasks::Task(name, ctx)
20 {
21 }
22
23 //! Update internal state with new parameter values.
24 void
25 onUpdateParameters(void)
26 {
27 }
28
29 //! Reserve entity identifiers.
30 void
31 onEntityReservation(void)
32 {
33 }
34
35 //! Resolve entity names.
36 void
37 onEntityResolution(void)
38 {
39 }
40
41 //! Acquire resources.
42 void
```

```
43 onResourceAcquisition(void)
44 {
45 }
46
47 //! Initialize resources.
48 void
49 onResourceInitialization(void)
50 {
51 }
52
53 //! Release resources.
54 void
55 onResourceRelease(void)
56 {
57 }
58
59 //! Main loop.
60 void
61 onMain(void)
62 {
63
64 vector<double> target_data_ecef =
       createTarget_PosData_ECEF();
65 vector<double> target_data_llh = createTarget_PosData_LLH
       ();
66 usvs receivers;
67 fish target_ecef;
68 fish target_llh;
69 double tot_time_steps_target = 11999;
70
71 std::string line;
72 std::ifstream x_hat_ecef_data("/home/pi/dune/src/august/
       results/x_hat-ecef.txt");
73 std::ifstream x_hat_llh_data("/home/pi/dune/src/august/
       results/x_hat-llh.txt");
74
75 for (int i = 0; i < 80; i++) {
76 fishpos2(target_data_ecef, &target_ecef, i,
       tot_time_steps_target);
77 fishpos2(target_data_llh, &target_llh, i,
       tot_time_steps_target);
78 string s;
79 std::stringstream ss;
80 //creating ecef errors
81 getline(x_hat_ecef_data, line);
```

```
82 vector <string> x_hat_ecef_msg = splitString(line);
83 ss << setprecision(16)<< target_ecef.x - atof(
       x_hat_ecef_msg[1].c_str()) << ", " << target_ecef.y -
       atof(x_hat_ecef_msg[2].c_str()) << ", " << target_ecef.z
        - atof(x_hat_ecef_msg[3].c_str());
84 s = ss.str();
85 //saving ECEF error
86 ofstream ecefmsgesout;
87 ecefmsgesout.open("/home/pi/dune/src/august/results/
       ecef_errors.txt", std::ios_base::app);
88 ecefmsgesout << s <<endl;
89 ecefmsgesout.close();
90 ss.str("");
91
92 //creating llh errors
93 getline(x_hat_llh_data, line);
94 vector <string> x_hat_llh_msg = splitString(line);
95 ss << setprecision(16)<< target_llh.x - atof(x_hat_llh_msg
       [1].c_str()) << ", " << target_llh.y - atof(
       x_hat_llh_msg[2].c_str()) << ", " << target_llh.z - atof
       (x_hat_llh_msg[3].c_str());
96 s = ss.str();
97 //saving LLH error
98 ofstream llhmsgesout;
99 llhmsgesout.open("/home/pi/dune/src/august/results/
       llh_errors.txt", std::ios_base::app);
100 llhmsgesout << s <<endl;
101 llhmsgesout.close();
102 ss.str("");
103 }
104 while (!stopping())
105 {
106 waitForMessages(1.0);
107 }
108 }
109 };
110 }
111 }
112
113 DUNE_TASK
```

D.7 funkytions.h

```
1 #include <cstdlib>
2 #include <cmath>
3 #include <cstring>
```

```
4  #include <string>
5  #include <vector>
6  //#include <iostream>
7  #include <fstream>
8  #include <cctype>
9  #include <algorithm>
10 #include <fstream>
11 #include <vector>
12 #include <iostream>
13 #include <algorithm>
14 #include <sstream>
15 #include <math.h>
16 #include <utility>
17 #include <tuple>
18 #include <fstream>
19 using namespace std;
20 #include <DUNE/DUNE.hpp>
21
22 struct fish {
23         double x;
24         double y;
25         double z;
26 };
27
28 struct usv {
29         double x;
30         double y;
31         double z;
32         double dist2target;
33 };
34
35 struct usvs {
36         usv rec1;
37         usv rec2;
38         usv rec3;
39         usv rec4;
40 };
41
42 struct TOA_data {
43         double TOA_rec1;
44         double TOA_rec2;
45         double TOA_rec3;
46         double TOA_rec4;
47 };
48
```

```
49 struct TDOA_data {
50         double TDOA_21;
51         double TDOA_31;
52         double TDOA_41;
53 };
54
55 struct receiver_data
56 {
57         std::string msg;
58         int receiver_number;
59         double seconds;
60         int milliseconds;
61         int fishtagID;
62         double depth;
63         int SNRratio;
64         double lon;
65         double lat;
66         double hae;
67         int active;
68 };
69
70 struct range_real {
71         double rec1;
72         double rec2;
73         double rec3;
74         double rec4;
75 };
76
77 struct pseudoranges {
78         double rec1;
79         double rec2;
80         double rec3;
81         double rec4;
82 };
83
84 struct EKF {
85         int init;
86         Matrix x_hat;
87         Matrix F;
88         Matrix D;
89         Matrix P_hat;
90         Matrix R;
91         Matrix Q;
92         Matrix H;
93
```

```
94          Matrix y_hat;
95          Matrix x_hat_pred;
96          Matrix y;
97          Matrix P_pred;
98          Matrix K;
99          fish testfish;
100 };
101
102 struct datamsges {
103          receiver_data data1;
104          receiver_data data2;
105          receiver_data data3;
106          receiver_data data4;
107 };
108
109 vector <double> createUSV_PosData_LLH() ;
110 vector <double> createUSV_PosData_ECEF() ;
111 vector <double> createTarget_PosData_ECEF() ;
112 vector <double> createTarget_PosData_LLH() ;
113 bool compare_tupple(const tuple<int, double, double, double
    , double, double>&i, const tuple<int, double, double,
    double, double, double>&j);
114 void fishpos2(vector <double> target_data, fish *
    fishobject, int timestep, int tot_timesteps);
115 void usvpos2(vector <double> usv_data, usv * usvobject, int
     timestep, int receiver, int tot_timesteps);
116 void usvspos2(vector <double> usv_data, usvs * usvobjects,
    int timestep, int tot_timesteps);
117 double euclid3(fish * fish_pos, usv * usv_pos) ;
118 double euclid3mat(Matrix x_hat, usv * usv_pos);
119 void getUsvsDist2Target(fish * fishobject, usvs *
    usvobjects) ;
120 void createTOA(TOA_data * TOAs, usvs * receivers, double
    timestep, double speedOfSound) ;
121 void createTDOA(TOA_data * TOAs, TDOA_data * TDOAs) ;
122 vector<string> splitString(string s);
123 receiver_data parseFakeFishTagData2(string msg);
124 void calculate_y_hat(Matrix * y_hat, Matrix x_hat, usvs *
    receivers, double depth);
125 void createJacobian2(Matrix * H, usvs * receivers, fish *
    x_hat);
126 void create_y(Matrix * y, TDOA_data * TDOAs, Matrix
    x_hat_pred, double speedOfSound);
127 double computeRn(double lat);
128 void fromECEF(double x, double y, double z, double* lat,
```

```
    double* lon, double* hae);
129 void toECEF(double lat, double lon, double hae, double* x,
    double* y, double* z);
130 void createJacobian6(Matrix * H, usvs * receivers, fish *
    x_hat) ;
131 void getRange(range_real * ranges, fish * target, usvs *
    boats);
132 void getTDOAs(TDOA_data * tdoas, range_real * ranges,
    double soundspeed);
133 void createPseudorange(pseudoranges * pseudos, fish * x_hat
    , usvs * boats);
```

D.8 funkytions.cpp

```
1  #include <cstdlib>
2  #include <cmath>
3  #include <cstring>
4  #include <string>
5  #include <vector>
6  //#include <iostream>
7  #include <fstream>
8  #include <cctype>
9  #include <algorithm>
10 #include <fstream>
11 #include <vector>
12 #include <iostream>
13 #include <algorithm>
14 #include <sstream>
15 #include <math.h>
16 #include <utility>
17 #include <tuple>
18 #include "funkytions.h"
19 using namespace std;
20 #include <DUNE/DUNE.hpp>
21 double computeRn(double lat)
22 {
23 // Code from Dune/Coordinates/WGS84
24 static const double c_wgs84_a = 6378137.0;
25 static const double c_wgs84_e2 = 0.00669437999013;
26 double lat_sin = std::sin(lat);
27 return c_wgs84_a / std::sqrt(1 - c_wgs84_e2 * (lat_sin *
    lat_sin));
28 }
29
30 void fishpos2(vector<double> target_data, fish *
    fishobject, int timestep, int tot_timesteps) {
```

```
31 vector<double>::const_iterator i = target_data.begin();
32 fishobject->x = *(i + timestep);
33 fishobject->y = *(i + 1 + timestep + tot_timesteps);
34 fishobject->z = *(i + 2 + timestep + tot_timesteps * 2);
35 }
36
37 void usvpos2(vector <double> usv_data, usv * usvobject, int
       timestep, int receiver, int tot_timesteps) {
38 vector<double>::const_iterator i = usv_data.begin() +
      timestep * 3 + 3 * tot_timesteps * (receiver - 1);
39 usvobject->x = *i;
40 usvobject->y = *(i + 1);
41 usvobject->z = *(i + 2);
42 //return usvobject;
43 }
44
45 void usvspos2(vector <double> usv_data, usvs * usvobjects,
      int timestep, int tot_timesteps) {
46 usvpos2(usv_data, &usvobjects->rec1, timestep, 1,
      tot_timesteps);
47 usvpos2(usv_data, &usvobjects->rec2, timestep, 2,
      tot_timesteps);
48 usvpos2(usv_data, &usvobjects->rec3, timestep, 3,
      tot_timesteps);
49 usvpos2(usv_data, &usvobjects->rec4, timestep, 4,
      tot_timesteps);
50 }
51
52 void fromECEF(double x, double y, double z, double* lat,
      double* lon, double* hae)
53 {
54 // Code from Dune/Coordinates/WGS84
55 //! Semi-major axis.
56 static const double c_wgs84_a = 6378137.0;
57 //! Semi-minor axis.
58 static const double c_wgs84_b = 6356752.3142;
59 //! First eccentricity squared.
60 static const double c_wgs84_e2 = 0.00669437999013;
61 //! Second (prime) eccentricity squared.
62 static const double c_wgs84_ep2 = 0.00673949674228;
63 //! Flattening.
64 static const double c_wgs84_f = 0.0033528106647475;
65 //assert(lat != 0);
66 //assert(lon != 0);
67 //assert(hae != 0);
```

```
68
69  double p = std::sqrt(x * x + y * y);
70  *lon = std::atan2(y, x);
71  *lat = std::atan2(z / p, 0.01);
72  double n = computeRn(*lat);
73  *hae = p / std::cos(*lat) - n;
74  double old_hae = -1e-9;
75  double num = z / p;
76
77  while (std::fabs(*hae - old_hae) > 1e-4)
78  {
79  old_hae = *hae;
80  double den = 1 - c_wgs84_e2 * n / (n + *hae);
81  *lat = std::atan2(num, den);
82  n = computeRn(*lat);
83  *hae = p / std::cos(*lat) - n;
84  }
85  }
86
87  void toECEF(double lat, double lon, double hae, double* x,
        double* y, double* z)
88  {
89  // Code from Dune/Coordinates/WGS84
90  //! Semi-major axis.
91  static const double c_wgs84_a = 6378137.0;
92  //! Semi-minor axis.
93  static const double c_wgs84_b = 6356752.3142;
94  //! First eccentricity squared.
95  static const double c_wgs84_e2 = 0.00669437999013;
96  //! Second (prime) eccentricity squared.
97  static const double c_wgs84_ep2 = 0.00673949674228;
98  //! Flattening.
99  static const double c_wgs84_f = 0.0033528106647475;
100 /*assert(x != 0);
101 assert(y != 0);
102 assert(z != 0);*/
103 double cos_lat = std::cos(lat);
104 double sin_lat = std::sin(lat);
105 double cos_lon = std::cos(lon);
106 double sin_lon = std::sin(lon);
107 double rn = computeRn(lat);
108 *x = (rn + hae) * cos_lat * cos_lon;
109 *y = (rn + hae) * cos_lat * sin_lon;
110 *z = (((1.0 - c_wgs84_e2) * rn) + hae) * sin_lat;
111 }
```

```
112
113 vector <double> createUSV_PosData_LLH() {
114 ifstream infile2("/home/pi/dune/src/august/data/usv_pos_llh
        .txt");
115 if (infile2.is_open()) {
116 cout << "fila usv_pos_llh.txt er  pna " << endl;
117 }
118 else {
119 cout << "fila usv_pos_llh.txt er ikke funnet " << endl;
120 }
121
122 vector<double> record2;
123 while (infile2) {
124 string s;
125
126 if (!getline(infile2, s))break;
127 istringstream ss(s);
128
129
130 while (ss)
131 {
132
133 string s;
134 if (!getline(ss, s, ','))break;
135 record2.push_back(atof(s.c_str()));
136
137 }
138
139 }
140 return record2;
141 }
142
143 vector <double> createUSV_PosData_ECEF() {
144
145 ifstream infile2("/home/pi/dune/src/august/data/
        usv_pos_ecef.txt");
146 if (infile2.is_open()) {
147 cout << "fila er usv_pos_ECEF.txt  pna" << endl;
148 }
149 else {
150 cout << "fila usv_pos_ECEF.txt er ikke funnet " << endl;
151 }
152
153 vector<double> record2;
154 while (infile2) {
```

```
155 string s;
156
157 if (!getline(infile2, s))break;
158 istringstream ss(s);
159
160
161 while (ss)
162 {
163
164 string s;
165 if (!getline(ss, s, ','))break;
166 record2.push_back(atof(s.c_str()));
167
168 }
169
170 }
171 return record2;
172 }
173
174
175 vector<double> createTarget_PosData_ECEF() {
176
177 ifstream infile2("/home/pi/dune/src/august/data/target_ecef
    .txt");
178 if (infile2.is_open()) {
179 cout << "fila target_ecef.txt er  pna " << endl;
180 }
181 else {
182 cout << "fila target_ecef.txt er ikke funnet " << endl;
183 }
184
185 vector<double> record2;
186 while (infile2) {
187 string s;
188
189 if (!getline(infile2, s))break;
190 istringstream ss(s);
191
192
193 while (ss)
194 {
195
196 string s;
197 if (!getline(ss, s, ','))break;
198 record2.push_back(atof(s.c_str()));
```

```
199
200 }
201
202 }
203 return record2;
204 }
205
206 vector <double> createTarget_PosData_LLH() {
207
208 ifstream infile2("/home/pi/dune/src/august/data/target_llh.
    txt");
209 if (infile2.is_open()) {
210 cout << "fila target_llh.txt er  pna" << endl;
211 }
212 else {
213 cout << "fila target_llh.txt er ikke funnet " << endl;
214 }
215
216 vector<double> record2;
217 while (infile2) {
218 string s;
219
220 if (!getline(infile2, s))break;
221 istringstream ss(s);
222
223
224 while (ss)
225 {
226
227 string s;
228 if (!getline(ss, s, ','))break;
229 record2.push_back(atof(s.c_str()));
230
231 }
232
233 }
234 return record2;
235 }
236
237 bool compare_tupple(const tuple<int, double, double, double
    , double, double>&i, const tuple<int, double, double,
    double, double, double>&j)
238 {
239 return get<1>(i) < get<1>(j);
240 //return i.second < j.second;
```

```
241 }
242
243 double euclid3(fish * fish_pos, usv * usv_pos) {
244 return (sqrt(pow((fish_pos->x - usv_pos->x), 2) + pow((
        fish_pos->y - usv_pos->y), 2) + pow((fish_pos->z -
        usv_pos->z), 2)));
245 }
246
247 double euclid3mat(Matrix x_hat, usv * usv_pos) {
248 return (sqrt(pow((x_hat(0, 0) - usv_pos->x), 2) + pow((
        x_hat(1, 0) - usv_pos->y), 2) + pow((x_hat(2, 0) -
        usv_pos->z), 2)));
249 }
250
251 void getUsvsDist2Target(fish * fishobject, usvs *
        usvobjects) {
252 usvobjects->rec1.dist2target = (euclid3(fishobject, &
        usvobjects->rec1));
253 usvobjects->rec2.dist2target = (euclid3(fishobject, &
        usvobjects->rec2));
254 usvobjects->rec3.dist2target = (euclid3(fishobject, &
        usvobjects->rec3));
255 usvobjects->rec4.dist2target = (euclid3(fishobject, &
        usvobjects->rec4));
256 }
257
258 void createTOA(TOA_data * TOAs, usvs * receivers, double
        timestep, double speedOfSound) {
259 TOAs->TOA_rec1 = (double)(receivers->rec1.dist2target) / (
        double)speedOfSound + timestep*0.5;
260 TOAs->TOA_rec2 = (double)(receivers->rec2.dist2target) / (
        double)speedOfSound + timestep*0.5;
261 TOAs->TOA_rec3 = (double)(receivers->rec3.dist2target) / (
        double)speedOfSound + timestep*0.5;
262 TOAs->TOA_rec4 = (double)(receivers->rec4.dist2target) / (
        double)speedOfSound + timestep*0.5;
263 }
264
265 void createTDOA(TOA_data * TOAs, TDOA_data * TDOAs) {
266 TDOAs->TDOA_21 = 1484 * (TOAs->TOA_rec2 - TOAs->TOA_rec1);
267 TDOAs->TDOA_31 = 1484 * (TOAs->TOA_rec3 - TOAs->TOA_rec1);
268 TDOAs->TDOA_41 = 1484 * (TOAs->TOA_rec4 - TOAs->TOA_rec1);
269 }
270
271 vector<string>
```

```
272 splitString(string s)//$TBR05, 1446716612, 123, S256, 2,
        233, 50
273 {
274 replace(s.begin(), s.end(), '$', ' ');
275 replace(s.begin(), s.end(), '*', ' ');
276 replace(s.begin(), s.end(), ',', ' ');
277 size_t found = s.find("  ");
278 if (found != string::npos)
279 {
280 s.replace(found, 2, " ");
281 //debug(DTR("Found double space"));
282 }
283
284 //debug(DTR("MSG %s"), s.c_str());
285
286 istringstream stream(s);
287 vector<string> result;
288 for (;; )
289 {
290 string word;
291 if (!(stream >> word))
292 {
293 break;
294 }
295 result.push_back(word);
296 }
297 return result;
298 }
299
300 receiver_data parseFakeFishTagData2(string msg) {
301
302 receiver_data data;
303 vector<string> data_fields = splitString(msg);
304 //$!TBR05, 1446716612, 123, S256, 2, 233, 50, lat, lon, hae
305 if (data_fields.size() == 11) {
306 data.msg = msg.c_str();
307 data.seconds = atof(data_fields[1].c_str());
308 data.milliseconds = atof(data_fields[2].c_str());
309 data.fishtagID = atof(data_fields[4].c_str());
310 data.depth = atof(data_fields[5].c_str()); // assumed depth
311 data.SNRratio = atof(data_fields[6].c_str());
312 data.lon = atof(data_fields[8].c_str());
313 data.lat = atof(data_fields[9].c_str());
314 data.hae = atof(data_fields[10].c_str());
315 }
```

```
316 else
317 {
318 //inf(DTR("Wrong number of fields in the message, received
        %d"), (int)data_fields.size());
319 }
320
321 return data;
322 }
323
324 void calculate_y_hat(Matrix * y_hat, Matrix x_hat, usvs *
        receivers, double depth) {
325 double pseudorangeRef = euclid3mat(x_hat, &receivers->rec1)
        ;
326 double pseudorange2 = euclid3mat(x_hat, &receivers->rec2);
327 double pseudorange3 = euclid3mat(x_hat, &receivers->rec3);
328 double pseudorange4 = euclid3mat(x_hat, &receivers->rec4);
329
330 double y_21 = pseudorange2 - pseudorangeRef;
331 double y_31 = pseudorange3 - pseudorangeRef;
332 double y_41 = pseudorange4 - pseudorangeRef;
333 //double y_depth = target->z; // assume perfect depth
        measurement for now
334
335 double temp[] = { y_21,
336 y_31,
337 y_41,
338 depth };
339
340 y_hat->fill(4, 1, temp);
341 }
342
343 void createJacobian6(Matrix * H, usvs * receivers, fish *
        x_hat) {
344 double x = x_hat->x;
345 double y = x_hat->y;
346 double z = x_hat->z;
347 //temp = zeros(m - 1, 6);
348 double x_ref = receivers->rec1.x;
349 double y_ref = receivers->rec1.y;
350 double z_ref = receivers->rec1.z;
351
352 ///for j = 1:m - 1
353 double x_rj = receivers->rec2.x;
354 double y_rj = receivers->rec2.y;
355 double z_rj = receivers->rec2.z;
```

```
356 double h_top1 = (2 * x - 2 * x_ref) / (2 * sqrt(pow((x -
        x_ref), 2) + pow((y - y_ref), 2) + pow((z - z_ref), 2)))
         - (2 * x - 2 * x_rj) / (2 * sqrt(pow((x - x_rj), 2) +
        pow((y - y_rj), 2) + pow((z - z_rj), 2)));
357 double h_top2 = (2 * y - 2 * y_ref) / (2 * sqrt(pow((x -
        x_ref), 2) + pow((y - y_ref), 2) + pow((z - z_ref), 2)))
         - (2 * y - 2 * y_rj) / (2 * sqrt(pow((x - x_rj), 2) +
        pow((y - y_rj), 2) + pow((z - z_rj), 2)));
358 double h_top3 = (2 * z - 2 * z_ref) / (2 * sqrt(pow((x -
        x_ref), 2) + pow((y - y_ref), 2) + pow((z - z_ref), 2)))
         - (2 * z - 2 * z_rj) / (2 * sqrt(pow((x - x_rj), 2) +
        pow((y - y_rj), 2) + pow((z - z_rj), 2)));
359
360 x_rj = receivers->rec3.x;
361 y_rj = receivers->rec3.y;
362 z_rj = receivers->rec3.z;
363 double h_mid1 = (2 * x - 2 * x_ref) / (2 * sqrt(pow((x -
        x_ref), 2) + pow((y - y_ref), 2) + pow((z - z_ref), 2)))
         - (2 * x - 2 * x_rj) / (2 * sqrt(pow((x - x_rj), 2) +
        pow((y - y_rj), 2) + pow((z - z_rj), 2)));
364 double h_mid2 = (2 * y - 2 * y_ref) / (2 * sqrt(pow((x -
        x_ref), 2) + pow((y - y_ref), 2) + pow((z - z_ref), 2)))
         - (2 * y - 2 * y_rj) / (2 * sqrt(pow((x - x_rj), 2) +
        pow((y - y_rj), 2) + pow((z - z_rj), 2)));
365 double h_mid3 = (2 * z - 2 * z_ref) / (2 * sqrt(pow((x -
        x_ref), 2) + pow((y - y_ref), 2) + pow((z - z_ref), 2)))
         - (2 * z - 2 * z_rj) / (2 * sqrt(pow((x - x_rj), 2) +
        pow((y - y_rj), 2) + pow((z - z_rj), 2)));
366
367 x_rj = receivers->rec4.x;
368 y_rj = receivers->rec4.y;
369 z_rj = receivers->rec4.z;
370 double h_bot1 = (2 * x - 2 * x_ref) / (2 * sqrt(pow((x -
        x_ref), 2) + pow((y - y_ref), 2) + pow((z - z_ref), 2)))
         - (2 * x - 2 * x_rj) / (2 * sqrt(pow((x - x_rj), 2) +
        pow((y - y_rj), 2) + pow((z - z_rj), 2)));
371 double h_bot2 = (2 * y - 2 * y_ref) / (2 * sqrt(pow((x -
        x_ref), 2) + pow((y - y_ref), 2) + pow((z - z_ref), 2)))
         - (2 * y - 2 * y_rj) / (2 * sqrt(pow((x - x_rj), 2) +
        pow((y - y_rj), 2) + pow((z - z_rj), 2)));
372 double h_bot3 = (2 * z - 2 * z_ref) / (2 * sqrt(pow((x -
        x_ref), 2) + pow((y - y_ref), 2) + pow((z - z_ref), 2)))
         - (2 * z - 2 * z_rj) / (2 * sqrt(pow((x - x_rj), 2) +
        pow((y - y_rj), 2) + pow((z - z_rj), 2)));
373
```

```
374 double temp[] = { h_top1, h_top2, h_top3, 0, 0, 0,
375 h_mid1, h_mid2, h_mid3, 0, 0, 0,
376 h_bot1, h_bot2, h_bot3, 0, 0, 0,
377 0, 0, 1, 0, 0, 0 };
378
379 /*double h_temp = { h_top, 0, 0, 0,
380 h_mid, 0, 0, 0,
381 h_bot, 0, 0, 0,
382 0,0,1, 0, 0, 0 };*/
383 H->fill(4, 6, temp);
384 };
385
386 void create_y(Matrix * y, TDOA_data * TDOAs, Matrix
        x_hat_pred, double speedOfSound) {
387 double y_1 = TDOAs->TDOA_21;
388 double y_2 = TDOAs->TDOA_31;
389 double y_3 = TDOAs->TDOA_41;
390 double lat;
391 double lon;
392 double hae;
393 fromECEF(x_hat_pred(0, 0), x_hat_pred(1, 0), x_hat_pred(2,
        0), &lat, &lon, &hae);
394 double y_4 = hae;
395 double yd[] = { y_1,
396 y_2,
397 y_3,
398 y_4 };
399 y->fill(4, 1, yd);
400 }
401
402 void getRange(range_real * ranges, fish * target, usvs *
        boats) {
403 ranges->rec1 = euclid3(target, &boats->rec1);
404 ranges->rec2 = euclid3(target, &boats->rec2);
405 ranges->rec3 = euclid3(target, &boats->rec3);
406 ranges->rec4 = euclid3(target, &boats->rec4);
407 }
408
409 void getTDOAs(TDOA_data * tdoas, range_real * ranges,
        double soundspeed) {
410 tdoas->TDOA_21 = (ranges->rec2 - ranges->rec1) / soundspeed
        ;
411 tdoas->TDOA_31 = (ranges->rec3 - ranges->rec1) / soundspeed
        ;
412 tdoas->TDOA_41 = (ranges->rec4 - ranges->rec1) / soundspeed
```

```
    ;
}

void createPseudorange(pseudoranges * pseudos, fish * x_hat
    , usvs * boats) {
pseudos->rec1 = euclid3(x_hat, &boats->rec1);
pseudos->rec2 = euclid3(x_hat, &boats->rec2);
pseudos->rec3 = euclid3(x_hat, &boats->rec3);
pseudos->rec4 = euclid3(x_hat, &boats->rec4);
}
```

E MATLAB code

E.1 TDOA localization in $\mathbb{R}^2$

Listing 1:

```
clear;clc;
c = 1484; %[m/s] speed of sound in water
%point locations
A=[0,1];
B = [7,1];
C=[2,5];
D=[5,4];
T = [3,3]; %pretended unknow target.
x_points=  [0,7,2,5,3];
y_points= [1,1,5,4,3];
scatter(x_points,y_points)
hold on
ab = euclid(A(1),A(2),B(1),B(2));
ac = euclid(A(1),A(2),C(1),C(2));
bc = euclid(B(1),B(2),C(1),C(2));
at = euclid(A(1),A(2),T(1),T(2));
bt = euclid(B(1),B(2),T(1),T(2));
ct = euclid(C(1),C(2),T(1),T(2));
dt = euclid(D(1),D(2),T(1),T(2));

%calculating the actual difference of arrival time
    experienced by the receivers. Delta T is the difference
    in distance travelled by the actual signal divided by
    the value of the speed of the signal.
deltaTab = (at-bt) / c;
deltaTac = (at-ct) / c;
deltaTad = (dt-at) / c;

syms x;syms y;
```

```
y_solved_ab =   solve(c*deltaTab ==  sqrt( (x-A(1)).^2 + (y
    -A(2)).^2) - sqrt((x-B(1)).^2+(y-B(2)).^2), y);
fplot(y_solved_ab(1))
hold on
fplot(y_solved_ab(2))
hold on

y_solved_ac =   solve(c*deltaTac ==  sqrt( (x-A(1)).^2 + (y
    -A(2)).^2) - sqrt((x-C(1)).^2+(y-C(2)).^2), y);
fplot(y_solved_ac(1))
hold on
fplot(y_solved_ac(2))
hold on

y_solved_ad =   solve(c*deltaTad == sqrt( (x-A(1)).^2 + (y-
    A(2)).^2) - sqrt((x-D(1)).^2+(y-D(2)).^2), y);
fplot(y_solved_ad(1))
hold on
fplot(y_solved_ad(2))
hold on
legend('Time difference R1 and R2: solution 1','Time
    difference R1 and R2: solution 2','Time difference R1
    and R3: solution 1','Time difference R1 and R3: solution
     2','Time difference R2 and R3: solution 1','Time
    difference R2 and R3: solution 2')
axis([-2,8,-2,8])
```

E.2 TDOA localization in $\mathbb{R}^3$ with 4 receivers

Listing 2:

```
clear;clc; clf;
%% Initialization
c = 1484;
receiver_pos = [0,1,-2;
7,1,0;
2,5,-4;
5,4,-6];
target_pos = [3,3,-3];
range=zeros(4,1);
 for j = 1:4
                  range(j) = euclid3( target_pos,
                      receiver_pos(j,:));
 end
TDOA = zeros(3,j);
range_ref = range(1);
 for j = 2:4
```

```
TDOA(j-1) = (range(j) - range_ref)/c;
 end

%% Calculating true solutions
for j = 2:4
syms x y z;
S = solve(c*TDOA(j-1) == sqrt((x-receiver_pos(j,1))^2+(y-
    receiver_pos(j,2))^2+(z-receiver_pos(j,3))^2) - sqrt((x-
    receiver_pos(1,1))^2+(y-receiver_pos(1,2))^2+(z-
    receiver_pos(1,3))^2), z );
f(x,y)=S(1);
r=1:0.25:5;
[X,Y] = meshgrid(r);
Z = double(f(X,Y));
surf(X,Y,Z,'FaceAlpha',0.5);
zlim([-3,-1]);
hold on
end

%% Plots
scatter3(receiver_pos(:,1),receiver_pos(:,2),receiver_pos
    (:,3),200,'b','filled')
hold on
scatter3(target_pos(1),target_pos(2),target_pos(3),300,'r',
    'filled')
hold on
object_names = {'R1','R2','R3','R4','Target'} ;
cellstring = cellstr(object_names);
for j = 1:4
text(receiver_pos(j,1)-3,receiver_pos(j,2)+2,receiver_pos(j
    ,3), cellstring(j)); %name plotting.
hold on
end
text(target_pos(1)-3,target_pos(2)+2,target_pos(3),
    cellstring(5)); %name plotting.
axis([-2,10,-2,10,-8,2])
xlabel('x')
ylabel('y')
zlabel('z')
```

E.3 TDOA localization in $\mathbb{R}^3$ with 3 receivers and a depth measurement

Listing 3:

```
clear;clc; clf;
```

```
%% Initialization
c = 1484;
receiver_pos = [0,1,-2;
7,1,0;
2,5,-0.5 ;];
target_pos = [3,3,-3];
range=zeros(4,1);
for j = 1:3
        range(j) = euclid3( target_pos, receiver_pos(j,:));
end
TDOA = zeros(3,j);
range_ref = range(1);
for j = 2:3
TDOA(j-1) = (range(j) - range_ref)/c;
end

%% Calculating true solutions
for j = 2:3
syms x y z;
S = solve(c*TDOA(j-1) == sqrt((x-receiver_pos(j,1))^2+(y-
    receiver_pos(j,2))^2+(z-receiver_pos(j,3))^2) - sqrt((x-
    receiver_pos(1,1))^2+(y-receiver_pos(1,2))^2+(z-
    receiver_pos(1,3))^2), z );
f(x,y)=S(1);
r= 1:0.25:5;
[X,Y] = meshgrid(r);
Z = double(f(X,Y));
surf(X,Y,Z,'FaceAlpha',0.5);
hold on
end
[X,Y] = meshgrid(-2:0.3:10,-2:0.3:10);
height = 0.2;
Z = height*sin(X) + height*cos(Y);
C = X*Z*Y;
s= surf(X,Y,Z,'FaceAlpha',0.1);

%% Depth
r= 1:0.25:5;
[X,Y] = meshgrid(r);
Z = ones(size(X))*target_pos(3);
surf(X,Y,Z,'FaceAlpha',0.4);

%% Plots
scatter3(receiver_pos(:,1),receiver_pos(:,2),receiver_pos
    (:,3),200,'b','filled')
```

```
hold on
scatter3(target_pos(1),target_pos(2),target_pos(3),300,'r',
    'filled')
hold on
object_names = {'R1','R2','R3','Target'} ;
cellstring = cellstr(object_names);
dx=-0.;
for j = 1:3
text(receiver_pos(j,1)+dx,receiver_pos(j,2)+dx,receiver_pos
    (j,3)-0.5, cellstring(j)); %name plotting.
hold on
end
text(target_pos(1)-0.5,target_pos(2)-0.5+dx,target_pos(3)
    -0.5, cellstring(4)); %name plotting.
axis([-2,10,-2,10,-8,2])
xlabel('x')
ylabel('y')
zlabel('z')
```

E.4 Single measurement localization in $\mathbb{R}^3$

Listing 4:

```
clear;clc; clf;
%% Initialization
c = 1484;
receiver_pos = [0,0,0];
target_pos = [3,3,-6];
domeRadius = euclid3( target_pos, receiver_pos);
%% Dome plot
[x,y,z] = sphere(20);
xEast  = domeRadius * x;
yNorth = domeRadius * y;
zUp    = domeRadius * z;
zUp(zUp < 0) = 0;
figure('Renderer','opengl')
surf(xEast, yNorth, -zUp,'FaceColor','yellow','FaceAlpha'
    ,0.3)
hold on
[X,Y] = meshgrid(-10:0.3:10,-10:0.3:10);
height = 0.2;
Z = height*sin(X) + height*cos(Y);
s= surf(X,Y,Z,'FaceAlpha',0.1);
%% Disc plot
radius  = sqrt(target_pos(1)^2 + target_pos(2)^2);
[T,R] = meshgrid(linspace(0,2*pi,64),linspace(0,radius,16))
    ;
```

```
23 X = R.*cos(T);
24 Y = R.*sin(T);
25 Z = ones(size(X))*target_pos(3);
26 surf(X,Y,Z)
27 %% Receiver and transmitter plot
28 scatter3(receiver_pos(:,1),receiver_pos(:,2),receiver_pos
       (:,3),400,'b','filled')
29 hold on
30 scatter3(target_pos(1),target_pos(2),target_pos(3),400,'r',
       'filled')
31 hold on
32 object_names = {'R1','Target'} ;
33 cellstring = cellstr(object_names);
34 dx=-0.;
35 text(receiver_pos(1)+1+dx,receiver_pos(2)+dx,receiver_pos
       (3)-2, cellstring(1)); %name plot.
36 hold on
37 text(target_pos(1)+1,target_pos(2)+dx,target_pos(3)+1,
       cellstring(2)); %name plot.
38 hold on
39 %% Red line circle plot
40 theta = 0:pi/50:2*pi;
41 xunit = radius * cos(theta);
42 yunit = radius * sin(theta);
43 h = plot3(xunit, yunit, target_pos(3)*ones(size(theta)),'r'
       );
44 set(h,'linewidth',3);
45 axis([-10,10,-10,10,-10,10])
46 xlabel('x')
47 ylabel('y')
48 zlabel('z')
```