Visualization of Ships in a Mixed-Reality Environment and Automated Situational Awareness using Live AIS Data

Thesis · December 2018

CITATIONS
7

READS
3,415

1 author:

Sindre Fossen
Maritime Robotics AS
7 PUBLICATIONS **104** CITATIONS

SEE PROFILE

Sindre Fossen

Visualization of Ships in a Mixed-Reality Environment and Automated Situational Awareness using Live AIS Data

Master's thesis in Simulation and Visualization

Supervisor: Robin T. Bye and Ottar L. Osen

December 2018

Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of ICT and Natural Sciences

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

Visualization of Ships in a Mixed-Reality Environment and Automated Situational Awareness using Live AIS Data

Sindre Fossen

Master of Science in Simulation and Visualization

12 December 2018

Supervisor: Associate Professor Robin T. Bye
Co-supervisor: Associate Professor Ottar Osen
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of ICT and Natural Sciences

Summary

The aim of this graduation thesis entitled "Visualization of Ships in a Mixed-Reality Environment and Automated Situational Awareness using Live AIS Data" is to develop a framework for an automated situational awareness system for human operators using live AIS measurements. In this context, situational awareness means having an accurate understanding of what is happening around you and what is likely to happen. The main tool for this is 3-D visualization of AIS detected ships and a remotely operated unmanned surface vehicle (USV) in a mixed-reality environment. The chosen environment is the fjord outside the city of Trondheim in Norway, which is implemented in Unity as a virtual world. The 3-D model is based on geographical information system (GIS) data and the AIS detected ships are overlaid the virtual world using their real geographical positions, which are live decoded AIS messages.

The main result of the thesis is an automated situational awareness system, which consists of subsystems for decoding of the AIS messages, state estimation and motion prediction, risk assessment, 3-D visualization and a graphical user interface (GUI).

Several case studies for motion prediction and computation of online risk collision risk indexes are presented to demonstrate the concept of automated situational awareness. The results have been published in two conference papers and one journal paper.

Finally, the author recommends that an automated situational awareness system onboard a commercial ship or an USV combines navigation and sensory data from other sources than AIS. Redundancy is important from a safety point of view. AIS is not considered to be a reliable system due to loss of signals, limited range, slow update rates and the problem that the system can be turned off during operation. Consequently, an industrial system for automated situational awareness should be based on sensor fusion where information from radars, lidars, infrared thermal cameras, optical cameras, satellites etc. are combined in an optimal manner.

In addition to this, it is recommended to develop a 3-D visualization system, which makes it possible for a human operator to analyze a large amount of data in real time without saturating the human cognitive system.

Sammendrag

(Norwegian translation of the summary)

Målet med masteroppgaven som har tittel "Visualization of Ships in a Mixed-Reality Environment and Automated Situational Awareness using Live AIS Data" er å utvikle et rammeverk og et system for automatisk situasjonsforståelse for menneskelige operatører basert på AIS-målinger. I denne sammenhengen betyr automatisk situasjonsforståelse å forstå hva som skjer rundt deg og hva som kommer til å skje. Hovedverktøyet er 3-D visualisering av AIS-detekterte skip og et fjernstyrt ubemannet overflatefartøy (USV) i et miljø basert på blandet virkelighet. Det valgte miljøet er en fjord utenfor Trondheim som er implementert i Unity som en virtuell verden. 3-D modellen er basert på data fra geografiske informasjonssystemer (GIS) og AIS-detekterte skip er lagt oppå den virtuelle verdenen ved å bruke deres virkelige geografiske posisjoner som dekodes fra AIS-meldingene i sann tid.

Hovedresultatet i masteroppgaven er et system for automatisk situasjonsforståelse som består av delsystemer for dekoding av AIS-meldinger, tilstandsestimering, prediksjon av fartøybevegelse, risikovurdering, 3-D visualisering samt grafisk brukergrensesnitt (GUI).

Flere eksempelstudier for prediksjon av fartøybevegelse og beregning av kollisjonsindekser i sann tid er inkludert for å demonstrere konseptet for automatisk situasjonsforståelse. Disse resultatene er også publisert internasjonalt som to konferanseartikler og en tidsskriftartikkel.

Til slutt anbefaler forfatteren at et system for automatisk situasjonsforståelse ombord i et kommersielt skip eller en USV kombinerer navigasjon- og sensordata fra andre kilder enn AIS. Redundans er viktig fra et sikkerhetsperspektiv. AIS blir ikke sett på som et pålitelig system pga. signaltap, begrenset rekkevidde, liten oppdateringsrate og problemet med at systemet kan slåes av under en operasjon. På grunn av dette bør et industrielt system for automatisk situasjonsforståelse baseres på sensorfusjon hvor man kombinerer informasjon fra radar, lidar, infrarøde termiske kamera, optiske kamera, satellitter m.m. på en optimal måte.

I tillegg til dette anbefales det å utvikle et 3-D visualiseringssystem som gjør det mulige for den menneskelige operatøren å analysere store mengder med data i sann tid uten å mette det menneskelige kognitive systemet.

Preface

This monograph is submitted in partial fulfillment of the requirements for the degree of *Master of Science in Simulation and Visualization* at the Norwegian University of Science and Technology (NTNU).

The work presented herein has been completed at the Department of ICT and Natural Sciences, NTNU, in close cooperation with Maritime Robotics AS. My supervisor has been Associate Professor Robin T. Bye and Associate Professor Ottar L. Osen has been co-supervising the project.

Acknowledgments

The completion of this thesis and the work it represents have been one of the most challenging things I have ever done. Yet, it has been an incredible rewarding journey. I have had the fortunate opportunity to work with the engineers in Maritime Robotics AS. They have all been a great inspiration and invaluable when discussing visualization, situational awareness and operation of autonomous vehicles in general. I am particular grateful to Vegard Evjen Hovstein, Arild Hepsø and Kenan Trnka who believed in me and gave me the opportunity to work as summer intern in the period June to August 2018. Working in a company like Maritime Robotics AS is a dream for all engineering students who have studied cybernetics, robotics and computer science. I am also grateful to Rambøll AS and Trondheim municipality for permission to use their 3-D models.

To my supervisors, Associate Professors Robin T. Bye and Ottar L. Osen, thanks for all the discussions, ideas and late night email responses. You have been a great motivators and invaluable as co-authors on one of my papers.

I am grateful to my family; my parents Heidi and Thor Inge, and my sister Lone Moa, who always have supported me during my studies. I am particular grateful to my father, Thor Inge, who besides co-authoring two of my papers has been invaluable when discussing feedback control, programming, situational awareness and cybernetics in general.

Sindre Fossen
12 December 2018

Contents

List of Figures

List of Tables

Acronyms

2-D 2-dimensional

3-D 3-dimensional

AIS Automatic identification system

API Application programming interface

AR Augmented reality

ARPA Automatic radar plotting aid

ASCII American standard code for information interchange

ASV Autonomous surface vehicle

COG Course over ground

COLREGS International regulations for preventing collisions at sea

DCPA Distance to closest point of approach

EKF Extended Kalman filter

FPS Frames per second

GES Global exponential stability

GIS Geographic information system

GNSS Global navigation satellite systems

GUI Graphical user interface

IDL Interface description language

IMO International maritime organization

KF Kalman filter

LTV Linear time-varying model

NMEA National marine electronics association

SAR Search and rescue

SDLC Software development life cycle

SOG Speed over ground

TCP Internet transmission control protocol

TCPA Time to closest point of approach

UDP Internet user datagram protocol

UML Unified modeling language

USV Unmanned surface vehicle

VHF Very-high frequency

VR Virtual reality

WGS World geodetic system

XKF eXogenous Kalman filter

1 Introduction

This chapter describes the background, system overview and research questions of the thesis. The objectives and main contributions are stated at the end of the chapter.

1.1 Background

Unmanned surface vehicles (USV) and autonomous surface vehicles (ASV) are vehicles that operate on the surface of the water (watercraft) without a crew, see Fig. 1.1. Such vehicles are useful for mapping and monitoring, oceanography, survey, search and rescue (SAR) operations to mention some.

An USV can be remotely piloted as opposed to an ASV, which is autonomous. In order to operate an USV remotely, the pilot must be highly skilled and have good situational awareness.

Autonomous ships will also enter the market in a couple of years. Full autonomy is the "holy grail" in shipping since it can reduce costs and eliminate maritime accidents, which mainly are caused by human errors. The vessel "MS YARA Birkeland" (see Fig. 1.2) will be the world's first fully electric and autonomous container ship (Kongsberg, 2018). It is developed by Kongsberg and in 2020 the ship will sail within 12 nautical miles from the coast, between 3 ports in southern Norway.

Figure 1.1: The Maritime Robotics MARINER USV operating in the Trondheim fjord (https://maritimerobotics.com). Reproduced with kind permission of Maritime Robotics AS.

Figure 1.2: The Yara Birkeland, length 79.5 m, width 14.8 m and draught 6 m (https://www.km.kongsberg.com). Reproduced with kind permission of Kongsberg.

1.1.1 Virtual reality, augmented reality and mixed reality

The difference between virtual reality (VR), augmented reality (AR) and mixed reality is often unclear. According to Quora (2018), VR, AR and mixed reality can be classified as:

Virtual Reality immerses users in a fully artificial digital environment. Everything you see is artificial. Hence, the user experience takes place within a simulated environment.

Augmented Reality overlays digital information (objects) on the real-world environment. Moreover, objects that reside in the real-world are "augmented" by computer-generated information.

Mixed Reality not just overlays but anchors virtual objects to the real world. This can be done in two ways:

i) Mixed reality that starts with the real world. Virtual objects are not just overlaid on the real world but they can interact with it. In this case, a user remains in the real-world environment, while digital content is added to it. This form of mixed reality can be considered as an advanced form of AR.

ii) Mixed reality that starts with the virtual world. The digital environment is anchored to and replaces the real world. The digital objects overlap the real ones whereas in conventional VR the virtual environment is not connected to the real world around a user.

Figure 1.3: The Maritime Robotics AS vehicle control station (VCS) for 3-D visualization of ships in a mixed-reality environment.

Mixed reality is therefore a blending of the physical world with the digital world. The term first appeared in Milgram and Kishino (1994) where it was stated: *"The conventionally held view of a VR environment is one in which the participant-observer is totally immersed in, and able to interact with, a completely synthetic world. Such a world may mimic the properties of some real-world environments, either existing or fictional; however, it can also exceed the bounds of physical reality by creating a world in which the physical laws ordinarily governing space, time, mechanics, material properties, etc. no longer hold"*.

Mixed reality is defined in Wikipedia as:

Definition 1.1 (Mixed Reality). *Mixed reality, sometimes referred to as hybrid reality, is the merging of real and virtual worlds to produce new environments and visualizations where physical and digital objects co-exist and interact in real time. Mixed reality takes place not only in the physical world or the virtual world, but is a mix of reality and VR, encompassing both AR and augmented virtuality via immersive technology.*

Visualization of ships in a mixed-reality environment

The thesis addresses the problem of visualization of real ships using sensory information by augmenting the ships to a simulated world, which is generated using GIS data. More specific, real ships detected by AIS are

overlaid a simulated environment of the Trondheim fjord and visualized in real time.

In addition to the real ships, a remotely-piloted USV is visualized while operating close to the ships. The USV can be real (using GNSS position measurements) or the position, speed and course can be simulated using a mathematical model. In both cases, the operator can steer the USV by using a joystick or a keyboard, see Fig. 1.3.

1.1.2 Automated situational awareness

By visualizing real ships in a mixed-reality environment human operators can perform more complex operations. Situational awareness means having an accurate understanding of what is happening around you and what is likely to happen. 3-D visualization of ship operations in real time will improve situational awareness and help the human operator to do the right decisions.

Definition 1.2 (Situational Awareness). *According to MSQ (2018), situational awareness can be defined as:*

i) Having a good perception of your surroundings at all times.

ii) Comprehending what's happening around you.

iii) Predicting how this will affect your ship.

For good situational awareness, it is necessary to be aware of your environment, including:

- Other ships in the area.
- Communications between vessel traffic services and other ships.
- Weather, sea state and depth of water.
- Tide and current.

In addition to this, good situational awareness means:

Having mode awareness—know your ship's configuration, equipment and systems. These systems include autopilot, radar, GNSS, AIS, compass, propulsion and their engaged modes.

Keep spatial orientation—know the geographical position of the ship within the operational location.

Keep a time horizon—manage time for things such as fuel status and always allow time for unplanned events or emergencies.

Definition 1.3 (Automated Situational Awareness). *The human cognitive process of situation awareness is limited to the amount of data and the level of complexity between the data elements (Holsopple et al., 2010). Situation assessment, encompassing automated threat and impact assessment, should assist human analysts by estimating the motion of ships when operating unmanned vehicles in restricted waters such as a fjord.*

The following definition is used for an “AIS ship" in the remaining of the thesis.

Definition 1.4 (AIS ship). *An AIS ship is a full-scale ship, which is visualized in a virtual world using live AIS data.*

Research questions

From Definition 1.3, the following research questions have been proposed:

Q1 How can a human operator be assisted by estimating and predicting future ship positions?

Q2 How can the human operator’s cognitive process of situational awareness be improved by 3-D visualization and simulation of ship motions in a mixed-reality environment?

Q3 Is it possible to compute an online index for ship collision using estimated data to encompass automated threat and impact assessments?

1.2 System Overview

The topic of the thesis is “visualization of ships in a mixed-reality environment and automated situational awareness using live AIS data”. Motivated by the technology changes in maritime transport and the increased use of autonomy and unmanned vehicles in advanced marine operations, the thesis focuses on visualization and automated situational awareness to assist the human operator.

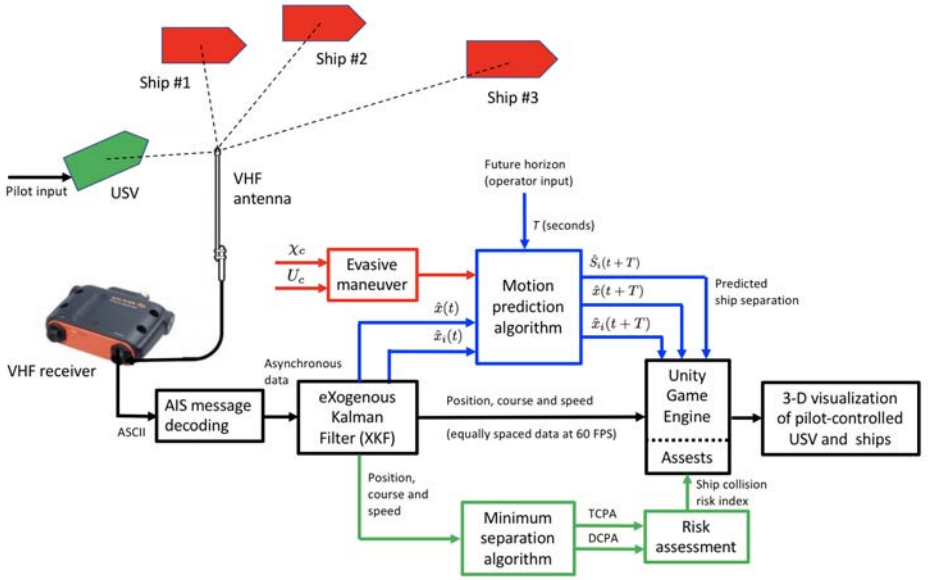


Figure 1.4: System block diagram showing decoding of AIS messages and visualization of the USV and AIS ships using the Unity Game Engine. The XKF interpolates the asynchronous AIS data for smooth visualization at 30-60 FPS. Future motions (T seconds) including evasive maneuvers are computed using a motion prediction algorithm to enhance automated situational awareness. This is further improved by adding algorithms for detection of ship collisions and risk assessment.

The automated situational awareness system in the thesis includes the following features (see Fig. 1.4):

AIS decoder: Parser for decoding of live AIS messages as specified by the National Marine Electronics Association (NMEA, 2018).

State estimator: Two nonlinear Kalman filter algorithms (EKF and XKF) are designed for interpolation of asynchronous AIS measurements at 30-60 Hz. This makes smooth visualization possible. The Kalman filter algorithms also handle TCP and UDP packet losses.

Motion prediction: Prediction of ships and USV motions for a user specified future horizon. USV evasive maneuvers are also included in the motion predictor by using feedback control theory.

Risk assessment by minimum separation between ships: Computation of online minimum separation parametrized in terms of time and dis-

tance to the closest point of approach (TCPA and DCPA). These are the main metrics for computation of the risk index.

Unity Game Engine: Unity is used for 3-D visualization of full-scale ships moving in the Trondheim fjord, Norway. This includes realistic waves and their effect on the ship motions as well as buildings in the Trondheim harbor and the surroundings.

3-D Visualization: Automated situational awareness is visualized by showing the USV and AIS ships in real time using a mixed-reality environment when operating in the Trondheim fjord and harbor. Predicted USV and ship motions, evasive maneuvers and risk index (shown as a predicted colored path) are used to assist the human operator.

1.3 Objectives

The objectives of the work are summarized below:

- 3-D visualization of the Trondheim fjord showing live ship and boat motions using Unity (2018). Wave-induced ship motions should be generated using physical models.
- A remotely-piloted USV should be visualized in real time using a full-scale USV or simulated while operating in the Trondheim fjord. The simulator should use operator inputs and simulate the motions of the USV using a realistic model. This includes interfacing of a joy-stick to the simulator such that the operator can steer the USV manually.
- Live AIS data should be decoded using a parser in C#, which must be interfaced to the Unity. The TCP, alternatively UDP, data stream should be buffered and time stamped such that it presents current positions and speeds of all ships in the vicinity of the USV in real time.
- Future motions of the “AIS ships" should be predicted and visualized using a state estimator. In addition, asynchronous AIS data should be interpolated at 30-60 FPS for smooth visualization results.
- Automated situational awareness and simple concepts for collision detection/warning should be demonstrated using the motion prediction algorithms in a simulated environment.

1.4 Main Contributions and Publications

The following constitutes the main contributions of the thesis:

- A novel automated situational awareness systems for a pilot-operated USV in a mixed-reality environment with live visualization of AIS detected ships to improve the human operators cognitive processes.
- Decoding of live AIS data from a VHF antenna using a parser, which is scripted in C# and implemented in Unity.
- An extended Kalman filter and a globally exponentially stable non-linear observer (eXogenous Kalman filter) for ship tracking, motion prediction and 3-D visualization at 30-60 FPS. Both state estimators handle asynchronous data and TCP/UDP packet losses.
- On-line ship collision risk assessment and visualization of risk for enhanced automated situational awareness using Kalman filters for motion prediction.

The thesis work has resulted in three international publications:

Journal paper

S. Fossen and T. I. Fossen (2018a). eXogenous Kalman Filter (XKF) for Visualization and Motion Prediction of Ships using Live Automatic Identification System (AIS) Data. *Modeling, Identification and Control*, MIC-39(4):233–244, October 2018.

"The first author has written the paper, contributed with the implementation and testing of the XKF in Matlab and developed a parser in C# for live processing of AIS data. The second author has contributed with valuable discussions, proofreading, stability analysis and the XKF algorithm."

Conference papers

S. Fossen, R. T. Bye and O. Osen (2018). Visualization and Collision Risk Assessment of Real Ships in a Mixed Reality Environment using Live Automatic Identification System (AIS) Data. *Proc. of the 2nd European Conference on Electrical Engineering and Computer Science (EECS'18)*, Bern, 20–22 December 2018.

"The first author has written the paper, contributed with the algorithms for risk assessment as well as developed the 3-D visualization software and algorithms. The second and third authors have contributed with valuable discussions, proofreading and discussion regarding the case studies"

S. Fossen and T. I Fossen (2018b). Extended Kalman Filter Design and Motion Prediction of Ships using Live Automatic Identification System (AIS) Data. *Proc. of the 2nd European Conference on Electrical Engineering and Computer Science (EECS'18)*, Bern, 20–22 December 2018.

"The first author has written the paper, contributed with the implementation and testing of the EKF in Matlab and developed a parser in C# for live processing of AIS data. The second author has contributed with valuable discussions, proofreading and algorithms for effective implementation of the EKF in order to handle asynchronous AIS data."

1.5 Organization of the Thesis

The thesis is organized as follows:

Chapter 1 Introduction including system overview, research questions, objectives, main contributions and publications.

Chapter 2 The chapter describes the software requirement specifications, the agile development model and implementation aspects. This includes the Unity game engine, interfacing and parsing of AIS data, GUI, the object manager and 3-D models used for visualization.

Chapter 3 The chapter describes how the future position and course of a ship can be estimated in real time from AIS data using state estimators. The algorithms are experimentally validated by using AIS data from the Trondheim harbor in Norway.

Chapter 4 The chapter demonstrates how automated situational awareness can be visualized. This includes motion prediction, collision detection, evasive maneuvers and online risk assessment.

Chapter 5 concludes the thesis and adds some final remarks on future work.

2 *Software Description*

This chapter describes the software used for simulation and 3-D visualization of live ships in a mixed-reality environment using the Unity Game Engine (Unity, 2018). This includes:

- Structure of the code, specifications and design
- Agile phases for software development
- Interfacing and parsing of AIS data
- 3-D modeling
- Graphical user interface

An overview of the software used in this project is found in Appendix C.

2.1 Code Structure, Requirement Specifications and Design

In this section, the software requirements and development model are presented together with a Unified Modeling Language (UML) diagram for visualization and documentation of the software system.

2.1.1 Software requirement specifications

The software requirement specifications describe how the software system must be developed to comply with the objectives of the project (see Section 1.3):

R1 It must be possible to visualize 100 "AIS ships" (see Definition 1.4) simultaneously in real time using the Unity Game Engine.

R2 It must be possible to compute the position, velocity and course angle by running a Kalman filter for each ship in real time using live AIS data. Future motions of the "AIS ships" should also be predicted by the Kalman filter algorithm.

R3 The AIS ships should be visualized in a mixed-reality environment (see Definition 1.1) using 3-D models of the Trondheim fjord, ships and buildings onshore.

R4 A remotely-piloted USV must be visualized and simulated in real time while operating in the Trondheim fjord. The USV simulator must accept operator inputs such that it can be steered manually and its motion must be computed by using physical models.

R5 Live AIS data must be decoded using a parser and interfaced to the Unity Game Engine by standard Internet protocols.

R6 Asynchronous AIS data must be interpolated up to 60 FPS for satisfactory visualization results.

R7 It must be possible to perform online risk assessment by using motion prediction algorithms.

R8 Automated situational awareness capabilities (see Definitions 1.2 and 1.3) for collision detection/warning must be implemented using motion prediction algorithms in a simulated environment.

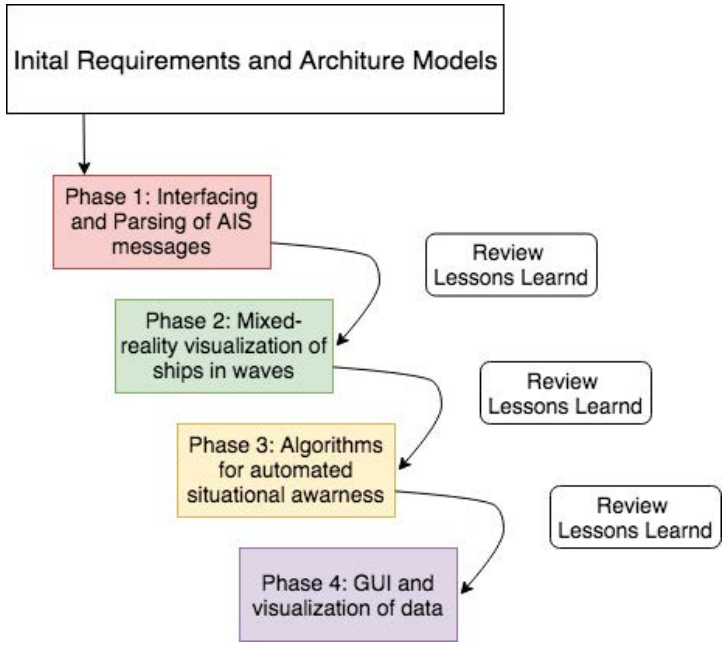


Figure 2.1: Agile model showing the four phases in the project. The software is delivered in increments and requirements can be updated at each step.

Table 2.1: Pros and Cons of different SDLC methods (Balaji and Murugaiyan, 2012).

Method	Pros	Cons
Waterfall	• Requirement is clear before development starts • Each phase is completed in a specified period of time after that it moves to the next phase • Easy to implement • For each phase proper documentation is developed to ensure the quality of the SDLC	• The problems are never solved completely in each phase and in fact many problems arise after the phase is finished. This can result in a badly structured system • If the user wants the requirement to be changed, it cannot be changed at each stage
V-model	• Same as waterfall model • A new advantage is that the Tester role will be a part of the requirement phase • Requirement changes are possible in any phase	• Very rigid and little flexible • If changes happen midway, not only the requirement documents but also the test documentation must be updated • It is not suited for short time projects since reviews are required at each stage
Agile	• An agile model responds to changing requirements of the project • Tailor-made for small projects (adaptive) • There is no guesswork between the developer and the enduser since there is face-to-face communication and continuous inputs from the enduser	• For large project it is difficult to judge the efforts and the time required for the project in the SDLC • Require senior developers to take the decisions necessary for agile type of developments. This limits the use and integration of newbie programmers in the project.

2.1.2 Software development model

When developing the software, a method for planning, analysis, design, and implementation must be chosen. There exists a large number of software development life cycle (SDLC) models such as waterfall, spiral, V-model, rapid prototyping and incremental (Sommerville, 2016). The software project in the thesis is based on the Unity Game Engine and own developed code. The time for development of the code was quite limited.

Commonly used SDLC methods are (Balaji and Murugaiyan, 2012):

Waterfall model Sequential development model where the requirements must be clear before going to the next phase. Testing and documentation are carried out once the code has been fully developed.

V-model The validation and verification model is a modified version of the waterfall model. As opposed to the waterfall model, this model is not sequential. Instead the stages turn back upwards after the coding phase is done. The developmental process relies on the verification from the previous steps before proceeding forward to the next step.

Agile model Agile methods are incremental and they move quickly. An agile method is able to respond to changing requirements (adaptive) even late in the development. Hence, working software is delivered in increments, often in weeks instead of months. The most important feature is customer satisfaction by giving rapid and continuous delivery of useful code.

The Pros and Cons for these methods are summarized in Table 2.1.

Since the software project had to be finished in a few months, extensive models such as the waterfall model and the V-model were less suited. The agile model, however, was optimal since it allowed for iterations and incremental updates of the software. Hence, the agile method was chosen for the development of the software and it was decided to use four phases as shown in Fig. 2.1.

2.1.3 Visualization of the agile model by UML

UML is unified modeling language diagram for visualization and documentation of software systems. As discussed in Section 2.1.2 an agile model was chosen for development of the project. A list of the different classes and there functionality is found in Table 2.2. The color codes for the agile phases in Fig. 2.1 are also used in the UML diagram (see Fig. 2.2). The different phases are chosen as:

Phase 1: The TCP communication protocol was implemented to read the AIS packages from the Norwegian Costal Administration. Subsequently, a parser was implemented to decode the ASCII messages. A UDP communication protocol was also implemented to read messages from the Maritime Robotics AS proprietary Protocol Buffer, witch is used to communicate with the USV at a 6 Hz data rate. The messages contain live AIS data from local ships as well as the USV position, speed and course. The *Communication* class initializes the transmission protocols and forwards the data to the *ObjectManager* class as shown in Fig. 2.2.

Phase 2: The classes *AISVessels*, *RemotelyPilotedUsv* and *SimulatedVessel* were created to visualize the AIS ships, USV and a simulated USV. The simulated vehicle is based on a physical model (see Appendix B.3) and wave-induced heave-roll-pitch responses (see Appendix B.4). The above mentioned classes can access methods in the classes *RollPitch*, *VesselsStaticFucntions* and *CammraController*. The latter classes have methods for latitude and longitude conversion to Cartesian coordinates x and y, heave-roll-pitch wave responses as well as setting the camera to track a manually-controlled USV.

Phase 3: The classes *ShipModels, MotionPrediction* and *ShipRiskIndex* are used by the class *AISVessels* for visualization of the AIS ships, motion prediction based on the EKF algorithm (see Appendix B.1) and online computation and visualization of the collision risk index algorithm (see Section 4.3).

Phase 4: Finally, tools for automated situational awareness were developed by using graphical overlay such as plotting the predicted path of the ships and USV as 2-D colored lines. In addition, optionally circles and boxes around the AIS ships were used to increase visibility, while colors (green-yellow-green) were used to warn for collision using a collision risk index formulae to compute the color scheme.

Table 2.2: Description of C# classes.

.

Class name	**Description**
Communication	Initialize the UDP and TCP threads.
UdpThread	Read UDP messages from the Maritime Robotics USV on a separate thread.
ProtoMsg	Deserialize the Protocol Buffer messages in UdpThread.
TcpThread	Read AIS messages from the VHF antenna.
AisParser	Parse the AIS messages in TcpThread.
ObjectManager	Store the vessel objects in the system RAM. Initialize and add new vessel objects. Transmit parsed and deserialized data from TCP/UDP to vessel object.
RemotelyPilotedUsv	Responsible for the Maritime Robotics 3-D USV model and vessel specific functions such as converting latitude/longitude to xy coordinates, risk index computations, motion prediction, etc.
SimulatedVessel	Simulated USV in Unity. Can be controlled by a pilot using joystick and keyboard.
AisShip	Responsible for the AIS ship 3-D models and vessel specific functions such as converting latitude/longitude to xy coordinates, risk index computations, motion prediction, etc.
RollPitch	Computes AIS ship wave-induced heave, roll and pitch motions for 3-D visualization.
CameraController	Change camera perspective.
VesselsStaticFunction	Common functions for RemotelyPilotedUsv, SimulatedVessel and AisShip.
ShipModels	Load the 3-D model for a given AIS ship.
MotionPrediction	EKF interpolation and motion prediction algorithm.
CollisionRiskIndex	Calculates the risk index, separation between ships and DCPA/TCPA values.
ObjectList	ObjectManager list.
SaveLoadVesselData	Save/load the static AIS messages.
Labels	Shows the highest risk indexes and corresponding DCPA/TCPA values, MMSI numbers, ship types, etc. for ships in the vicinity of the USV to the operator graphically.
Plot2dLine	Plots the colored lines for motion prediction/risk warning on the ocean surface (optionally).
Plot3dBox	Plot a box around the AIS ship to increase visibility (optionally).
Plot2dCircle	Plot a circle around the AIS ship on the ocean surface to indicate the risk safety margin (optionally).

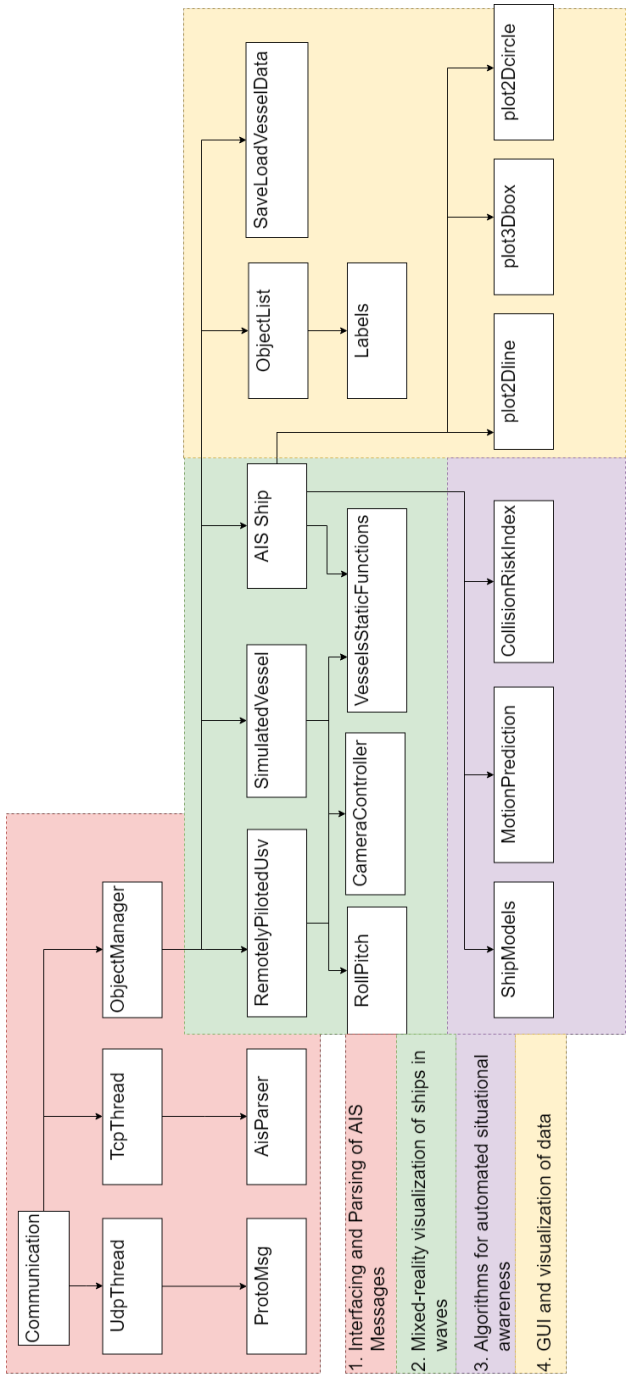


Figure 2.2: UML diagram showing the software system. The color codes corresponds to the four phases in the agile model (see Fig. 2.1).

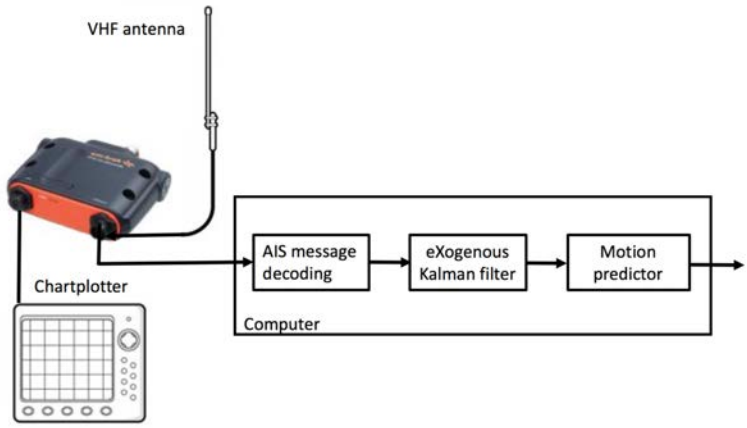


Figure 2.3: AIS receiver and software components for motion prediction.

2.2 Agile Phases

The four agile phases are described in detail below.

2.2.1 Phase 1: Interfacing and parsing of AIS messages

AIS data are transmitted from ships globally and a very-high frequency (VHF) AIS receiver picks up the signals as coded ASCII characters in a format specified by the National Marine Electronics Association (NMEA, 2018). The AIS sentences must be decoded using a parser to obtain real-time ship position, course and speed measurements, see Fig. 2.3.

AIS is a global system, which allows ships to view marine traffic in their area and to be seen by other ships (Automatic Identification System, 2018). Vessels are equipped with a dedicated VHF AIS transceiver, which allows local traffic to be viewed on a chartplotter or computer monitor while transmitting information about the ship itself to other AIS receivers, see Fig. 2.3. Ships can be tracked by AIS base stations located along coast lines or satellites when far away from the base stations. The International Maritime Organization (IMO) requires AIS to be fitted aboard international voyaging ships with 300 or more gross tonnage.

There are 27 AIS messages with different priority that are transmitted using class A and B transceivers (US Coast Guard, 2018). Class B transceivers are smaller, simpler and lower cost than Class A transceivers. For ship tracking and motion prediction the position reports of messages 1, 2, 3 and 18 are particularly useful (see Table 2.4).

Table 2.3: Decoded AIS message.

Message 1,2,3 - Position Report

!AIVDM,1,1,,B,13m=18003v0gPJVTC?6503wd00S4,0*7C

Parm#	Parameter	Value	Description
01	Message ID	1	
02	Repeat indicator	0	No repeat (default)
03	User ID (MMSI)	257114400	
04	Navigational status	0	Under way using engine
05	Rate of turn ROTAIS	+0.0	
06	SOG	25.4	
07	Position accuracy	0	Low (> 10 m) (default)
08	Longitude	10.3779517	
09	Latitude	63.4398267	
10	COG	128.0	
11	True heading	127	
12	Time stamp	54	
13	Special manoeuvre indicator	0	
14	Spare	0	
15	RAIM-flag	0	RAIM not in use (default)
Communication State			
Parm#	**Parameter**	**Value**	**Description**
16	Sync State	0	UTC direct
17	Slot Time-out	0	
18	Slot offset	2244	

Table 2.4: AIS position reports.

AIS Message	Usage	Comments
Message 1, 2, 3: Position Report Class A	Reports navigational information	Longitude and latitude, time, heading, speed, ships navigation status (under power, at anchor...)
Message 18: Position Report Class B	A less detailed report than types 1-3 for vessels using Class B transmitters	Does not include navigation status nor rate of turn

AIS message format

The coded AIS sentences are ASCII characters as defined by the NMEA (2018) format. A typical VHF AIS message looks like:

```
!AIVDM,1,1,,B,13m=18003v0gPJVTC?6503wd00S4,0*7C
```

where

```
!AIVDM   NMEA message type is "received data from other vessels"
1        Number of Sentences (some messages need more then one)
1        Sentence Number (1 unless it is a multi-sentence message)
         Blank is the Sequential Message ID (for multi-sentences)
B        AIS Channel (A or B)
13m=...  Encoded AIS data
0*       End of Data
7C       NMEA Checksum
```

Each ASCII character in the encoded data corresponds to 6 binary bits. Notice that standard ASCII uses 8 bits. The procedure to get 6-bits binary data is to subtract 48 from the ASCII value. If the result is a decimal number larger than 40, subtract 8 and convert to binary. This must be done for each ASCII value in the coded data "13m=18003v0gPJVTC?6503wd00S4". The first characters in the string become:

Char	ASCII	Subtract 48	Subtract 8	6-bit binary
1	49	1	-	000001
3	51	3	-	000011
m	101	53	45	101101
=	61	13	-	001101
...				

The bits can then be converted to ship motion measurements by using the US Coast Guard (2018) specifications. From this it follows that the first 6-bit binary is the AIS message number in the range 1-27. The ship Position Reports are transmitted as Message Numbers 1, 2, 3 and 18. For our example, we recognize the number 1 as Position Report 1.

The remaining bits in the Position Reports, which are needed to decode the motion measurements are:

Position in bit vector	Description
1-6	Message Type
7-8	Repeat Indicator
9-38	userID (MMSI)
39-42	Navigation Satus
43-50	Rate of Turn (ROT)
51-60	Speed Over Ground (SOG)
61-61	Position Accuracy
62-89	Longitude
90-116	Latitude
117-128	Course Over Ground (COG)
129-137	True Heading (HDG)

The MMSI number or Maritime Mobile Service Identity is a 9 digits number, which uniquely identify all ships. The first 3 digits is the country code. There are several on-line services such that the Marine Vessel Traffic webpage: http://www.marinevesseltraffic.com/2013/06/mmsi-number-search.html, which can be used to deduce the ship name and other data from the MMSI number.

AIS message parser

It is necessary to program a NMEA parser in order to decode the AIS messages in real time. Several open source codes are available at GitHub (http://www.github.com. To check if the decoding is successful an on-line decoder such as: http://www.maritec.co.za/tools/aisvdmvdodecoding/ can be used. The result for "!AIVDM,1,1,,B,13m=18003v0gPJVTC?6503wd00S4,0*7C" is shown in Table 2.3.

Table 2.5: Description of Google Protocol Buffer messages.

Message name	Data field	Description
MsgRapid	Latitude, longitude, COG, SOG, etc.	Telemetron position and speed information
MsgObject	MMSI, latitude, longitude, COG, SOG, etc.	AIS object from the USV Telemetron parser
MsgObjectStatic	Name, MMSI, length, beam, etc.	Ship characteristics

Interfacing the USV by Google Protocol Buffers

Protocol Buffers are Google's language-neutral and cross-platform methods of serializing structured data. Serializing is the process of translating data structures or object states into a format that can be stored or transmitted and reconstructed later. The Protocol Buffers are well suited for cross-software communication.

The method involves an interface description language (IDL), which is a specification language used to describe software components application programming interfaces (APIs). IDLs describe an interface in a language-independent way, enabling communication between software components that do not share one language, for instance programs written in C++ and C#. The proprietary software of Maritime Robotics AS is written in C++, while Unity uses C# or Javascript. Maritime Robotics AS also interface their USVs by using Protocol Buffers in order to communicate with the vehicle control station. Consequently, it was decided to implement the Protocol Buffers in Unity to receive data directly from the vehicles.

For the USV Telemetron, three Protocol Buffer messages were extracted from the UDP stream (see Table 2.5). A C# code snippet for deserializing the Protocol Buffer message is shown in Listing 2.1 below.

Listing 2.1: C# pseudocode for deserializing the Protocol Buffer message.

```
1  public class UdpThread
2  {
3      Create UDP socket
4      Start Thread
5      Send Message To server (im alive)
6      Receive receiveBytes From Server
7
8      deserializeMsgObject(receiveBytes);
9
10     public void deserializeMsgObject(byte[] data)
11     {
12         using (var stream = new MemoryStream(data))
13         {
14             m_MsgObject = Serializer.Deserialize<MsgObject>(stream);
15         }
16     }
17 }
```

The *ProtoContract* is defined above the class and indicates that the class will serialize or deserialize. This is a sealed class, which inherits from the attribute class. The C# code snippet for *ProtContract* is shown in Listing 2.2 below.

Listing 2.2: C# ProtoContract snippet for MsgObject.

```
1  [ProtoContract]
2  public class MsgObject
3  {
4      [ProtoMember(1)]                          // Source of object
5      public UInt32 object { get; set; }
6      [ProtoMember(2)]                          // MMSI number
7      public UInt32 uid { get; set; }
8      [ProtoMember(3)]                          // Latitude (rad)
9      public float latitude { get; set; }
10     [ProtoMember(4)]                          // Longitude (rad)
11     public float longitude { get; set; }
12     [ProtoMember(5)]                          // Course over ground (rad)
13     public float cog { get; set; }
14     [ProtoMember(6)]                          // Speed over ground (m/s)
15     public float sog { get; set; }
16     ... etc.
```

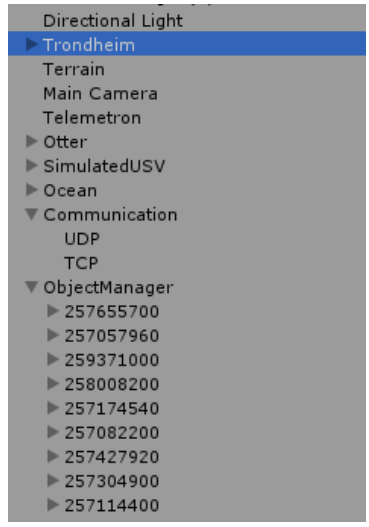
2.2.2 Phase 2: Mixed-reality visualization of ships

Section 2.2.1 explained how to parse live AIS messages using a UDP stream. This is necessary in order to generate real-time position, speed and course angle data for Unity. In this section, the Unity base classes "GameObjects" and "MonoBehaviour" are described. These classes include important features and methods that will be used to visualize the ships in a mixed-reality environment.

Base Class "GameObjects"

"GameObjects" are the fundamental objects in Unity and they represent characters, props and scenery. The "GameObjects" act as containers for "Components", which implement the real functionality. For instance, an AIS ship object is created by attaching a script and a 3-D model to "GameObjects".

Unity allows searching for "GameObjects" by their respective names such as the USVs named Otter, Telemetron, etc. Hence, each time you find "GameObjects" objects you can access the objects variables and class member functions. The discovered vehicles are shown in the list to the right together with the "ObjectManager" numbers, which are MMSI numbers for AIS ships detected in the vicinity of the USV.



A C# code snippet for initializing and updating AIS ships is illustrated in Listing 2.3.

Listing 2.3: C# snippet showing the ObjectManager.

```
1  public void findVessel(Protobuffers.MsgObject m_MsgObject)
2  {
3     if (GameObject.name.Find(m_MsgObject.MMSI) == null)
4          {
5          instantiateVessel(m_MsgObject);
6          }
7      else
8          {
9          updateVessel(m_MsgObject);
10         }
11 }
12
13 private void initsiateVessel(Protobuffers.MsgObject m_MsgObject)
14 {
15     m_vessel = new GameObject();
16     m_vessel.name = m_MsgObject.MMSI;
17     // Add AisShip class properties to the Game object
18     m_vessels = m_vessel.AddComponent <AisShip>();
19     m_vessels.initsiateShip(m_MsgObject);
20     }
21
22 private void updateVessel(Protobuffers.MsgObject m_MsgObject)
23 {
24     m_vessel = GameObject.Find(m_MsgObject.MMSI);
25     AisShipComponent = m_vessel.GetComponent <AisShip>();
26     AisShipComponent.updateVesselData(m_MsgObject);
27 }
```

Base Class "MonoBehaviour"

"MonoBehaviour" is the base class from which every Unity script derives. "MonoBehaviour" also offers life cycle functions that simplify development of object-oriented programs, see Fig. 2.4. Therefore when doing scripting in Unity, the base class helps you systematical controlling your game objects and their behavior.

The C# pseudocode for the AIS ship class is presented in Listing 2.4. On initialization, all selected properties such as motion prediction (see Chapter 3), risk calculation (see Chapter 4), etc. are initialized. In addition, a 3-D model is assigned to the actual AIS ship.

FixedUpdate updates the ship states such as position, velocity and course angle as well as the risk index parameters, while the call *OnRenderObject*, which is responsible for the graphical overlay, runs every frame.

Listing 2.4: C# pseudocode for the AIS ship class.

```
public void Start(Protobuffers.MsgObject m_MsgObject)
{
    m_Vessel.AddComponent<ShipModels>();

    m_Vessel.GetComponent<ShipModels>();

    // loads the 3-D model to the AIS object
    m_ShipModels.loadShipModel(MMSI);

    // add class properties to the AIS ship GameObjects
    m_Vessel.AddComponent<MotionPrediction>();
    m_Vessel.AddComponent<CollisionRiskIndex>();
    m_Vessel.AddComponent<PlotPredictedPath>();
    m_Vessel.AddComponent<RiskCircle>();

    // loads the cube for the inscribed AIS ship
    GameObject.Instantiate(RiskCube);
    }

    void FixedUpdate ()
    {
        MotionPrediction;
        PathPrediction;
        RiskIndex;
        TransformPosition;
    }

    void OnRenderObject()
    {
        DrawLine;
        DrawCube;
        DrawCircle;
    }
```

2.2.3 Phase 3: Algorithms for automated situational awareness

The algorithms for automated situational awareness are based on:

- Ship motion estimation and prediction using Kalman filter algorithms.
- A collision risk index algorithm using time and distance to the closest point of approach.

The Kalman filter algorithms are designed for interpolation of asynchronous AIS measurements at 30-60 FPS. This makes smooth visualization possible. In addition, the Kalman filters are the main algorithms for ship and USV motion prediction. USV evasive maneuvers are programmed as closed-loop feedback control systems, which are represented as state-space models with pilot inputs from a joystick. The state-space models and Kalman filter algorithms are derived in Sections 3.2 and 3.3 and case studies using experimental data documents the performance of the algorithms. Matlab test scripts are found in Appendix B.

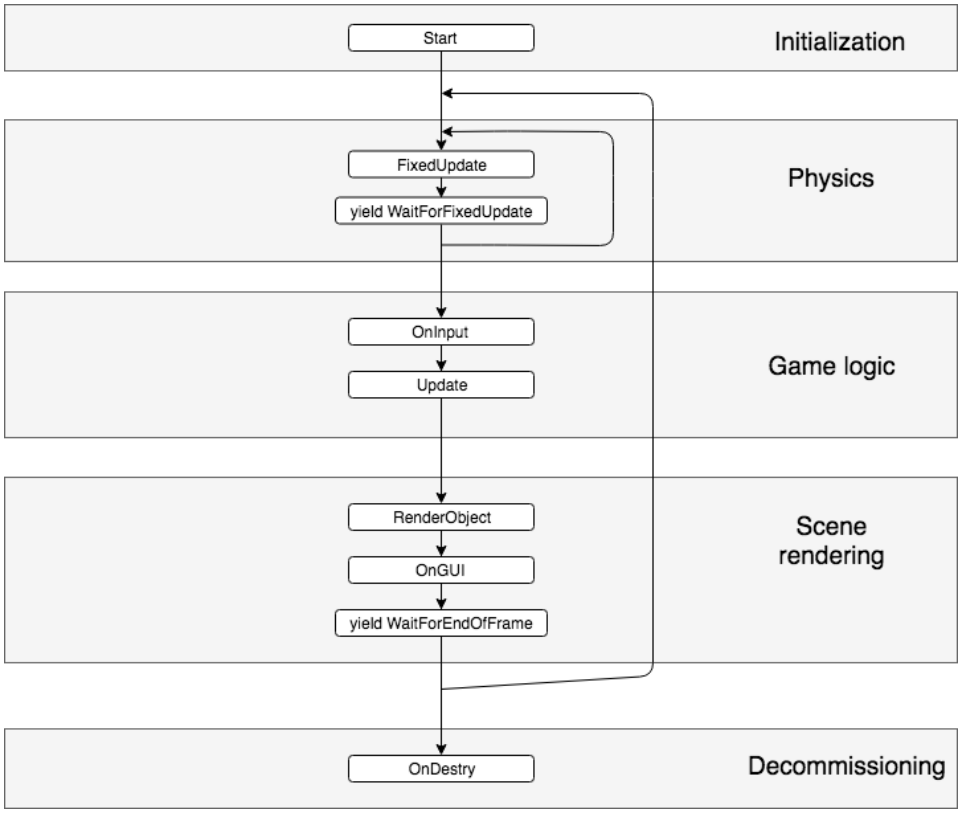


Figure 2.4: AisShip class life cycle. The function *Start* is a constructor for object initialization, *Update* runs as fast as possible, *FixedUpdate* updates at each time sample, while *OnGUI* runs every frame.

The collision risk index algorithm makes use of the TCPA and DCPA values for all AIS ships as outlined in Section 4.3. These are the main metrics for computation of the risk index. The case studies in Section 4.3 use experimental data to demonstrate how online risk assessment can be visualized in a mixed-reality environment.

The open source library *Math.NET Numerics* (https://numerics.mathdotnet.com/#Math-NET-Numerics) was used for numerical implementation of the mathematical formulas in C#.

2.2.4 Phase 4: GUI and visualization of data

When going from a 2-D to a 3-D world, waves and terrain can make ships difficult to spot. Therefore different options for graphical overlaying to increase visibility have been explored. For instance, Unity supports *OpenGL*, which allows for drawing polygons.

Consequently, the following classes have been developed to enhance 3-D visualization of the AIS ships as well as properties for situational awareness:

Plot2dCircle has been constructed to draw a circle around the AIS ships, see Fig. 2.5. The circle is overlaid the ocean surface and the color indicates the risk index. The RGBA values are chosen according to Formula 4.19. The circle coordinates are specified in terms of polar coordinates and the mesh is represented as triangles. Each meshed triangle has its origin in the center of the ship.

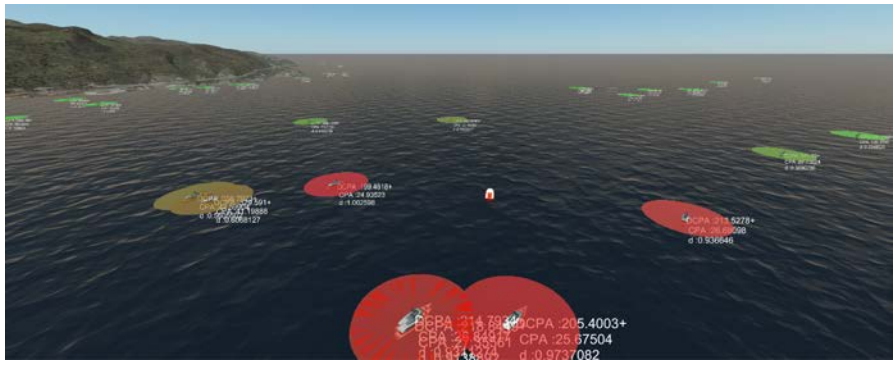


Figure 2.5: 3-D visualization of AIS ships and their respective risk collision indexes (colored circles).

Plot2dLine plots a colored line showing the predicted motion of an AIS ship for a given time in seconds using the Kalman filter algorithms in Chapter 3, see Fig. 2.6. The risk collision index is used to compute the color code as specified by Formula 4.19 and this illustrates the risk for collision in an intuitive way (green-yellow-red). The straight line was constructed using two triangles, while curved paths can be constructed using a large number of boxes where the position is slightly shifted for each box.

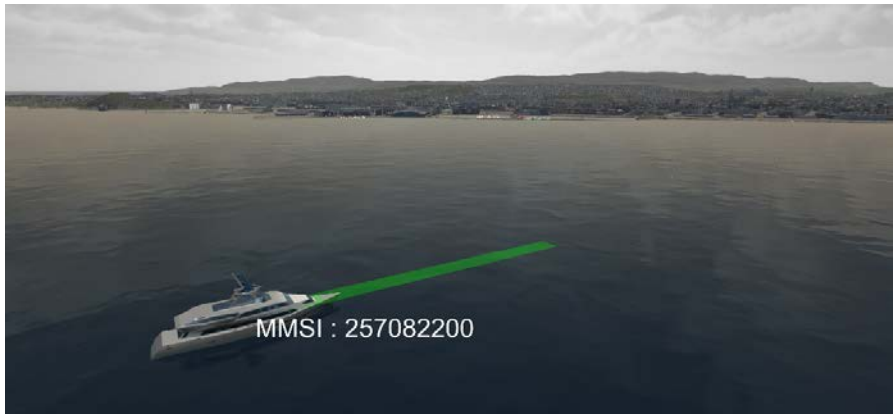


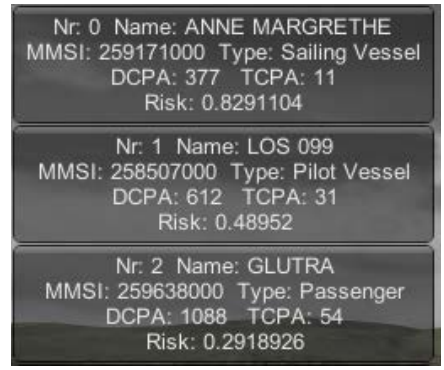
Figure 2.6: 3-D visualization of an AIS ships and a straight line indicating the predicted motion of the ship. The color represent the risk collision index in which green means no risk for collision.

Plot3dBox plots a rectangular prism around the AIS ship to increase visibility. The rectangular prism is constructed by applying texture to a cube such that the AIS ship is inscribed in the prism. The texture is given a color according to the risk collision index as specified by Formula 4.19. When doing this, it was necessary to reduce the alpha value by 30-50 % to increase visibility of the ship inside the rectangular prism, see Fig. 2.7.

Figure 2.7: 3-D visualization of an AIS ship inscribed a rectangular prism. The color represent the risk collision index.

The GUI for the human operator is designed to allow users to interact with the situational awareness system. For this purpose a visual indicator was constructed to show the numerical values for the collision risk index corresponding to the DCPA and TCPA values, MMSI number, etc. for each AIS ship.

The ship name and type are deuced from the AIS messages. This information can be used to visualize a lookalike ship from a database containing 3-D ship models.



The AIS ship risk indexes are sorted and numbered as $0, 1, 2, \cdots$. The ship with the smallest number represents the highest collision risk to the USV, that is *ANNE MARGRETHE*.

2.3 3-D Modeling

In 3-D computer graphics, 3-D modeling is the process of developing a mathematical representation of any surface of an object in three dimensions. The product is called a 3-D model. Unity can import a large number of different 3-D models such as the FPX, OBJ and BLEND file formats.

This section presents the terrain, environment and ship models used for mixed-reality visualization.

2.3.1 Terrain and environmental modeling

For mixed-reality applications it is important that the terrain is as realistic as possible. The digital environment should be anchored to and replace the real world. In this project two different sources for GIS data have been used. Google Earth has been used for terrestrial models, while high-resolution data from Rambøll AS and Trondheim municipality have been used to model Trondheim city including the buildings, the river Nidelven as well as the Trondheim fjord.

Ocean wave generation

To enhance the graphical user experience, realistic waves were implemented in the virtual world. For this purpose the *Hydroform Ocean System* plugin from the *Unity Asset Store* was used for generation of ocean waves, see Fig. 2.8. The plugin allows the user to specify wave amplitude, frequency, speed, wave pattern, etc. The wave parameters can be modified during simulation.

Figure 2.8: Ocean waves generated using the *Hydroform Ocean System* plugin from the *Unity Asset Store*.

Terrain generation

The *TerraLand* plugin from *Terraunity* contains multiple components for GIS data to load and create photo-realistic terrain from any part of the Earth. A realistic terrain is created in two steps. First, elevation data are used to generate a 3-D model based on the heightmap data of the terrain. Second, images from the *Google satellite* are used as textures to make the terrain as realistic as possible. An example showing a region of the Trondheim fjord is shown in Fig. 2.9.

In 2014, the Trondheim municipality developed a digital 3-D urban model for an area of approximately 36 km^2 in and around the city center. The Trondheim 3-D city model is a digital reproduction of the Trondheim municipality in the form of a larger 3-D model based on information from map data, laser data and orthophotos from 2014 as well as oblique images from 2013 and street images from 2014.

The Trondheim model consists of terrain data based on laser and map data, buildings (textured from images) and other historic objects, see Fig. 2.10.

Figure 2.9: A region of the Trondheim fjord generated using the *TerraLand* plugin from the *Terraunity*.

Figure 2.10: The Trondheim 3-D city model. Reproduced with kind permission of Rambøll AS and the Trondheim municipality.

Mesh cutting

When producing models from GIS data it is necessary to cut away the ocean surface because it needs to be replaced by a dynamic 3-D model of the ocean, see Fig. 2.11. With the *Mesh Cutter* from *Unity Asset Store* you can select different parts of the mesh and cut, copy, paste or remove them. The mesh cutter preserves the materials and submeshes of the large mesh and create a new mesh and prefab whenever you cut or copy a part. This transfers the materials and submeshes to the new object.

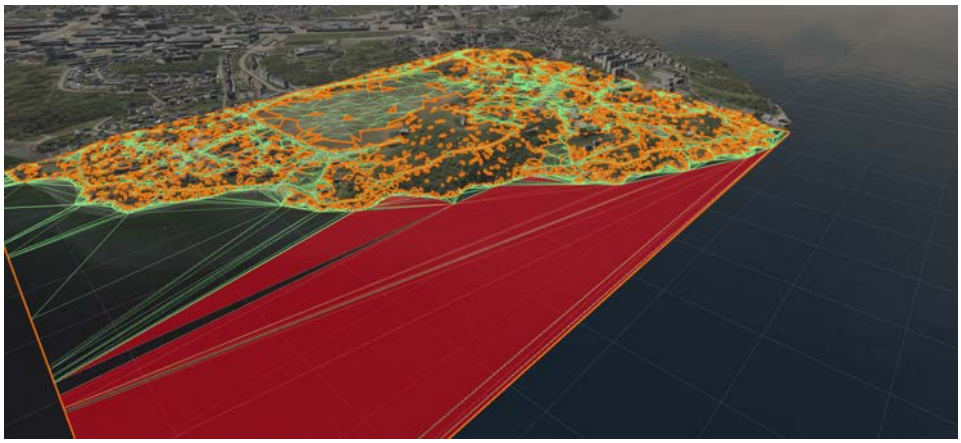


Figure 2.11: The flat ocean is removed from the GIS model by using the *Mesh Cutter* plugin from the *Unity Asset Store*. The polygons in red represent the deleted mesh.

2.3.2 Ship and USV modeling

To make the mixed-reality experience as realistic as possible for a human operator, it is important to have good 3-D models of the AIS ships and USVs. The AIS messages contain information about ship type and size. The ship MMSI number can also be used to find detailed information about the ship. Moreover, the MMSI number can be used to choose a lookalike ship. For instance, a ship with the AIS information, length 80 m, beam 15 m and name tag ferry can be mapped to a scaled ferry in the 3-D world. This of course requires a database of 3-D models, which can be sorted using ship types and main dimensions.

3-D models of AIS ships and USVs can, however, be obtained from several commercial companies or even developed in-house. In this project the following models have been used:

Unity Asset Store models: Library of free and commercial assets, which includes 3-D ship models (see Fig. 2.12).

Figure 2.12: Unity Asset Store ship models.

Blender models: Open-source 3-D computer graphics software toolset, which allows the user to create 3-D models for use in Unity (see Fig. 2.13).

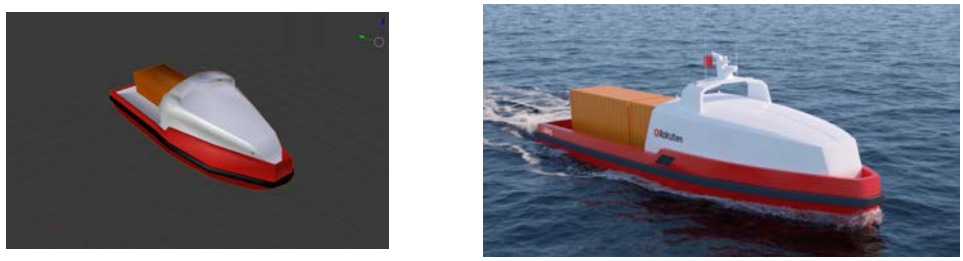


Figure 2.13: The blender model to the left is developed by using the USV picture shown to the right. Reproduced with kind permission of Maritime Robotics AS.

Sketchup models: 3-D Warehouse is an open library in which SketchUp users may upload and download 3-D models to share.

Proprietary models: Proprietary USV models as the Maritime Robotics AS model in Fig. 2.14, which is is small USV for surveillance and data acquisition.

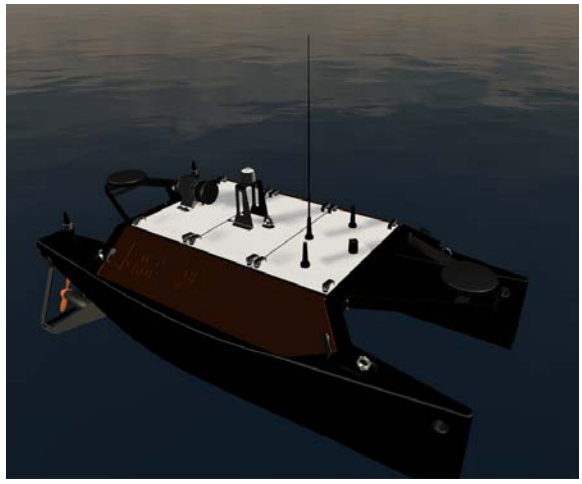


Figure 2.14: The Maritime Robotics OTTER USV. Reproduced with kind permission of Maritime Robotics AS.

2.4 Concluding Remarks

This chapter has given an overview of the software and 3-D models used in the project. An agile model approach has been used for SDLC. Since the software project had to be finished in a few months, the agile model was found to be optimal since it allowed for iterations and incremental updates of the software.

In addition to this, the structure of the code, specifications and design have been presented. This includes interfacing and parsing of AIS data using standard Internet protocols, description of the Unity visualization software, graphical user interface and 3-D models for AIS ships and USVs.

3

Ship Motion Estimation using Live Automatic Identification System Data

This chapter addresses the problem of ship motion estimation using live data from AIS. The main results have been published in the following papers:

S. Fossen and T. I. Fossen (2018a). eXogenous Kalman Filter (XKF) for Visualization and Motion Prediction of Ships using Live Automatic Identification System (AIS) Data. *Modeling, Identification and Control (MIC)*. MIC-39(4):233–244, October 2018.

S. Fossen and T. I Fossen (2018b). Extended Kalman Filter Design and Motion Prediction of Ships using Live Automatic Identification System (AIS) Data. *Proc. of the 2nd European Conference on Electrical Engineering and Computer Science (EECS'18)*, Bern, 20–22 December 2018.

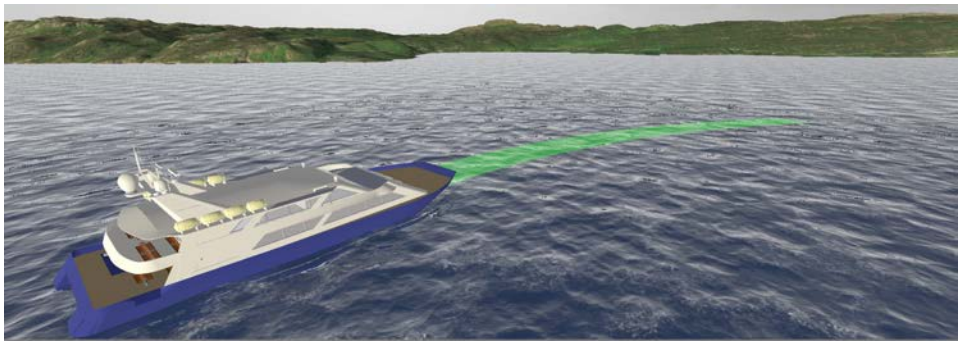


Figure 3.1: The green arrow shows the predicted path of a catamaran passenger boat operating in the Trondheim fjord using the Unity game engine (Unity, 2018), and the Hydroform Ocean System and Terraland plugins from the Unity Asset Store.

This chapter shows how the EKF and XKF algorithms can be used in ship motion estimation and prediction (see Fig. 3.1). Some benefits are:

- Since the AIS data are transmitted using the TCP and UDP Internet protocols, the state estimator can be designed to handle delayed measurements, asynchronous communication as well as loss of packets. Both the EKF and XKF run at a fixed time step and the estimates will be evenly-spaced data.
- The human eye requires 30-60 FPS to make pictures appear as a smooth film. Hence, for a pilot operating a USV using game technology and 3-D visualization, the position and course of the AIS ships should be estimated at 30-60 Hz to satisfy the human constraints.
- A state estimator makes it possible to predict future ship motions, at least for a limited time horizon. This is very useful in a simulation and visualization environment

where the operator would like to observe actual and future movements of ships in his operational space.

- A state estimator will improve the accuracy of the AIS data by removing measurement noise, wildpoints and corrupted data.

3.1 Ship Dynamics and Measurements

This section presents the equations of motion needed for ship motion estimation. The proposed model is intended for short-time prediction (typically less than 1 minute) as opposed to long-time motion prediction, which makes use of statistical data.

3.1.1 Ship model

Let the North-East positions and course angle be denoted (x, y) and χ, respectively. Consequently, a ship moving at forward speed U and course χ is given by (Fossen, 2011):

$$\dot{x} = U\cos(\chi) \tag{3.1}$$

$$\dot{y} = U\sin(\chi) \tag{3.2}$$

$$\dot{U} = a \tag{3.3}$$

$$\dot{\chi} = r \tag{3.4}$$

The linear acceleration a and course rate r are ship dependent and unknown.

For slow AIS measurements, i.e. update frequencies less than 0.5 Hz, it is recommended to choose the inputs as $a = r = 0$. AIS measurements faster than 0.5 Hz can be used to compute estimates r_c and a_c of a and r, respectively by numerical differentiation. The numerically differentiated data are filtered in order to avoid rapid changes. Moreover,

$$\dot{a} = \frac{1}{T_a}\left(\text{sat}(a_c) - a\right) \tag{3.5}$$

$$\dot{r} = \frac{1}{T_r}\left(\text{sat}(r_c) - r\right) \tag{3.6}$$

where T_a and T_r are user defined time constants. The saturation function $\text{sat}(x)$ ensures that $|x| \leq x_{\max}$, which adds robustness to AIS wildpoints.

Let the last three AIS speed and course measurements at times $(t_m,\ t_{m-1}, t_{m-2})$ be denoted $(U_m^{\text{AIS}},\ U_{m-1}^{\text{AIS}}, U_{m-2}^{\text{AIS}})$ and $(\chi_m^{\text{AIS}},\ \chi_{m-1}^{\text{AIS}}, \chi_{m-2}^{\text{AIS}})$, respectively. Furthermore, let $h_1 = t_m - t_{m-1}$ and $h_2 = t_{m-1} - t_{m-2}$. Unfortunately, the data is not evenly spaced, that is $h_1 \neq h_2$. To deal with this problem, a *backward difference approximation* of the first derivative for asynchronous data is derived (see Appendix A.1 for details):

$$a_c = \frac{(1-\alpha)U_m^{\text{AIS}} + \alpha U_{m-1}^{\text{AIS}} - U_{m-2}^{\text{AIS}}}{(1-\alpha)h_1 + h_2} \tag{3.7}$$

$$r_c = \frac{(1-\alpha)\chi_m^{\text{AIS}} + \alpha \chi_{m-1}^{\text{AIS}} - \chi_{m-2}^{\text{AIS}}}{(1-\alpha)h_1 + h_2} \tag{3.8}$$

where

$$\alpha = \frac{(h_1 + h_2)^2}{h_1^2} \tag{3.9}$$

3.1.2 AIS measurements

The AIS data are transmitted from ships globally and a VHF receiver is used to pick up the signals, which appears as coded ASCII characters in a format specified by the National Marine Electronics Association (NMEA, 2018). The AIS sentences are decoded using a parser (see Section 2.2.1) to obtain real-time ship position, course and speed measurements.

The following AIS measurements are used by the state estimator:

- SOG and COG corresponding to the states U and χ.
- Longitude l and latitude μ.

Longitude and latitude can be mapped to Cartesian coordinates using the World Geodetic System (WGS-84), which is the standard coordinate system for the Earth.

For local navigation and visualization it is convenient to use a flat Earth approximation based on WGS-84. The procedure is outlined below (Farrell, 2008).

3.1.3 North-East positions from longitude and latitude

Assume that the flat Earth coordinate origin is located at longitude and latitude (l_0, μ_0) and define:

$$\Delta l := l - l_0 \tag{3.10}$$

$$\Delta \mu := \mu - \mu_0 \tag{3.11}$$

The Earth radius of curvature R_N in the prime vertical and the radius of curvature R_M in the meridian are (Farrell, 2008):

$$R_N = \frac{a}{\sqrt{1 - e^2 \sin^2(\mu_0)}} \tag{3.12}$$

$$R_M = R_N \frac{1 - e^2}{\sqrt{1 - e^2 \sin^2(\mu_0)}} \tag{3.13}$$

where $a = 6\,378\,137$ m is the semi-minor axis (equatorial radius) and $e = 0.0818$ is the Earth eccentricity (WGS-84). Small changes in the North and East positions (x, y) are computed as:

$$x = \frac{\Delta \mu}{\text{atan2}(1, R_M)} \tag{3.14}$$

$$y = \frac{\Delta l}{\text{atan2}(1, R_N \cos(\mu_0))} \tag{3.15}$$

where atan2(y,x) is the 4-quadrant inverse tangent confining the result to $(-\pi, \pi]$.

3.1.4 Course angle from North-East positions

The AIS course angle measurement is often unreliable. In these cases, it is recommended to compute the course angle from the North-East positions $(x(k-1), y(k-1))$ and $(x(k), y(k))$ at times t_{k-1} and t_k, respectively, using the path-tangential angle:

$$\chi(k) \approx \text{atan2}\left(y(k) - y(k-1), x(k) - x(k-1)\right) \tag{3.16}$$

3.2 Extended Kalman filter for AIS Data

Live AIS data has been used for state estimation by Jaskolski (2017) who applied a linear discrete-time Kalman filter. Our approach differs from this work in that a nonlinear kinematic model is used to describe the ship motions. Another difference is how the ship's acceleration and yaw rate are estimated. Since the system model is nonlinear, an EKF is used for state estimation and prediction (Gelb, 1974).

Ship motion prediction has been addressed by numerous authors. The most popular approaches are time-series prediction using the Support Vector Machine (Sapankevych and Sankar, 2009) (Fu et al., 2010) (Jiang et al., 2013), Kalman filtering (Triantafyllou and Bodson, 1982) (Perera and Soares, 2010), on-line neural networks (Yin and Zou, 2011) (Yin et al., 2013) and autoregressive models (Yumori, 1981) (Lin et al., 2011).

Statistical methods have been applied to AIS data for maritime traffic probabilistic forecasting (Xiao et al., 2017), analysis of motion patterns (Ristic et al., 2008), and position prediction using historical AIS data (Mazzarella et al., 2015).

3.2.1 Extended Kalman filter

The nonlinear system model (3.1)–(3.6) can be expressed as:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{B}\mathbf{u} + \mathbf{w} \tag{3.17}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{e} \tag{3.18}$$

where $\mathbf{x} = [x, y, U, \chi]^\top$, $\mathbf{u} = [a, r]^\top$, $\mathbf{w} \in \mathbb{R}^4$ and $\mathbf{e} \in \mathbb{R}^4$ are Gaussian process and measurement noise, respectively,

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} x_3 \cos(x_4) \\ x_3 \sin(x_4) \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.19}$$

and $\mathbf{C} = \mathbf{I}_4$. The EKF makes use of the linearized expression:

$$\mathbf{A} = \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} 0 & 0 & \cos(x_4) & -x_3 \sin(x_4) \\ 0 & 0 & \sin(x_4) & x_3 \cos(x_4) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.20}$$

The AIS data are transmitted using the TCP or UDP Internet protocols. Hence, the EKF must handle delayed measurements, asynchronous communication as well as loss of packets. The EKF runs at a fixed time step and the estimates are evenly-spaced data. Since, the AIS data are transmitted at asynchronous time samples, the EKF is implemented in discrete time using the *predictor-corrector* representation shown in Table 3.1 (Gelb, 1974).

The EKF is initialized with $\hat{\mathbf{x}}^-(0) = \mathbf{x}(0)$ and $\hat{\mathbf{P}}^-(0) = \mathbf{P}(0)$, while $\mathbf{Q} = \mathbf{Q}^\top > 0$ and $\mathbf{R} = \mathbf{R}^\top > 0$ are the process covariance and measurement matrices, respectively. The state transition matrix is computed as:

$$\mathbf{\Phi}(k) = \mathbf{I}_4 + h\mathbf{A} + \frac{1}{2}h^2\mathbf{A}^2 + , \ldots, + \frac{1}{n!}h^n\mathbf{A}^n \tag{3.25}$$

Table 3.1: Four-states extended Kalman filter for AIS data.

Kalman gain:

$$\mathbf{K}(k) = \hat{\mathbf{P}}^{-}(k)\mathbf{C}^{\top}\left(\mathbf{C}\hat{\mathbf{P}}^{-}(k)\mathbf{C}^{\top} + \mathbf{R}\right)^{-1} \tag{3.21}$$

Corrector:

$$\hat{\mathbf{x}}^{+}(k) = \hat{\mathbf{x}}^{-}(k) + \mathbf{K}(k)\left(\mathbf{y}(k) - \mathbf{C}\hat{\mathbf{x}}^{-}(k)\right) \tag{3.22}$$

$$\hat{\mathbf{P}}^{+}(k) = (\mathbf{I}_4 - \mathbf{K}(k)\mathbf{C})\hat{\mathbf{P}}^{-}(k)\left(\mathbf{I}_4 - \mathbf{K}(k)\mathbf{C}\right)^{\top} + \mathbf{K}(k)\mathbf{R}\mathbf{K}^{\top}(k)$$

Predictor:

$$\hat{\mathbf{x}}^{-}(k+1) = \hat{\mathbf{x}}^{+}(k) + h\mathbf{f}(\hat{\mathbf{x}}^{+}(k)) + h\mathbf{B}\mathbf{u}(k) \tag{3.23}$$

$$\hat{\mathbf{P}}^{-}(k+1) = \mathbf{\Phi}(k)\hat{\mathbf{P}}^{+}(k)\mathbf{\Phi}^{\top}(k) + \mathbf{Q} \tag{3.24}$$

where h is the sampling time.

Let $\varepsilon(k) = \mathbf{y}(k) - \mathbf{C}\hat{\mathbf{x}}^{-}(k)$ denote the estimation error in (3.22). When implementing the corrector, it is necessary to map the course angle estimation error:

$$\varepsilon_4(k) = \chi(k) - \hat{\chi}^{-}(k) \tag{3.26}$$

to the interval $[-\pi, \pi)$. This is referred to as the *smallest signed angle*, which can be computed using the function $\text{ssa}(\varepsilon_4)$, which is defined as (MSS, 2004):

$$\text{ssa}(x) := \text{mod}(x + \pi, 2\pi) - \pi \tag{3.27}$$

3.2.2 Ship motion predictor

The ship model (3.1)–(3.6) and the EKF can be used to predict the ship motions from the last AIS measurement at time t_0 to $t = t_0 + T$ where T is the final time, see Fig. 1.4. Let h be the sampling time. Hence, the discrete-time predictor for the interceptor (USV) using Euler's method becomes:

USV:

$$x(k+1) = x(k) + hU(k)\cos(\chi(k)) \tag{3.28}$$

$$y(k+1) = y(k) + hU(k)\sin(\chi(k)) \tag{3.29}$$

$$U(k+1) = U(k) + ha(k) \tag{3.30}$$

$$\chi(k+1) = \chi(k) + hr(k) \tag{3.31}$$

$$a(k+1) = a(k) + \frac{h}{T_a}\left(\text{sat}(a_c) - a(k)\right) \tag{3.32}$$

$$r(k+1) = r(k) + \frac{h}{T_r}\left(\text{sat}(r_c) - r(k)\right) \tag{3.33}$$

where $x(0) = x(t_0)$, $y(0) = y(t_0)$, $U(0) = U(t_0)$, $\chi(0) = \chi(t_0)$, $a(0) = a(t_0)$ and $r(0) = r(t_0)$.

The AIS ship motion predictors for N ships are obtained in a similar manner ($i = 1, 2, ..., N$):

AIS ship #i:

$$x_i(k+1) = x_i(k) + hU_i(k)\cos(\chi_i(k)) \tag{3.34}$$

$$y_i(k+1) = y_i(k) + hU_i(k)\sin(\chi_i(k)) \tag{3.35}$$

$$U_i(k+1) = U_i(k) + ha_i(k) \tag{3.36}$$

$$\chi_i(k+1) = \chi_i(k) + hr_i(k) \tag{3.37}$$

$$a_i(k+1) = a_i(k) + \frac{h}{T_a}\left(\text{sat}(a_{c_i}) - a_i(k)\right) \tag{3.38}$$

$$r_i(k+1) = r_i(k) + \frac{h}{T_r}\left(\text{sat}(r_{c_i}) - r_i(k)\right) \tag{3.39}$$

where $x_i(0) = x_i(t_0), y_i(0) = y_i(t_0), U_i(0) = U_i(t_0), \chi_i(0) = \chi_i(t_0), a_i(0) = a_i(t_0)$ and $r_i(0) = r_i(t_0)$.

3.2.3 Experimental validation

AIS data for MS Trondheimsfjord II (see Fig. 3.2) are used to demonstrate ship motion prediction. The asynchronous AIS data are processed by the EKF to obtain equally-spaced data at 50 Hz. Fig. 3.3 shows the path when crossing the Trondheim fjord and Fig. 3.4 shows the corresponding state estimates.

Figure 3.2: The MS Trondheimsjord II is high-speed catamaran for passenger transport. Length 24.5 m, beam 8.0 m and maximum speed 16.5 m/s (32 knots). Reproduced with kind permission of Fosen Namsos Sjø (http://www.fosennamsos.no).

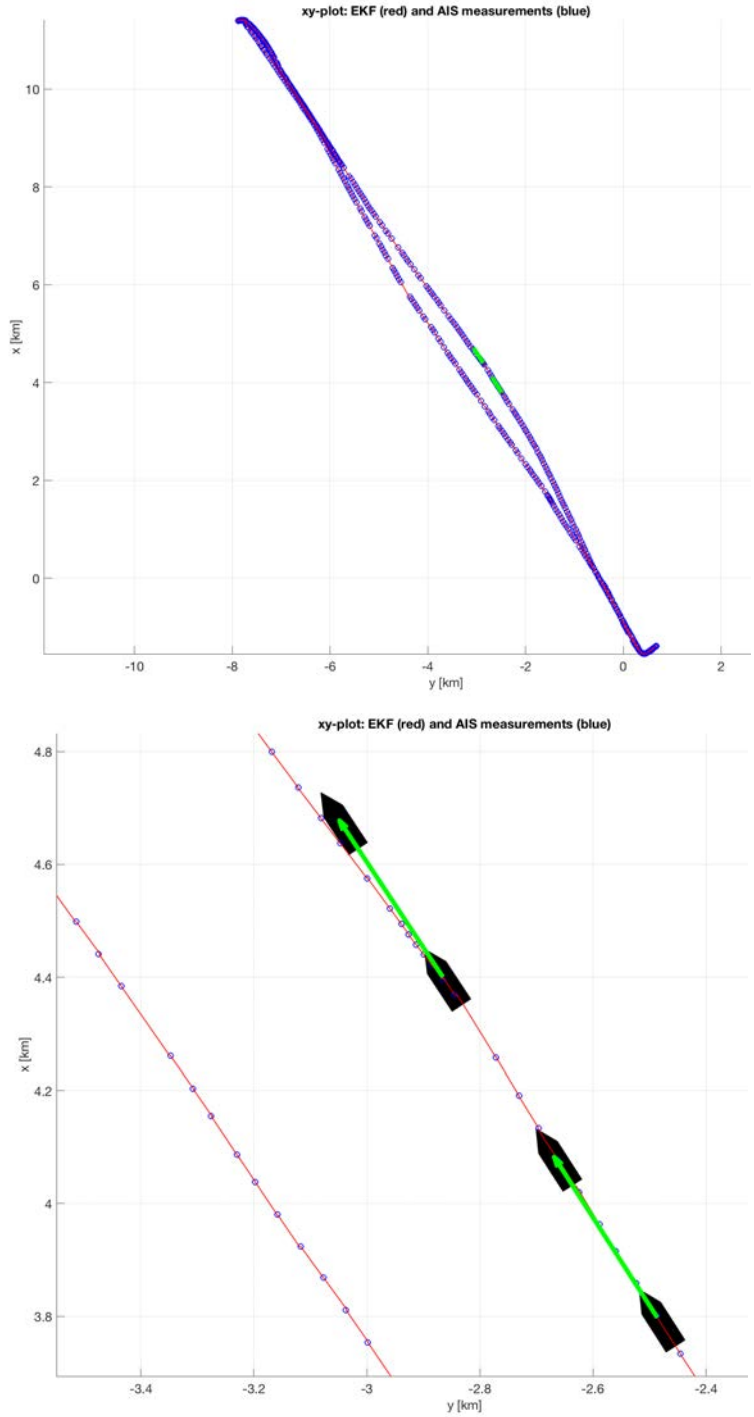


Figure 3.3: The upper plot shows the path of MS Trondheimfjord II when crossing the fjord. The lower zoomed plot shows the predicted ship motions at two different locations (black ships) for a future horizon of 30 seconds (green arrows).

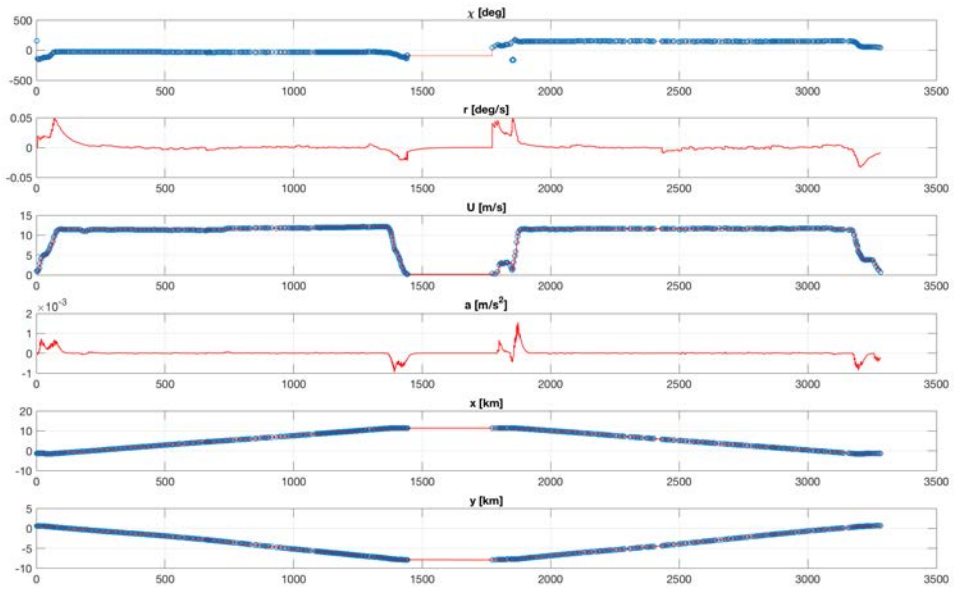


Figure 3.4: Estimated states (red) and AIS measurements (blue circles) as a function of time when crossing the fjord back and forth.

The EKF was implemented and tested in Matlab (see Appendix B.1) before it was coded in C# for Unity integration. The Kalman filter was initialized using:

$$\hat{\mathbf{P}}^{-}(0) = 0.1 \cdot \mathbf{I}_4 \tag{3.40}$$

$$\mathbf{Q} = \text{diag}\,(0.01, 0.01, 0.1, 0.1) \tag{3.41}$$

$$\mathbf{R} = \text{diag}\,(0.001, 0.001, 0.001, 0.01) \tag{3.42}$$

The initial states were chosen as:

$$\hat{\mathbf{x}}^{-}(0) = [x(0), y(0), U(0), \chi(0)]^{\top} \tag{3.43}$$

Motion prediction

The ship motions are predicted at two different locations with a 30 seconds horizon using (3.28)–(3.33). Fig. 3.5 shows how the predicted path can be visualized in a situational awareness system using the Unity game engine.

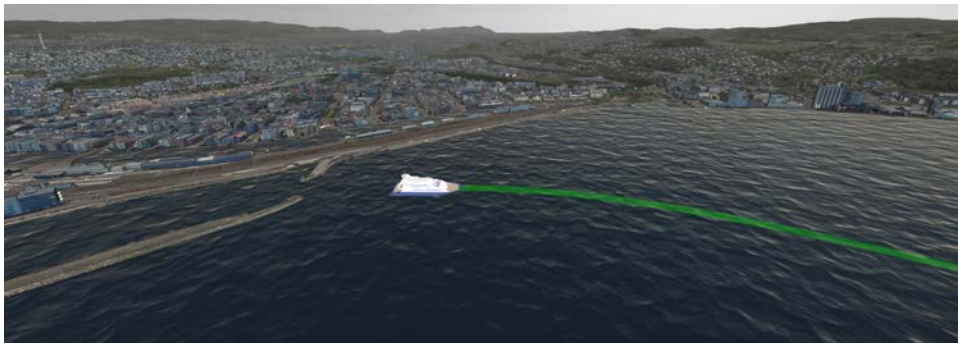


Figure 3.5: 3D-Visualization of a catamaran passenger boat in the Trondheim fjord using the Unity game engine Unity (2018), and the Hydroform Ocean System and Terraland plugins from the Unity Asset Store. The green arrow shows the predicted path of the vessel.

.

3.3 eXogenous Kalman filter for AIS Data

The main contribution of this section is the design of a globally exponentially stable (GES) observer for live AIS data as illustrated in Fig. 2.3 using the XKF (Johansen and Fossen, 2017). GES is an important property since it guarantees exponential convergence to zero of the estimation errors. Moreover, the estimation error equilibrium point $\tilde{\mathbf{x}} = \mathbf{x} - \hat{\mathbf{x}}$ is GES, if there exist positive constants K and α such that (Khalil, 2014):

$$\|\tilde{\mathbf{x}}(t)\| \leq K \exp(-\alpha t) \|\tilde{\mathbf{x}}(0)\|, \quad \forall t \geq 0 \tag{3.44}$$

The GES property guarantees that the observer is robust to large uniformly bounded disturbances.

It is well known that the linear KF is GES and optimal in the sense of minimum variance under some conditions. However, its nonlinear extension, known as the EKF, linearizes the system about the estimated state trajectories. Unfortunately, this can lead to stability problems. Johansen and Fossen (2017) propose to use a cascade of two observers where the first observer generates a globally convergent state estimate, which can be used to design a linearized KF. This approach is referred to as the XKF and it is illustrated in Fig. 3.6.

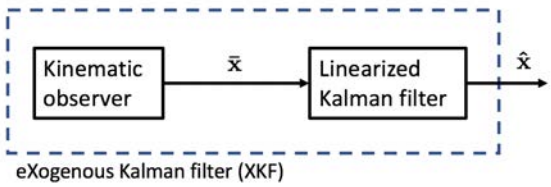


Figure 3.6: The eXogenous Kalman filter as a two-stage estimator. The kinematic observer produces an estimate $\bar{\mathbf{x}}$ used in a linearized KF, which returns the improved estimate $\hat{\mathbf{x}}$.

The first stage is to construct a deterministic observer with global stability properties. This is done without optimality considerations and by neglecting the stochastic noise on

the signals. The estimate, $\bar{\mathbf{x}}$, from the kinematic observer is an exogenous signal used in a linearized KF:

$$\dot{\hat{\mathbf{x}}} = \mathbf{f}(\bar{\mathbf{x}}, t) + \mathbf{F}(\bar{\mathbf{x}}, t)(\hat{\mathbf{x}} - \bar{\mathbf{x}}) + \mathbf{K}\left(\mathbf{y} - \mathbf{h}(\bar{\mathbf{x}}, t) - \mathbf{H}(\bar{\mathbf{x}}, t)(\hat{\mathbf{x}} - \bar{\mathbf{x}})\right) \tag{3.45}$$

where

$$\mathbf{F}(\bar{\mathbf{x}}, t) = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\bar{\mathbf{x}}, t), \quad \mathbf{H}(\bar{\mathbf{x}}, t) = \frac{\partial \mathbf{h}}{\partial \mathbf{x}}(\bar{\mathbf{x}}, t) \tag{3.46}$$

As shown by Johansen and Fossen (2017), the cascade of the kinematic observer and KF will inherit the global stability property of the kinematic observer and also benefit from the local optimality properties of the linearized KF.

3.3.1 Stage 1: Kinematic observer

The first step is to design a fixed-gain kinematic observer for the XKF in Fig. 3.6 based on the ship model in Section 3.1.1. The purpose of the kinematic observer is to generate a smooth globally convergent signal $\bar{\mathbf{x}} = [\bar{x}, \bar{y}, \bar{U}, \bar{\chi}]^\top$ for the linearized KF. This can be done by using four fixed-gain observers copying the dynamics (3.1)–(3.4):

$$\dot{\bar{x}} = U\cos(\chi) + K_1(x - \bar{x}) \tag{3.47}$$

$$\dot{\bar{y}} = U\sin(\chi) + K_2(y - \bar{y}) \tag{3.48}$$

$$\dot{\bar{U}} = a + K_3(U - \bar{U}) \tag{3.49}$$

$$\dot{\bar{\chi}} = r + K_4(\chi - \bar{\chi}) \tag{3.50}$$

where $K_i > 0$ $(i = 1, 2, 3, 4)$. It is straightforward to verify that the equilibrium point of the error dynamics $[x - \bar{x}, y - \bar{y}, U - \bar{U}, \chi - \bar{\chi}]^\top = \mathbf{0}$ is GES.

3.3.2 Stage 2: Linearized Kalman filter

The next step is to express the nonlinear system model (3.1)–(3.6) as:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{B}\mathbf{u} + \mathbf{w} \tag{3.51}$$

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{e} \tag{3.52}$$

where $\mathbf{w} \in \mathbb{R}^4$ and $\mathbf{e} \in \mathbb{R}^4$ are Gaussian process and measurement noise, and the only nonlinearity is the vector field $\mathbf{f}(\mathbf{x})$.

The XKF design process involves constructing a linearized KF about $\mathbf{x} = \bar{\mathbf{x}}$ using a Taylor-series expansion:

$$\dot{\hat{\mathbf{x}}} = \mathbf{f}(\bar{\mathbf{x}}) + \mathbf{F}(\bar{\mathbf{x}})(\hat{\mathbf{x}} - \bar{\mathbf{x}}) + \mathbf{B}\mathbf{u} + \mathbf{K}(\mathbf{y} - \mathbf{H}\hat{\mathbf{x}}) \tag{3.53}$$

$$\dot{\mathbf{P}} = \mathbf{F}(\bar{\mathbf{x}})\mathbf{P} + \mathbf{P}\mathbf{F}(\bar{\mathbf{x}})^\top + \mathbf{Q} - \mathbf{K}\mathbf{R}\mathbf{K}^\top \tag{3.54}$$

where $\bar{\mathbf{x}} = [\bar{x}, \bar{y}, \bar{U}, \bar{\chi}]^\top$, $\mathbf{u} = [a, r]^\top$,

$$\mathbf{f}(\bar{\mathbf{x}}) = \begin{bmatrix} \bar{x}_3\cos(\bar{x}_4) \\ \bar{x}_3\sin(\bar{x}_4) \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.55}$$

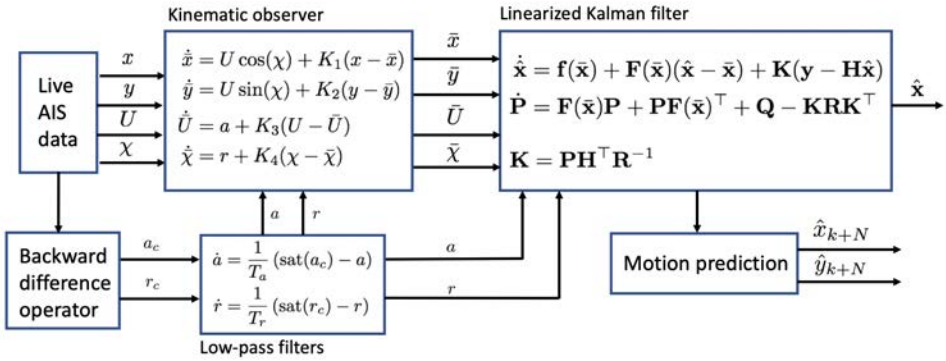


Figure 3.7: Signal flow of the eXogenous Kalman filter.

$$\mathbf{F}(\bar{\mathbf{x}}) = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\bar{\mathbf{x}}) = \begin{bmatrix} 0 & 0 & \cos(\bar{x}_4) & -\bar{x}_3\sin(\bar{x}_4) \\ 0 & 0 & \sin(\bar{x}_4) & \bar{x}_3\cos(\bar{x}_4) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.56}$$

The Kalman gain is computed as:

$$\mathbf{K} = \mathbf{P}\mathbf{H}^\top\mathbf{R}^{-1} \tag{3.57}$$

Theorem 3.1 (XKF)**.** *The fixed-gain kinematic observer* (3.47)–(3.50) *in cascade with the linearized KF* (3.53)–(3.54) *as shown in Fig. 3.7 renders the equilibrium point of the two-stage observer error dynamics GES*[1]*.*

Sketch of proof: *A detailed version of the proof is found in Johansen and Fossen (2017, Theorem 2.1), which makes use of three Assumptions A1-A3. The kinematic observer and linearized KF is a cascaded system. The equilibrium point of the error dynamics of the kinematic observer in Section 3.3.1 is GES. Hence, Assumption A3 clearly holds. This also implies boundedness of* $\mathbf{F}(\bar{\mathbf{x}})$ *due to the smoothness of the vector field* $\mathbf{f}(\bar{\mathbf{x}})$*. Hence, Assumption A1 holds since the LTV system* $(\mathbf{F}(\bar{\mathbf{x}}), \mathbf{B}, \mathbf{H})$ *is uniformly completely observable and controllable. Finally, Assumption A2 is satisfied by choosing the KF tuning matrices* $\mathbf{P}(0), \mathbf{Q}$ *and* $\mathbf{R}$ *symmetric and positive definite. Since Assumptions A1-A3 are satisfied, Johansen and Fossen (2017, Theorem 2.1) guarantees that the equilibrium point of the cascaded system inherits the GES property of the kinematic observer.*

3.3.3 Ship motion predictor

The ship motion predictor is designed by using a discrete-time version of the system model (3.1)–(3.6). Consequently, an N-step predictor with $k = 1, ..., N$ using Euler's integration

[1] As shown by Bhat and Bernstein (2000), systems with rotational degrees of motion cannot be globally stabilized by continuous feedback due to the topological obstruction imposed by SO(3). Hence, the GES property is based on the assumption that $\chi \in \mathbb{R}$ and not $[-\pi, \pi)$. However, if χ is mapped to $[-\pi, \pi)$ when implementing the XKF, Theorem 3.1 still guarantees local exponential stability.

method with sampling time h becomes:

$$\hat{x}(k+1) = \hat{x}(k) + h\hat{U}(k)\cos(\hat{\chi}(k)) \tag{3.58}$$

$$\hat{y}(k+1) = \hat{y}(k) + h\hat{U}(k)\sin(\hat{\chi}(k)) \tag{3.59}$$

$$\hat{U}(k+1) = \hat{U}(k) + ha(k) \tag{3.60}$$

$$\hat{\chi}(k+1) = \hat{\chi}(k) + hr(k) \tag{3.61}$$

$$a(k+1) = a(k) + \frac{h}{T_a}\left(\text{sat}(a_c) - a(k)\right) \tag{3.62}$$

$$r(k+1) = r(k) + \frac{h}{T_r}\left(\text{sat}(r_c) - r(k)\right) \tag{3.63}$$

3.3.4 Implementation aspects for asynchronous AIS data

The AIS data are transmitted using the TCP and UDP Internet protocols. Hence, the XKF must handle delayed measurements, asynchronous communication as well as loss of packets. The XKF (3.47)–(3.50) and (3.53)–(3.54) will run at a fixed time step, typically 30-60 Hz for 3-D visualization applications. Since, the AIS data are transmitted at asynchronous time samples the XKF is implemented in discrete time using the *predictor-corrector* representation (Gelb, 1974). Moreover,

- The discrete-time system model of the XKF is propagated at 30-60 Hz to predict positions, velocity and course angle.
- The state vector is updated and corrected at each time an AIS measurement is received. This happens at much lower frequency (typically 1-2 Hz). Hence, the update times of the corrector must be chosen in multiples of the sampling frequency.

The discrete-time XKF makes use of the superscripts $^+$ and $^-$, which denote the states after and before the measurements are applied. Table 3.2 summarizes the *predictor-corrector* representation for (3.53)–(3.54) and (3.57):

Table 3.2: Four-states eXogenous Kalman filter for AIS data.

Kalman gain:

$$\mathbf{K}(k) = \hat{\mathbf{P}}^-(k)\mathbf{H}^\top\left(\mathbf{H}\hat{\mathbf{P}}^-(k)\mathbf{H}^\top + \mathbf{R}\right)^{-1} \tag{3.64}$$

Corrector (AIS measurements at 1-2 Hz):

$$\hat{\mathbf{x}}^+(k) = \hat{\mathbf{x}}^-(k) + \mathbf{K}(k)\left(\mathbf{y}(k) - \mathbf{H}\hat{\mathbf{x}}^-(k)\right) \tag{3.65}$$

$$\hat{\mathbf{P}}^+(k) = (\mathbf{I}_4 - \mathbf{K}(k)\mathbf{H})\hat{\mathbf{P}}^-(k)\left(\mathbf{I}_4 - \mathbf{K}(k)\mathbf{H}\right)^\top + \mathbf{K}(k)\mathbf{R}\mathbf{K}^\top(k) \tag{3.66}$$

Kinematic observer:

$\bar{\mathbf{x}}(k)$ is computed by (3.47)–(3.50)

Predictor (30-60 Hz for visualization):

$$\mathbf{x}^-(k+1) = \mathbf{x}^+(k) + h\left(\mathbf{f}(\mathbf{x}^+(k)) + \mathbf{F}(\bar{\mathbf{x}}^+(k))(\hat{\mathbf{x}}^+(k) - \bar{\mathbf{x}}(k)) + \mathbf{B}\mathbf{u}(k)\right) \tag{3.67}$$

$$\hat{\mathbf{P}}^-(k+1) := \left(\mathbf{I}_4 + h\mathbf{F}(\bar{\mathbf{x}}^+(k))\right)\hat{\mathbf{P}}^+(k)\left(\mathbf{I}_4 + h\mathbf{F}(\bar{\mathbf{x}}^+(k))\right)^\top + \mathbf{Q} \tag{3.68}$$

Figure 3.8: 3-D visualization of two AIS detected ships moving down the Nidelven river in Trondheim using Unity (2018).

3.3.5 Experimental validation

The XKF and its prediction capabilities were validated using live AIS data from the Trondheim harbor in Norway. Live AIS data were obtained using a VHF antenna. Approximately 30 ships were close to the harbor when logging the data. The North-East coordinate origin was chosen at Munkholmen island, approximately 2 km west of Trondheim harbor.

The longitude and latitude measurements were transformed to xy-positions using the WGS-84 reference systems as outlined in Section 3.1.3. Fig. 3.8 shows the live positions of two ships when moving down the Nidelven river in Trondheim.

The high-speed passenger ferry MS LADEJARL was chosen in the case study, see Fig. 3.9. The ship has MMSI number 257 082 200. Detailed information about the ship can be obtained by using the Marine Vessel Traffic webpage[2] where the MMSI number uniquely identifies the ship.

The top speed of the ship is 37 knots (19.0 m/s). Data was logged for approximately two hours such that the ship had time to move out the fjord via Lensvik towards the North Atlantic and return to Trondheim harbor, see Fig. 3.10.

Motion prediction

The linearized KF was implemented in discrete time at 50 Hz using the predictor-corrector representation (Gelb, 1974) and Euler's method for numerical integration.

The fixed-gain kinematic observer was implemented with $K_1 = K_2 = 10$, $K_3 = 30$ and $K_4 = 50$ (see the Matlab code in Appendix B.2). The KF covariance matrices were chosen as:

$$\hat{\mathbf{P}}^{-}(0) = \mathbf{I}_4 \tag{3.69}$$

$$\mathbf{Q} = \text{diag}(1.0, 1.0, 10.0, 10.0) \tag{3.70}$$

$$\mathbf{R} = \mathbf{I}_4 \tag{3.71}$$

[2] http://www.marinevesseltraffic.com/2013/06/mmsi-number-search.html

Figure 3.9: MS LADEJARL - Passenger ship (gross tonnage 490 t, length overall 38 m and breadth 10 m). Reproduced with kind permission of Fosen Namsos Sjø (http://www.fosennamsos.no).

Sensitivity to wildpoints and corrupted measurements is handled by using a relative large value for the yaw rate time constant T_r in the estimator. This explains why the yaw rate time constant $T_r = 50$ s is chosen 5 times larger than the surge acceleration time constant $T_a = 10$ s.

Two cases as shown in Fig. 3.10 were considered:

- **Case 1:** 30 s motion prediction just before the ship arrives Lensvik harbor.
- **Case 2:** 60 s motion prediction on the route to Trondheim harbor.

As seen from Figs. 3.11–3.12 the ship positions are predicted quite well both for the 30 s and 60 s cases. The first ship indicates where motion prediction is started and the second ship shows the pose of the ship after 30 and 60 s, respectively. The position accuracy will of course be best for ships on a straight course. However, the motion predictor is restarted each time a new measurement is received such that it automatically adjusts to turning maneuvers.

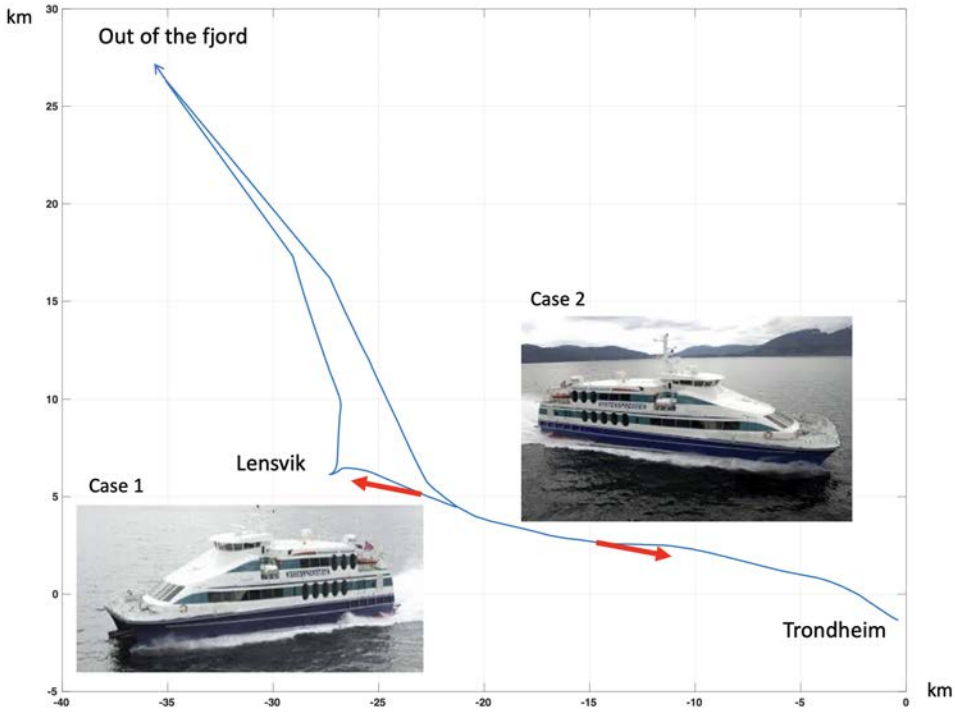


Figure 3.10: North-East positions of MS LADEJARL in km when moving from Trondheim to Lensvik on its way out of the fjord. The return bypasses Lensvik harbor. The red arrows indicate the chosen locations for motion prediction.

Figs. 3.11–3.12 also show the difference of the kinematic observer (red) and the XKF (cyan). The red curve is used for linearization of the Kalman filter. As expected the accuracy of the kinematic observer is not so good as the XKF since this is a simplified decoupled fixed-gain observer. However, the red trajectory is accurate enough to provide the XKF with a trajectory for linearization.

An obvious advantage of the XKF to the fixed-gain kinematic observer is that the covariance of the estimation errors are computed. Hence, it is possible to include outlier detection based on growth in the covariance estimates. Fig. 3.11 also indicates that the performance of the XKF is better than the fixed-gain kinematic observer. Similar observations were made in the case studies presented by Johansen and Fossen (2017). Fig. 3.13 is a 3-D visualization of catamaran vessel in the Trondheim fjord. The green arrow shows the predicted path of the vessel.

The computational requirements typically increase with 25 % when using the XKF instead of the fixed-gain kinematic observers. This is not a problem when using state-of-the-art computers.

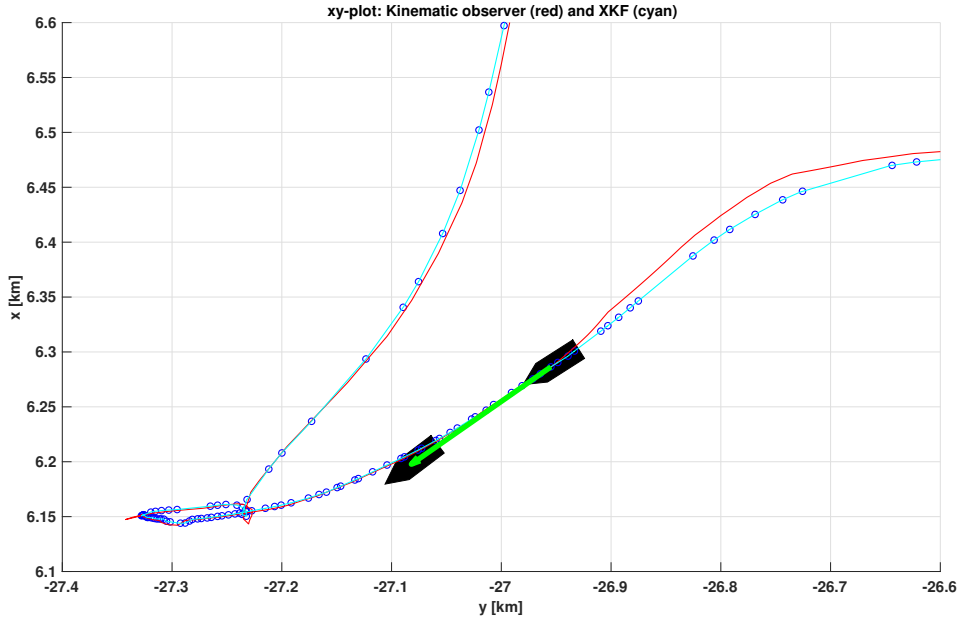


Figure 3.11: The 30 seconds predicted motion of MS LADEJARL just before the ship arrives Lensvik harbor. The red and cyan lines are the kinematic observer and XKF, respectively

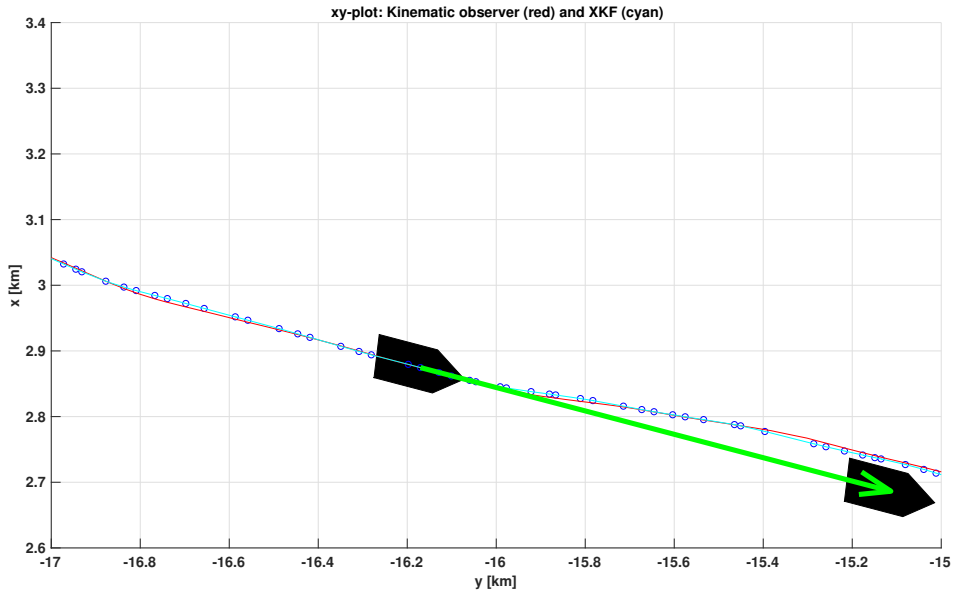


Figure 3.12: The 60 seconds predicted motion of MS LADEJARL when on route to Trondheim harbor. The red and cyan lines are the kinematic observer and XKF, respectively

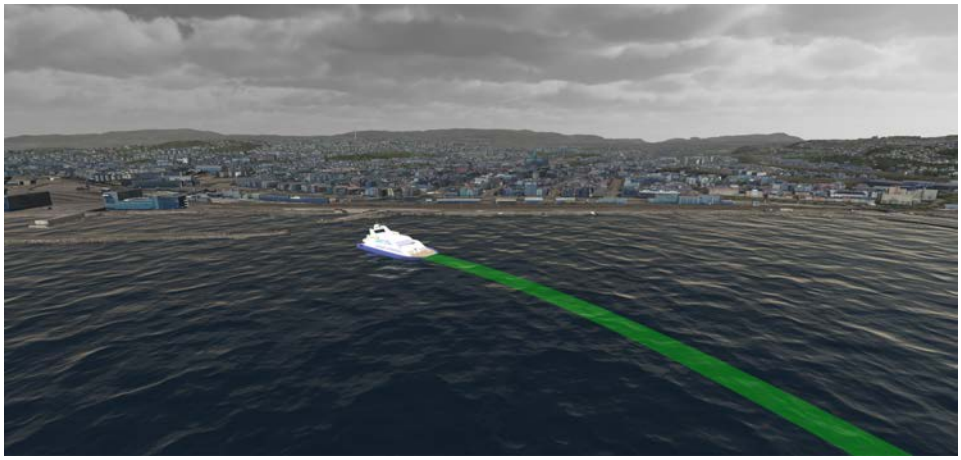


Figure 3.13: 3-D visualization of a catamaran passenger boat in the Trondheim fjord using Unity (2018), and the Hydroform Ocean System and Terraland plugins from the Unity Asset Store. The green arrow shows the predicted path of the vessel.

Fig. 3.14 shows the measured signals (circles) and the estimates in red and cyan. As before, red denotes the kinematic observer and cyan is the XKF. The smoothing effect and increased accuracy of the XKF is more visible in Fig. 3.11 where a clear performance improvement is observed.

3.4 Concluding Remarks

The problem of estimating the motion of ships from live AIS data has been addressed. To the author's knowledge, this problem has not been addressed previously using nonlinear observer theory. For this purpose both an EKF and a XKF for visualization and motion prediction of ships have been constructed. Both estimators show similar performance when using experimental data. It is well-known that the EKF only guarantees local stability, while the XKF renders the equilibrium point of the estimation error GES.

AIS data are transmitted from ships globally and a VHF AIS receiver was used to pick up the coded signals, which are in ASCII character format as specified by NMEA. The AIS sentences were successfully decoded using a parser to obtain real-time ship position, course and speed measurements.

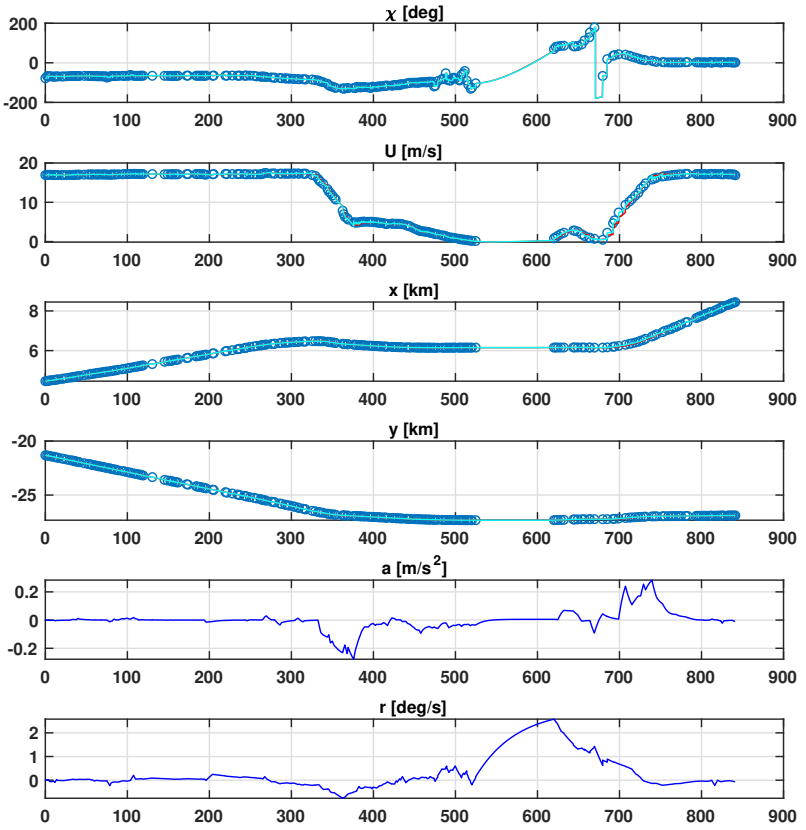


Figure 3.14: Course angle χ [deg], speed U [m/s], x-position [km], y-position [km], acceleration a [m/s^2] and yaw rate r [deg/s] versus time s. The circle denotes the live AIS measurements, while the red and cyan lines are the kinematic observer and XKF estimates, respectively. The period of inactivity corresponds to the stop in Lensvik harbor.

Both the EKF and XKF were validated using asynchronous AIS data from the Trondheim harbor in Norway and it was demonstrated that both state estimators estimate ship position, velocity and course in real time with good accuracy at 50 Hz, which is a good frame rate for 3-D visualization. It was also demonstrated that the estimators could predict future ship positions and thus be used in decision-support systems.

Typical applications of AIS data are motion prediction and automated situation awareness of autonomous ship operations as well as operation of manned and unmanned ships together in restricted areas.

4 Automated Situational Awareness

Situational awareness as defined by Definition 1.3 in Section 1.1.2 is of major importance for a pilot operating an USV in the vicinity of larger ships. This chapter describes some novel concepts for situational awareness and how they can be implemented in a mixed-reality environment to assist the pilot when operating an USV in confined waters with a large number of ships close to the vehicle.

Situational awareness can be quite complex and extensive if a large number of sensory and navigation systems are made available. As pointed out in Section 1.1.2, automated situational awareness can be understood as using mathematical algorithms to issue warnings, estimate future ship positions and compute intercepting paths as well as computing the risk for collision. A human will not be able to process all this information if a large number of moving objects have to be analyzed in real time. Hence, it is necessary to simplify the information flow and illustrate this graphically for the pilot.

This chapter describes the algorithms for situational awareness and how they can be visualized in a mixed-reality environment. The main results have been published in the following paper:

S. Fossen, R. T. Bye and O. L. Osen (2018). Visualization and Collision Risk Assessment of Real Ships in a Mixed Reality Environment using Live Automatic Identification System (AIS) Data. *Proc. of the 2nd European Conference on Electrical Engineering and Computer Science (EECS'18)*, Bern, 20–22 December 2018.

Some benefits of the automated situational awareness system presented in this chapter are:

- 3-D visualization of a simulated remotely-piloted USV operating in common waters with real ships in a virtual world chosen as the Trondheim fjord in Norway.
- On-line situation assessment, encompassing automated threat and impact assessment, to assist the human operator by estimating the motion of ships when operating in restricted waters such as a fjord.
- Issue warnings by computing the minimum separation between the USV and AIS ships online and graphical presentation of the AIS ships collision risk indexes.
- When the minimum separation is to small, the situational awareness system can assist the USV operator by analyzing different evasive maneuvers to increase the distance between the USV and the AIS ships.

4.1 International Regulations for Preventing Collisions at Sea

The increased use of USV and ASV in common waters with ships introduce new operational aspects, which again introduces a need for new regulations. A USV is usually remotely controlled by a pilot who must analyze the ship traffic and other operational aspects in order to prevent a collision. The operator information is critical for safe operation and it is important to be aware of the limitations of the human cognitive processes.

Sensor fusion can be used to present the pilot with organized and coherent information to make timely and informed decisions. As the amount of sensor data increase, it becomes

more and more difficult for human operators to achieve situational awareness (Holsopple et al., 2010). Consequently, an automated process, which estimate and project situations using algorithms is needed. This is refereed to as *automated situational awareness*.

4.1.1 COLREGS

An automated situational awareness system for collision detection and ship collision risk assessment should comply with the *International Regulations for Preventing Collisions at Sea*, in which Rule 7 states (COLREGS, 1972):

1. "Every vessel shall use all available means appropriate to the prevailing circumstances and conditions to determine if risk of collision exists. If there is any doubt such risk shall be deemed to exist."
2. "Proper use shall be made of radar equipment if fitted and operational, including long-range scanning to obtain early warning of risk of collision and radar plotting or equivalent systematic observation of detected objects."
3. "Assumptions shall not be made on the basis of scanty information, especially scanty radar information."
4. "In determining if risk of collision exists the following considerations shall be among those taken into account: (a) such risk shall be deemed to exist if the compass bearing of an approaching vessel does not appreciably change; (b) such risk may sometimes exist even when an appreciable bearing change is evident, particularly when approaching a very large vessel or a tow or when approaching a vessel at close range."

4.2 Minimum Separation Algorithm

Collision avoidance for multiple ships can be stated as a problem of maintaining safe distance between ships in conflict. This can be solved as an optimal collision avoidance problem minimizing a given cost function, while simultaneously satisfying the constraints specified by COLREGS, see Johansen et al. (2016) and references therein.

The spatial and temporal closeness between two vessels (USV and an AIS ship) at any instant of time are categorized by the following two metrics (Xu et al., 2016):

- Time to Closest Point of Approach (TCPA)
- Distance at Closest Point of Approach (DCPA)

By computing the TCPA and DCPA values online for all AIS ships, it is possible to visualize this information and thus improve situational awareness. These are also the main parameters for risk assessment (Xu et al., 2016).

A marine radar with automatic radar plotting aid (ARPA) can also be used as source for TCPA and DCPA. However, this is an expensive solution compared to an AIS receiver and smaller vehicles such as a remotely-operated USV does not necessarily have a radar system onboard.

The coordinate origin of the ships are chosen midships on the centerline. Hence, in order to avoid a collision it is convenient to specify a safety distance to each AIS ship by

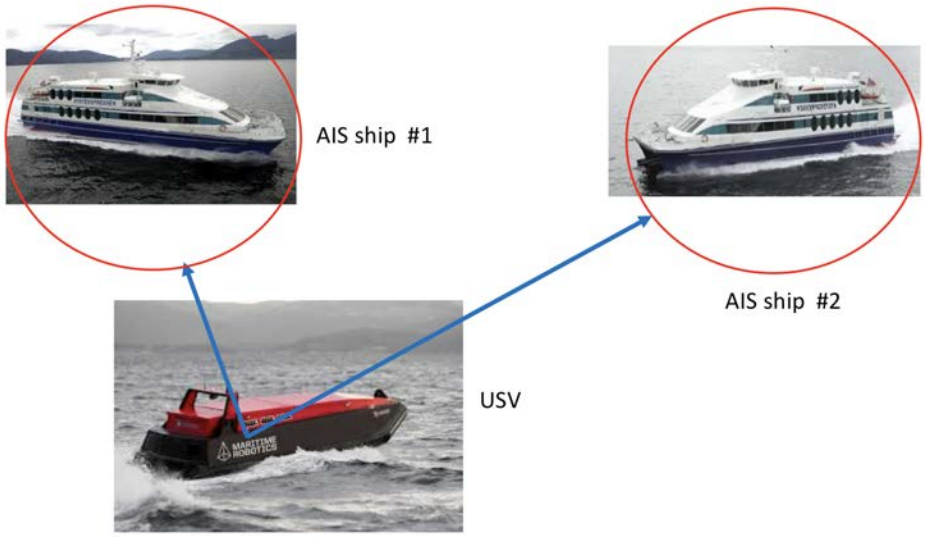


Figure 4.1: Distance and safety distance (red circle) between USV and AIS ships.

defining a circle with radius $R_i > 0$ as shown in Fig. 4.1. Furthermore, let (x, y) denote the *interceptor* (USV) position and let (x_i, y_i) where $i = 1, 2, \cdots, N$ be the AIS ship positions.

When computing the TCPA and DCPA metrics it will be assumed that $U = U(t_0) =$ constant and $\chi = \chi(t_0) =$ constant to find an explicit solution for each AIS measurement. This assumption can be relaxed by integrating the nonlinear system (3.1)–(3.4) numerically with a and r as optionally inputs.

Hence, the interceptor position at time t is obtained by integrating (3.1)–(3.2) from time t_0 to t with constant speed and course angle, which gives:

$$x(t) = x(t_0) + U(t_0)\cos(\chi(t_0))\,(t - t_0) \tag{4.1}$$

$$y(t) = y(t_0) + U(t_0)\sin(\chi(t_0))\,(t - t_0) \tag{4.2}$$

where $(x(t_0), y(t_0), \chi(t_0))$ is the initial pose of the ship. The AIS ship positions at time t are found in a similar manner:

$$x_i(t) = x_i(t_0) + U_i(t_0)\cos(\chi_i(t_0))\,(t - t_0) \tag{4.3}$$

$$y_i(t) = y_i(t_0) + U_i(t_0)\sin(\chi_i(t_0))\,(t - t_0) \tag{4.4}$$

The relative position errors between the USV and AIS ship #i then become:

$$e_{x_i}(t) = x(t) - x_i(t) := x_{0_i} + V_{x_i} t \tag{4.5}$$

$$e_{y_i}(t) = y(t) - y_i(t) := y_{0_i} + V_{y_i} t \tag{4.6}$$

where the four constants are recognized as:

$$x_{0_i} = x(t_0) - U(t_0)\cos(\chi(t_0))\,t_0 - x_i(t_0) + U_i(t_0)\cos(\chi_i(t_0))\,t_0 \tag{4.7}$$

$$y_{0_i} = y(t_0) - U(t_0)\sin(\chi(t_0))\,t_0 - y_i(t_0) + U_i(t_0)\sin(\chi_i(t_0))\,t_0 \tag{4.8}$$

$$V_{x_i} = U(t_0)\cos(\chi(t_0)) - U_i(t_0)\cos(\chi_i(t_0)) \tag{4.9}$$

$$V_{y_i} = U(t_0)\sin(\chi(t_0)) - U_i(t_0)\sin(\chi_i(t_0)) \tag{4.10}$$

The instantaneous separation S_i between the USV and AIS ship #i satisfies:

$$S_i^2 = e_{x_i}^2 + e_{y_i}^2 = (x_{0_i} + V_{x_i}t)^2 + (y_{0_i} + V_{y_i}t)^2 \tag{4.11}$$

or

$$S_i^2 = x_{0_i}^2 + y_{0_i}^2 + 2t(x_{0_i}V_{x_i} + y_{0_i}V_{y_i}) + (V_{x_i}^2 + V_{y_i}^2)t^2 \tag{4.12}$$

4.2.1 Time to closest point of approach (TCPA)

The closest point of approach is obtained from (4.12) by solving $\dot{S}_i = 0$ for t, which gives:

$$\mathrm{TCPA}_i = \frac{-(x_{0_i}V_{x_i} + y_{0_i}V_{y_i})}{V_{x_i}^2 + V_{y_i}^2} \tag{4.13}$$

A negative TCPA_i means that the distance to the closest point of approach has already happened.

4.2.2 Distance at closest point of approach (DCPA)

The minimum distance between the USV and AIS ship #i should never be less than the specified radius R_i. Hence, it is required that $\mathrm{DCPA}_i > R_i$ in order to avoid collision. Insertion of $t = \mathrm{TCPA}_i$ in (4.12) gives the implicit formula for the DCPA_i value:

$$\mathrm{DCPA}_i^2 = x_{0_i}^2 + y_{0_i}^2 + 2\,\mathrm{TCPA}_i\left(x_{0_i}V_{x_i} + y_{0_i}V_{y_i}\right) + \left(V_{x_i}^2 + V_{y_i}^2\right)\mathrm{TCPA}_i^2 \tag{4.14}$$

4.3 Ship Collision Risk Index

The ship collision risk index of Xu et al. (2016) is used for collision risk assessment. The USV maneuvering performance is specified in terms of three ship-dependent parameters V_0, TCPA_s and DCPA_s such that the USV speed satisfies:

$$V_0 = \frac{\mathrm{DCPA}_s}{\mathrm{TCPA}_s} \tag{4.15}$$

The risk index for an approaching AIS ship with TCPA_i and DCPA_i values (computed using (4.13) and (4.14)) is:

$$\mathrm{RI}_i = \frac{\sqrt{\mathrm{DCPA}_s^2 + (\eta_i\,V_0\,\mathrm{TCPA}_s)^2}}{\sqrt{\mathrm{DCPA}_i^2 + (\eta_i\,V_0\,\mathrm{TCPA}_i)^2}} \tag{4.16}$$

where the weighting factor η_i depends on from which side the approaching ship is coming from:

$$\eta_i = \begin{cases} 0.5\frac{\mathrm{TCPA}_i - \mathrm{TCPA}_s}{\mathrm{TCPA}_s} & \text{coming from starboard side} \\ 0.55\frac{\mathrm{TCPA}_i - \mathrm{TCPA}_s}{\mathrm{TCPA}_s} & \text{coming from port side} \end{cases} \tag{4.17}$$

Figure 4.2: The vehicle control station for operation of a pilot-controlled USV in a mixed-reality environment. Unity is used to visualize the Trondheim fjord. The grey circle represents the safety region of the USV, while AIS detected ships are plotted with green, yellow or red circles to indicate if the they present a collision risk or not for the USV. Reproduced with kind permission of Maritime Robotics AS.

Figure 4.3: 3-D visualization of a pilot-controlled USV, which is moving up the Nidelven river in Trondheim. One live AIS ship is detected on the other side of the bridge.

4.4 Experimental Validation

A vehicle control station is used to operate the USV in a mixed-reality environment, see Fig. 4.2. The graphics is implemented using Unity. Three 3-D visualization scenarios are presented below:

Case 1 (EKF motion data for visualization of AIS ships): Visualization of a pilot-controlled USV, which is moving up the Nidleven river in Trondheim.

Case 2 (Online risk assessment): Visualization of a pilot-controlled USV and online collision risk indexes for the approaching AIS ships.

Case 3 (Collision detection by motion prediction for varying evasive maneuvers): Motion prediction is used to identify possible collisions by computing the instantaneous separation between the ships for different evasive maneuvers.

The case studies show how the minimum separation and risk index algorithms can be visualized to enhance situational awareness. The main tool for this is interactive visualization in a mixed-reality environment using a vehicle control station. The vehicle control station is shown in Fig. 1.3.

Figure 4.4: 3-D visualization of the pilot-controlled USV, which is approaching several live AIS ships outside the Trondheim harbor. The colored lines show the EKF predicted path of the approaching ships using AIS data. The collision risk index is computed online and a green line indicates small risk for collision, yellow is moderate risk, while a red line indicates high risk for collision.

4.4.1 Case 1: EKF motion data for visualization of AIS ships

The first case study visualizes a pilot-controlled USV, which is moving up the Nidelven river in Trondheim. The EKF in Section 3.2 is implemented at 50 Hz in order to generate data

Figure 4.5: The "green-yellow-red" color map (4.19) for visualization of collision risk.

that can be used to visualize the AIS ships at 50 FPS. The scenario is the Trondheim fjord and the Nidelven river located on the west coast of Norway. Live AIS data were obtained using a VHF antenna and by decoding the NMEA messages. Approximately 20 ships were operating outside the Trondheim harbor when receiving the AIS data. Fig. 4.3 shows the virtual world used to illustrate the river in Unity.

4.4.2 Case 2: Online risk assessment and visualization

The second case study visualizes a USV and the online collision risk indexes for the approaching AIS ships, see Fig. 4.4. The USV risk assessment parameters are chosen as $V_0 = 5$ m/s and $\text{TCPA}_s = 30$ s. This gives.

$$\text{DCPA}_s = V_0 \cdot \text{TCPA}_s = 150 \text{ m} \tag{4.18}$$

Hence, the risk index for each AIS ships can be computed online using (4.16).

The USV is approaching several AIS ship and risk for collision is visualized by using an RGBA (red-green-blue-alpha) color space for varying risk indexes (RI) in order to improve situational awareness. The additional parameter, alpha, is used to specify opaqueness of each pixel. The following formula for the RGBA values was found to give good results:

$$[R, G, B, A] = [\min(RI, 1), 1 - \min(RI, 1), 0, 0.3] \tag{4.19}$$

where $\min(RI, 1)$ ensures that the R and G values are within $[0, 1]$. The B value is chosen as 0 to get a "green-yellow-red" warning for collision as shown in Fig. 4.5.

4.4.3 Case 3: Collision detection by motion prediction for varying evasive maneuvers

The motion predictors can be used to identify possible collisions by computing the instantaneous separation between the ships. Let the coordinate origin of the ships be located midships on the centerline. Furthermore, let (x, y) denote the *interceptor* (USV) position and (x_i, y_i) where $i = 1, 2, \cdots, N$ be the AIS ship positions. The relative position errors between the USV and AIS ship #i then become:

$$e_{x_i}(k) = x(k) - x_i(k) \tag{4.20}$$

$$e_{y_i}(k) = y(k) - y_i(k) \tag{4.21}$$

and the minimum separation $S(k)$ between the USV and AIS ships are:

$$S(k) = \min_i \sqrt{e_{x_i}^2(k) + e_{y_i}^2(k)} \tag{4.22}$$

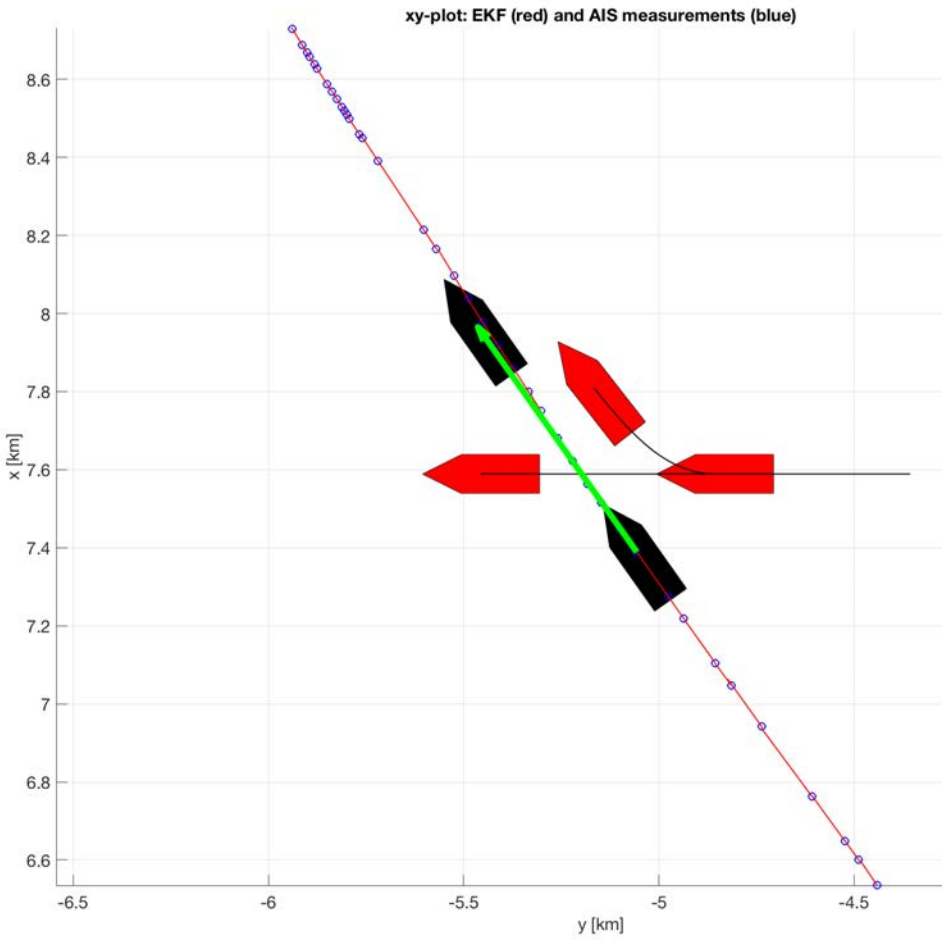


Figure 4.6: An USV approaches MS Trondheimsfjord II from the East. The ship and USV motions are predicted in 60 s to show the effect of an evasive maneuver.

Predictor for evasive maneuvers

If the minimum separation (4.22) is to small, the USV speed and course (3.30)–(3.31) can be modified to include the effect of an evasive USV maneuver.

A typical collision avoidance maneuver can be specified in terms of a speed command U_c and a course angle command χ_c, for instance:

- Reduce the intercepter speed $U(0)$ by 50 %:

$$U_c = 0.5\, U(0) \tag{4.23}$$

- Align the interceptor course $\chi(0)$ to the target course:

$$\chi_c = \chi_i(0) \tag{4.24}$$

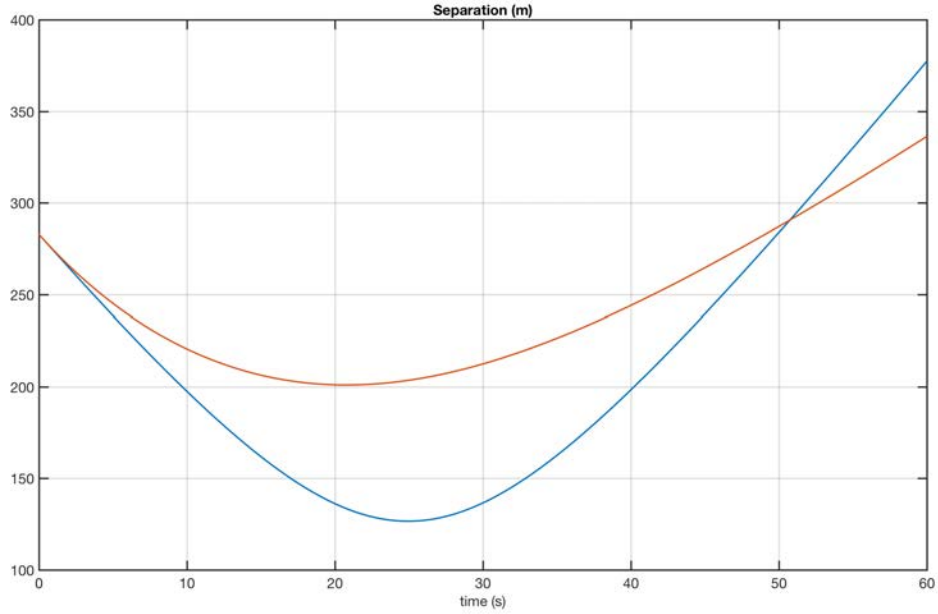


Figure 4.7: Minimum separation between an USV approaching MS Trondheimfjord II from the East (blue) and when performing an evasive maneuver (red). The minimum distance increases from 126.6 m to 201.0 m.

The next step is to replace (3.30)–(3.31) with the closed-loop speed and course dynamics, typically first-order systems, such that:

Motion predictor for USV evasive maneuvers:

$$x(k+1) = x(k) + hU(k)\cos(\chi(k)) \tag{4.25}$$

$$y(k+1) = y(k) + hU(k)\sin(\chi(k)) \tag{4.26}$$

$$U(k+1) = U(k) + \frac{h}{T_{\text{speed}}}(U_c - U(k)) \tag{4.27}$$

$$\chi(k+1) = \chi(k) + \frac{h}{T_{\text{course}}}\,\text{ssa}(\chi_c - \chi(k)) \tag{4.28}$$

where $\text{ssa}(\cdot)$ is the *smallest signed angle* defined by (3.27), $U(0) = U(t_0)$ and $\chi(0) = \chi(t_0)$. The time needed to perform the maneuver is specified by the user inputs T_{speed} and T_{course}.

AIS data for MS Trondheimsfjord II (see Fig. 3.2) are used to demonstrate ship motion prediction. The EKF parameters were chosen as in Section 3.2.3. The asynchronous AIS data are processed by the EKF to obtain equally-spaced data at 50 Hz.

The USV is approaching the MS Trondheimsfjord II from East. The ship and USV motions are predicted using a 60 s future horizon. The USV speed is 10 m/s and the course is −90 deg, see Fig. 4.6. In order to increase the minimum separation of the USV and the ship, an evasive maneuver is implemented using the motion predictor (4.25)–(4.28). Fig. 4.6 verifies that USV course aligns to the ship course as expected and that the USV speed is

reduced from 10 m/s to 5 m/s during the turn. Fig. 4.7 shows that the minimum separation between the approaching USV and the ship is increased from 126.6 m to 201.0 m by the evasive maneuver.

4.5 Concluding Remarks

This chapter has presented a novel concept for automated situational awareness using 3-D visualization. Unity was used for visualization of live ships operating in a mixed-reality environment. A pilot-controlled USV was simulated to show how it could operate close to real ships and visualization techniques were used for situational assessment.

The ship positions were obtained by decoding AIS data online using a VHF antenna. After decoding, an EKF was designed to run at a fixed time step, typically 30-60 Hz, for smooth visualization at high frame rates. Since the AIS data were transmitted at asynchronous time samples, the EKF is implemented in discrete time using the *predictor-corrector* representation to get evenly-spaced data and to handle loss of data packets. The EKF has been used for motion prediction and to evaluate the minimum separation between ships for varying evasive maneuvers.

An online algorithm for computation of the minimum distance between the USV and the AIS ships was coded in Unity's native C# scripting language. Finally, the concept of automated situational awareness was demonstrated by online computation of a ship collision risk index using live AIS data from the Trondheim harbor in Norway. An RGBA (red-green-blue-alpha) color space for varying risk indexes (RI) was used to illustrate risk i mixed-reality environment when approaching AIS detected ships.

5 *Conclusions and Challenges for Future Research*

This chapter concludes the work in the thesis and identify future topics for research related to automated situational awareness.

5.1 Conclusions

The thesis has presented a framework for an automated situational awareness system for human operators. In this context, situational awareness means having an accurate understanding of what is happening around you and what is likely to happen.

5.1.1 Automated situational awareness systems

The presented automated situational awareness system has been implemented in Unity by using an agile software development model. The advantage of this was that the software could be developed and tested in incremental steps, which allowed for dynamic updating of the requirement specifications. The following subsystems have been developed during this process:

AIS decoder: Parser for decoding of live AIS messages as specified by NMEA (2018).

State estimator: Two nonlinear Kalman filter algorithms (EKF and XKF) for interpolation of asynchronous AIS measurements at 30-60 FPS. This made smooth visualization possible. The Kalman filter algorithms also handle TCP and UDP packet losses.

Motion prediction: Prediction of ships and USV motions for a user specified future horizon. USV evasive maneuvers are also included in the motion predictor.

Risk assessment by minimum separation: Computation of online minimum separation parametrized in terms of time and distance to the closest point of approach (TCPA and DCPA). These are the main metrics for computation of the risk index.

Unity Game Engine: Unity was used to develop a mixed-reality environment for visualization of ships moving in the Trondheim fjord, Norway. This included realistic waves and their effect on the ship motions as well as buildings in the Trondheim harbor and the city.

3-D Visualization: Automated situational awareness was visualized by showing the USV and AIS ships in real time using a mixed-reality environment when operating in the Trondheim fjord and harbor. Predicted USV and ship motions, evasive maneuvers and risk index (parametrized using colors) were shown to assist the operator.

5.1.2 Main contributions

The main contributions of the thesis have been published in one journal paper (Fossen and Fossen, 2018a) and two conference papers (Fossen and Fossen, 2018b), (Fossen et al., 2018). The main contributions can be summarized as:

- A novel automated situational awareness systems for a pilot-operated USV in a mixed-reality environment with live visualization of AIS detected ships to improve the human operators cognitive processes.
- Decoding of live AIS data from a VHF antenna using a parser, which was scripted in C# and implemented in Unity.
- An extended Kalman filter and a globally exponentially stable nonlinear observer (eXogenous Kalman filter) for ship tracking, motion prediction and 3-D visualization at 30-60 FPS. Both state estimators handle asynchronous data and TCP/UDP packet losses.
- On-line ship collision risk assessment and visualization of risk for enhanced automated situational awareness using Kalman filters for motion prediction.
- Experimental validation of state estimators, motion prediction algorithms, collision detection algorithms and online risk assessment by 3-D visualization of AIS detected ships in a mixed-reality environment.

5.2 Future Work

The presented results are based on a single source of information, that is AIS measurements. Future research on automated situational awareness systems should look into the possibility on combing all type of navigation systems and sensory information using machine learning and state estimation methods. Sensor fusion is the combination of navigation and sensory data in an optimal manner such that the resulting information has less uncertainty than would be possible when these sources were used individually.

For automated situational awareness the following system should be considered for sensor fusion, risk assessment and decision making:

- AIS for global navigation data
- Ultra-wideband (UWB) radio for local navigation and docking of ships
- Radar for collision avoidance and detection, and tracking in ships using ARPA
- Lidar for collision avoidance and detection, and local navigation
- Infrared cameras for collision avoidance and detection, and local navigation
- Optical cameras for situational assessment
- Satellite data for prediction of wind, waves and currents

In addition to this, it is important to develop 3-D visualization systems, which makes it possible for humans to analyze large amount of information in real time without saturating the human cognitive system. This can be done by identifying processes that can be automated such that human decisions can be made by an automated situational awareness system.

Algorithms

A

A.1 Backward Difference Approximation of the First Derivative

A.1.1 Asynchronous Data

The backward difference approximation of the first derivative for three asynchronous data points $F(k)$, $F(k-1)$ and $F(k-2)$ at times t_k, t_{k-1} and t_{k-2}, respectively can be derived from the Taylor-series expansions:

$$F(k-1) = F(k) - h_1 F'(k) + \frac{1}{2} h_1^2 F''(k) + O(h_1^3) \tag{A.1}$$

$$F(k-2) = F(k) - (h_1 + h_2) F'(k) + \frac{1}{2} (h_1 + h_2)^2 F''(k) + O((h_1 + h_2)^3) \tag{A.2}$$

where $h_1 = t_k - t_{k-1}$ and $h_2 = t_{k-1} - t_{k-2}$. Multiplying (A.1) with

$$\alpha = \frac{(h_1 + h_2)^2}{h_1^2} \tag{A.3}$$

and subtracting (A.2) from (A.1) makes $F''(k)$ vanish in (A.1) and (A.2). Hence, the error will be of order $O((h_1 + h_2)^3)$. This gives

$$\alpha F(k-1) - F(k-2) = (\alpha - 1) F(k) + (-(\alpha - 1) h_1 + h_2)\, F'(k) \tag{A.4}$$

Solving for $F'(k)$, gives the formula:

$$\boxed{F'(k) = \frac{(1-\alpha) F(k) + \alpha F(k-1) - F(k-2)}{(1-\alpha) h_1 + h_2}} \tag{A.5}$$

A.1.2 Synchronous Data

For synchronous data $h = h_1 = h_2$. Hence, (A.4) gives $\alpha = 4$ and (A.5) reduces to:

$$\boxed{F'(k) = \frac{3F(k) - 4F(k-1) + F(k-2)}{2h}} \tag{A.6}$$

This expression is recognized as the second-order backward difference operator for $F'(k)$.

B Matlab Scripts

B.1 Extended Kalman Filter for AIS Data

The EKF and motion prediction algorithms in Section 3.2 were implemented and tested in Matlab using experimental data before they were coded as a methods in Unity. The experimental data was logged using an AIS receiver and the data format is:

```
% DATA = [ time   MMSI   x   y   U   chi ]
DATA = [...
 1.57   258177000  4459.52  -21316.17  32.900  -118.300
 1.62   257013700  1470.57  -9483.21   12.900   149.800
 3.25   257333000  2715.71  -15846.23  36.500   100.500
 ... ];
```

The EKF is implemented for the case when AIS data are received at a low measurement rate (typically 0.5 Hz or slower). For this case the code uses $a = r = 0$.

An extension to the case when the measurements arrive at higher measurement rates is found in Appendix B.2 where the backward difference operator is used to compute a and r (see Appendix A.1).

```
% 4-state discrete-time EKF with motion prediction (a = r = 0)
AISdata1                    % load AIS data
A = unique(DATA(:,2));      % find unique ship indexes in data set
idx = 4;                    % Ship #4, between Vanvikan-Trondheim
N1 = 800;                   % start sample
N2 = 1500                   % final sample

% extract all data for ship with index idx, store data in table ship1
j = 1;
for i = N1:N2
    if (DATA(i,2) == A(idx))
        ship1(j,:) = DATA(i,:);
        j = j+1;
    end
end
MMSI = ship1(1,2)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Motion prediction data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
k_p = 7000;                 % start sample
tfinal = 30;                % duration in seconds

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Measurements
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
t = round( ship1(:,1), 2);  t = t - t(1);   % 2 digits time
x = ship1(:,3);
```

```
y = ship1(:,4);
U = ship1(:,5) * 0.514444;                  % knots to m/s

chi = atan2( diff([0; y]),diff([0; x]) );   % path-tangential angle
chi = wrapToPi(chi);

t_max = t(length(t));
t_sampling = mean(diff(t))

h = 0.02;                                   % 50 Hz
k = 1;
M = round(t_max)/h;
N = length(t);
t_update = t(1);

% initialization of EKF:  X = [x y U chi]
Q = diag([0.01 0.01 0.1 0.1]);
R = diag([0.001 0.001 0.001 0.01]);
X_prd = [x(1) y(1) U(1) chi(1)]';
P_prd = 0.1 * eye(4);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MAIN LOOP
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

simdata = zeros(M-1,7);                     % memory allocation

for i = 1:M-1
    time = (i-1)*h;                         % time (sec)

    % Corrector with K = 0 (no update)
    X_hat = X_prd;
    P_hat = P_prd;

    % Measurements
    if (time ≥ t_update)
        x_k = x(k);
        y_k = y(k);
        U_k = U(k);
        chi_k = chi(k);

        z_k = [x_k y_k U_k chi_k]';
        eps = z_k - X_prd;
        eps(4) = wrapToPi(eps(4));

        % Corrector
        K = P_prd * inv(P_prd + R);

        X_hat = X_prd + K * eps;
        P_hat = (eye(4)-K) * P_prd * (eye(4)-K)' + K * R *K';

        if k < N
            k = k + 1;
```

```
82              t_update = t(k);
83          end
84      end
85
86      % Store simulation data in a table
87      simdata(i,:) = [time X_prd' P_prd(1,1) P_prd(2,2)];
88
89      % Predictor (k+1)
90      f_hat = [ X_hat(3) * cos(X_hat(4))
91                X_hat(3) * sin(X_hat(4))
92                0
93                0 ];
94
95      A   = [ 0 0 cos(X_hat(4)) -X_hat(3) * sin(X_hat(4))
96              0 0 sin(X_hat(4))  X_hat(3) * cos(X_hat(4))
97              0 0 0 0
98              0 0 0 0  ];
99
100     PHI = eye(4) + A * h;
101
102     X_prd = X_hat + h * f_hat;
103     P_prd = PHI * P_hat * PHI' + Q;
104
105 end
106
107 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
108 % Motion prediction from time t(k_p)
109 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
110 k_p          = round(t(k_p)/h);
111 x_p(1)       = simdata(k_p,2);
112 y_p(1)       = simdata(k_p,3);
113 U_p(1)       = simdata(k_p,4);
114 chi_p(1)     = simdata(k_p,5);
115
116 h_p = 0.1;
117 for i = 1:tfinal/h_p
118     x_p(i+1)    = x_p(i)   + h_p * U_p(i) * cos( chi_p(i) );
119     y_p(i+1)    = y_p(i)   + h_p * U_p(i) * sin( chi_p(i) );
120     U_p(i+1)    = U_p(i)   + h_p * 0;
121     chi_p(i+1)  = chi_p(i) + h_p * 0;
122 end
123
124 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
125 % PLOT SIMULATION DATA: x = [x y U chi] AND PREDICTED SHIP
126 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
127 t_prd = simdata(:,1);
128 x_prd = simdata(:,2)/1000;
129 y_prd = simdata(:,3)/1000;
130 U_prd = simdata(:,4);
131 chi_prd = simdata(:,5);
132 chi_prd = wrapToPi(chi_prd);
133
134 P11 = simdata(:,6);
```

```
135 P22 = simdata(:,7);
136
137 x = x/1000;
138 y = y/1000;
139 x_p = x_p/1000;
140 y_p = y_p/1000;
141
142 figure(1)
143 xship = [-1  , 1/3, 1/3, 1, 1/3, 1/3,-1  ];   % draw ship
144 yship = [-1/3,-1/3,-1/3, 0, 1/3, 1/3, 1/3];
145 g1 = hgtransform;
146 g2 = hgtransform;
147 patch('XData',xship,'YData',yship,'FaceColor','black','Parent',g1);
148 patch('XData',xship,'YData',yship,'FaceColor','black','Parent',g2);
149
150 hold on
151 plot(y,x,'bo',y_prd,x_prd,'r','LineWidth',1);
152 hold off; grid
153
154 set(gca,'fontsize',12)
155 xlabel('y [km]'); ylabel('x [km]')
156 title('xy-plot: EKF (red) and AIS measurements (blue)')
157
158 hold on
159 quiver(y_p(1),x_p(1),y_p(tfinal/h_p)-y_p(1),...
160        x_p(tfinal/h_p)-x_p(1),1,'g','LineWidth',5)
161 g1.Matrix = makehgtform('translate',[y_p(1),x_p(1),0],...
162        'scale',0.6,'zrotate',pi/2-chi_p(1));
163 g2.Matrix = ...
    makehgtform('translate',[y_p(tfinal/h_p),x_p(tfinal/h_p),0],...
164             'scale',0.6,'zrotate',pi/2-chi_p(tfinal/h_p));
165 drawnow
166 hold off
167
168 figure(2)
169 subplot(411)
170 plot(t,(180/pi)*chi,'o', ...
    t_prd,(180/pi)*chi_prd,'r'),grid,title('\chi [deg]');
171 set(gca,'fontsize',12)
172 subplot(412)
173 plot(t,U,'o', t_prd,U_prd,'r','LineWidth',1),grid,title('U [m/s]');
174 subplot(413)
175 plot(t,x,'o',t_prd,x_prd,'r','LineWidth',1),grid,title('x [km]');
176 set(gca,'fontsize',12)
177 subplot(414)
178 plot(t,y,'o',t_prd,y_prd,'r','LineWidth',1),grid,title('y [km]');
179 set(gca,'fontsize',12)
```

B.2 eXogenous Kalman Filter for AIS Data

The XKF and motion prediction algorithms in Section 3.3 were implemented and tested in Matlab using experimental data before they were coded as a methods in Unity. The experimental data was logged using an AIS receiver and the data format is:

```
% DATA = [ time   MMSI   x   y   U   chi ]
DATA = [...
 1.57   258177000   4459.52   -21316.17   32.900   -118.300
 1.62   257013700   1470.57   -9483.21    12.900    149.800
 3.25   257333000   2715.71   -15846.23   36.500    100.500
 ... ];
```

The XKF is implemented for the case when AIS data are received at high measurement rate (typically 0.5 Hz or faster). For this case the code uses the backward difference operator in Appendix A.1) to compute a and r from the AIS data.

```
% 4-state discrete-time XKF with motion prediction
AISdata1                    % load AIS data
A = unique(DATA(:,2));      % find unique ship indexes in data set
idx = 10;                   % Ship #10, between Trondheim-Brekstad
N2MAX = length(DATA(:,1));
N1 = 800;                   % start sample
N2 = 1500                   % final sample

CASE = 1;
if (CASE == 1)              % Out of the fjord
    N1 = 1;
    N2 = 1100;
elseif (CASE == 2)          % In the fjord
    N1 = 3200;
    N2 = N2MAX;
else  % all data
    N1 = 1;
    N2 = N2MAX;
end

j = 1;
for i = N1:N2
    if (DATA(i,2) == A(idx))
        ship1(j,:) = DATA(i,:);
        j = j+1;
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Motion prediction data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
h_p = 0.1;                  % 10 Hz plots

kmax = length(ship1);
```

```
k_p = round(kmax/2);          % start sample  k_p < k_max
tfinal = 30;                  % duration in seconds

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Measurements
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
t = round( ship1(:,1), 2);  t = t - t(1);       % 2 digits time
x = ship1(:,3);
y = ship1(:,4);
U = ship1(:,5) * 0.514444;

chi = atan2( diff([0; y]),diff([0; x]) );       % path-tangential angle
chi = wrapToPi(chi);

t_max = t(length(t));

h = 0.025;                                       % 40 Hz
k = 1;
M = round(t_max)/h;
N = length(t);
t_update = t(1);

% initialization of kinematic observer:  x = [x y U chi]
x_prd = x(1);
y_prd = y(1);
U_prd = U(1);
chi_prd = chi(1);
a = 0;
r = 0;

K1 = 10;
K2 = 10;
K3 = 30;
K4 = 50;

% initialization of LTV Kalman filter
Q = diag([1 1 10 10]);
R = eye(4);

P = eye(4);
x_hat = [x(1) y(1) U(1) chi(1)]';

T_a = 10;   % acceleration time constant
T_r = 50;   % yaw rate time constant

B = [ 0 0
      0 0
      1 0
      0 1 ];

H = eye(4);


```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MAIN LOOP
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
simdata = zeros(M-1,11);                    % memory allocation

for i = 1:M-1
    time = (i-1)*h;                          % time (sec)

    % Store simulation data in a table
    simdata(i,:) = [time x_prd y_prd U_prd chi_prd x_hat' a r ];

    % Measurements
    if (time ≥ t_update)
        x_k = x(k);
        y_k = y(k);
        U_k = U(k);
        chi_k = chi(k);

        % estimate of acceleration and yaw rate for sample k > 2
        if k > 2

            h1 =  t(k) - t(k-1);
            h2 =  t(k-1) - t(k-2);
            alp = ( (h1+h2)/h1 )^2;

            if (h1+h2)/2 > 4  % do not compute a and r if mean ...
                sampling time > 4 s
               a_c = 0;
               r_c = 0;
            else
               a_c = ((1-alp)*U(k)  + alp*U(k-1)  - U(k-2))   / ...
                   ((1-alp)*h1+h2);
               r_c = ((1-alp)*chi(k)+ alp*chi(k-1)- chi(k-2)) / ...
                   ((1-alp)*h1+h2);
            end

        else  % zero for first two data points
            a_c = 0;
            r_c = 0;
        end

        % max values (saturation) to avoid estimates using wildpoints
        r_max = pi/180;
        if r_c > r_max
            r_c = r_max;
        elseif r < -r_max
            r_c = -r_max;
        end

        % max values
        a_max = 1;
        if a_c > a_max
            a_c = a_max;
```

```
138         elseif a_c < -a_max
139             a_c = -a_max;
140         end
141
142         % Corrector Kalman tilter (update states if new measurement)
143         z_k = [x_k y_k U_k chi_k]';
144         eps = z_k - H * x_hat;
145         eps(4) = wrapToPi(eps(4));
146         K = P * H' * inv(H*P*H' + R);
147         x_hat = x_hat + K * eps;
148         P = (eye(4) - K*H) * P * (eye(4) - K*H)' + K * R * K';
149
150         % Corrector kinematic observer (update states if new ...
                measurement)
151         x_prd   = x_prd   + h * K1 * (x_k - x_prd);
152         y_prd   = y_prd   + h * K2 * (y_k - y_prd);
153         U_prd   = U_prd   + h * K3 * (U_k - U_prd);
154         chi_prd = chi_prd + h * K4 * wrapToPi(chi_k - chi_prd);
155
156         if k < N
157             k = k + 1;
158             t_update = t(k);
159         end
160     end
161
162     % Kalman filter model
163     X_prd = [x_prd y_prd U_prd chi_prd ]';
164
165     f_prd = [ X_prd(3) * cos(X_prd(4))
166               X_prd(3) * sin(X_prd(4))
167               0
168               0 ];
169
170     F   = [ 0 0 cos(X_prd(4)) -X_prd(3) * sin(X_prd(4))
171             0 0 sin(X_prd(4))  X_prd(3) * cos(X_prd(4))
172             0 0 0 0
173             0 0 0 0 ];
174
175     PHI = eye(4) + h * F;
176
177     % Predictor Kalman filter (k+1)
178     x_hat = x_hat + h * ( f_prd + F * (x_hat - X_prd) + B * [a r]');
179     x_hat(4) = wrapToPi(x_hat(4));
180     P = PHI * P * PHI' + Q;
181
182     % Predictor kinematic observer (k+1)
183     x_prd   = x_prd   + h * U_k * cos(chi_prd);
184     y_prd   = y_prd   + h * U_k * sin(chi_prd);
185     U_prd   = U_prd   + h * a;
186     chi_prd = chi_prd + h * r;
187     chi_prd = wrapToPi(chi_prd);
188     a = a + (a_c-a) / T_a;
189     r = r + (r_c-r) / T_r;
```

```
190 end
191
192 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
193 % Motion prediction from time t(k_p)
194 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
195 x_p(1) = x(k_p);
196 y_p(1) = y(k_p);
197 U_p(1) = U(k_p);
198 chi_p(1) = chi(k_p);
199
200 dt = t(k_p) - t(k_p-1);
201 a  = ( U(k_p) - U(k_p-1) ) / dt;
202 r  = wrapToPi( chi(k_p) - chi(k_p-1) ) / dt;
203
204 for i = 1:tfinal/h_p
205     x_p(i+1)    = x_p(i)   + h_p * U_p(i) * cos( chi_p(i) );
206     y_p(i+1)    = y_p(i)   + h_p * U_p(i) * sin( chi_p(i) );
207     U_p(i+1)    = U_p(i)   + h_p * a;
208     chi_p(i+1)  = chi_p(i) + h_p * r;
209 end
210
211 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
212 % PLOT SIMULATION DATA: x = [x y U chi] AND PREDICTED SHIP
213 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
214 t_prd   = simdata(:,1);
215 x_prd   = simdata(:,2)/1000;
216 y_prd   = simdata(:,3)/1000;
217 U_prd   = simdata(:,4);
218 chi_prd = simdata(:,5);
219
220 x_hat   = simdata(:,6)/1000;
221 y_hat   = simdata(:,7)/1000;
222 U_hat   = simdata(:,8);
223 chi_hat = simdata(:,9);
224
225 a_hat   = simdata(:,10);
226 r_hat   = simdata(:,11);
227
228 x = x/1000;
229 y = y/1000;
230 x_p = x_p/1000;
231 y_p = y_p/1000;
232
233 figure(1)
234 xship = [-1  , 1/3, 1/3, 1, 1/3, 1/3,-1  ];   % draw ship
235 yship = [-1/3,-1/3,-1/3, 0, 1/3, 1/3, 1/3];
236
237 g1 = hgtransform;
238 g2 = hgtransform;
239 patch('XData',xship,'YData',yship,'FaceColor','black','Parent',g1);
240 patch('XData',xship,'YData',yship,'FaceColor','black','Parent',g2);
241
242 hold on
```

```
plot(y,x,'bo',y_prd,x_prd,'r','LineWidth',1);
plot(y_hat,x_hat,'c','LineWidth',1);
%plot(y(k_p),x(k_p),'rp','MarkerSize',30);
hold off; grid

set(gca,'fontsize',12)
xlabel('y [km]'); ylabel('x [km]')
title('xy-plot: Kinematic observer (red) and XKF (cyan)')

hold on
quiver(y_p(1),x_p(1),y_p(tfinal/h_p)-y_p(1),x_p(tfinal/h_p)-x_p(1),...
       1,'g','LineWidth',5)
g1.Matrix = makehgtform('translate',[y_p(1),x_p(1),0],...
      'scale',0.6,'zrotate',pi/2-chi_p(1));
g2.Matrix = ...
    makehgtform('translate',[y_p(tfinal/h_p),x_p(tfinal/h_p),0],...
      'scale',0.6,'zrotate',pi/2-chi_p(tfinal/h_p));
drawnow
hold off

figure(2)
subplot(611)
plot(t,(180/pi)*chi,'o', t_prd,(180/pi)*chi_prd,'r', ...
    t_prd,(180/pi)*chi_hat,'c'),grid,title('\chi [deg]');
set(gca,'fontsize',12)
subplot(612)
plot(t,U,'o', t_prd,U_prd,'r', t_prd,U_hat,'c'),grid,title('U [m/s]');
subplot(613)
plot(t,x,'o',t_prd,x_prd,'r',t_prd,x_hat,'c'),grid,title('x [km]');
set(gca,'fontsize',12)
subplot(614)
plot(t,y,'o',t_prd,y_prd,'r',t_prd,y_hat,'c'),grid,title('y [km]');
set(gca,'fontsize',12)
subplot(615)
plot(t_prd,a_hat,'b','LineWidth',1),grid,title('a [m/s^2]');
set(gca,'fontsize',12)
subplot(616)
plot(t_prd,r_hat*180/pi,'b','LineWidth',1),grid,title('r [deg/s]');
set(gca,'fontsize',12)
```

B.3 USV Simulator

The USV can be operated by a pilot and visualized in Unity by logging live AIS data. However, in many cases it is convenient to simulate the USV under pilot control. This can be done by assuming that the pilot and USV is a closed-loop dynamic system where two time constants in surge and yaw specifies the responses. The closed-loop dynamics of the USV is simulated using the following script:

```m
% USV test script
h = 0.1;                    % sampling time

% initial values
x = 0;
y = 0;
chi = 10 * pi/180;
U = 10;
x_k = [x y chi U]';

% joystick commands
U_c = 8;                     % SOG
chi_c = 30 * pi/180;         % COG

N = 1000;
for i = 1:N+1,
    simdata(i,:) = x_k';
    x_k = USV(x_k,U_c,chi_c,h);
end

t   = h * (0:N);
x   = simdata(:,1);
y   = simdata(:,2);
chi = simdata(:,3);
U   = simdata(:,4);

subplot(311)
plot(y,x); title('xy')
subplot(312)
plot(t,chi*180/pi); title('COG')
subplot(313)
plot(t,U);  title('SOG')
```

```m
% x_k = USV(x_k,U_k,chi_c,h)
function x_k = USV(x_k,U_c,chi_c,h)
    T_chi = 10;      % Time constant course angle
    T_U = 10;        % Time constant speed

    % Inputs
    x = x_k(1); y = x_k(2); chi = x_k(3); U = x_k(4);

    % Numerical integration of the USV closed-loop equations
    x = x + h * U * cos(chi);
    y = y + h * U * sin(chi);
    chi = chi + h/T_chi * WrapToPi(chi_c - chi);
    U = U + h/T_U * (U_c - U);

    % Outputs
    x_k = [x, y, chi, U]';
end
```

B.4 Visualization of Wave-Induced Ship Motions

In order to visualize the ship motions in waves as a function of varying sea states, a closed-form hydrodynamic model (Jensen et al., 2004) is implemented in Unity. The model uses the main characteristics of the ship, wave amplitude and direction to compute the heave, roll and pitch responses due to a regular wave.

```
% WAVERESPONSE estimates wave-induced ship motions using closed-form
% formulae for heave, roll and pitch.
%
% Ref: J. Juncher Jensen, A. E. Mansour and A. S. Olsen. Estimation of
% ship motions using closed-form expressions. Ocean Engineering 31,
% 2004, pp. 61-85

% wave data
a = 1.5;            % wave amplitude (m)
beta = 230;         % wave direction (deg), 180 deg is head sea
T_0  = 12;          % Wave peak period (s), w_0 = 2*pi/T_0

% main ship data
V = 5;              % Ship speed (m/s)
L = 40;             % Length of ship (m)
B = 6;              % Breadth of ship (m)
T = 4;              % Draught of ship (m)

% additional roll data
zeta_roll = 0.2;    % Relative damping factor in roll
T_roll = 6;         % Natural roll period (s)
GM_T = 1;           % Transverse metacentric height (m)
Cwp = 0.75;         % Waterplane area coefficient
Cb  = 0.65;         % Block coefficient
Delta = 1/3;        % 0 < Delta = bow/L < Cwp

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Hydrodynamic parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
g = 9.81;                       % acceleration of gravity (m/s^2)
rho = 1025;                     % density of water (kg/m^3)
nabla = Cb * L * B * T;         % volume displacement (m^3)
Awp = Cwp * L * B;              % waterline area (m^2)
w_0 = 2 * pi / T_0;             % Wave peak frequency (rad/s)
k = w_0^2/g;                    % wave number
beta = beta * (pi/180);
w_e = w_0 - k * V * cos(beta);  % frequency of encounter
Fn = V / sqrt(g*L);             % Froude number

% Heave and pitch
alpha = w_e/w_0;
k_e = abs(k * cos(beta));
KL = k_e * L/2;

kappa = exp(-k_e * T);
```

```
A = 2 * sin(k*B*alpha^2/2) * exp(-k*T*alpha^2);
f = sqrt( (1-k*T)^2  + (A^2/(k*B*alpha^3))^2 ) ;
F = kappa * f * (1/KL) * sin(KL);
G = kappa * f * (1/KL)^2 * (6/L) * ( sin(KL) - KL*cos(KL) );

% Roll
w_roll = 2*pi / T_roll;         % natural frequency
C44 = rho * g * nabla * GM_T;   % spring coefficient
M44 = C44/w_roll^2;
B44 = 2 * zeta_roll * w_roll * M44;  % damping coefficient
gamma = (Cwp - Delta)/(1-Delta);
M = sin(beta) * sqrt( B44 * rho*g^2/w_e );

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Outputs
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
z_max     = a * F                   % max heave amplitude (m)
phi_max   = (180/pi) * M / C44      % max roll angle (deg)
theta_max = (180/pi) * a * G        % max pitch angle (deg)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plots
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
t = 0:0.1:10;

% Oscillators (steady-state solutions) are based on the assumption ...
    that the
% wave frequency w_0 is far away from the natural frequencies w3, ...
    w4, w5
z = z_max * cos(w_e*t);
phi = phi_max * cos(w_e*t);
theta = theta_max * sin(w_e*t);

% The solution of the ODE is valid for all frequencies including ...
    the resonance
% where w_0 is equal to the natural frequency.
%     https://en.wikipedia.org/wiki/Harmonic_oscillator

% Heave: F0  = a F m w3^2
w3 = sqrt( g/(2*T) );
zeta3 = (1/(2*w3)) * (A^2/(k*B*alpha^3*w_0))*(g/(2*T));
Z3 = sqrt( (2*w3*zeta3)^2 + (1/w_e^2)*(w3^2-w_e^2)^2 );
eps3 = atan( 2*w_e*w3*zeta3/(w3^2-w_e^2) );
z_response = (a*F*w3^2/(Z3*w_e)) * cos(w_e*t+eps3);

% Pitch: F0  = a G m w5^2
w5 = sqrt( g/(2*T) );
zeta5 = (1/(2*w5)) * (A^2/(k*B*alpha^3*w_0))*(g/(2*T));
Z5 = sqrt( (2*w5*zeta5)^2 + (1/w_e^2)*(w5^2-w_e^2)^2 );
eps5 = atan( 2*w_e*w5*zeta5/(w5^2-w_e^2) );
theta_response = (180/pi) * (a*G*w5^2/(Z5*w_e)) * sin(w_e*t+eps5);

% Roll  F0 = (M/C44) m w4^2
```

```
w4 = 2*pi/T_roll;
zeta4 = zeta_roll;
Z4 = sqrt( (2*w4*zeta4)^2 + (1/w_e^2)*(w4^2-w_e^2)^2 );
eps4 = atan( 2*w_e*w4*zeta4/(w4^2-w_e^2) );
roll_response = (180/pi) * ((M/C44)*w4^2/(Z4*w_e)) * cos(w_e*t+eps4);

% Plots (blue is steady-state and red is ODE)
figure(gcf)
subplot(311)
plot(t,z,t,z_response),title('Heave (m)'),xlabel('time (s)'), grid
subplot(312)
plot(t,phi,t,roll_response),title('Roll (deg)'),xlabel('time (s)'), ...
    grid
subplot(313)
plot(t,theta,t,theta_response),title('Pitch (deg)'),xlabel('time ...
    (s)'),grid
```

Software Versions

C

The mixed-reality 3-D visualization software has been developed using the programs listed in Table C.1.

Table C.1: Software versions and description.

Software	Version no.	Developer	Description
3D Warehouse	NA	Google	Open library in which SketchUp users may upload and download 3-D models to share.
Blender	2.79b	Blender foundation	Open-source 3-D computer graphics software toolset used for creating animated films, visual effects, art, 3-D printed models, interactive 3-D applications and video games.
C# compiler	NA	Mono Develop	Unity build-in C# compiler.
Hydroform Ocean System	1.3.6	XIX Interactive	Plugin for generation of ocean waves
Matlab	R2018b 64 bit	MathWorks	Technical computing and simulation.
SketchUp Free	8	Google	3-D modeling computer program for architectural, interior design, landscape architecture, civil and mechanical engineering, film and video game design.
TerraLand	2.3	Terraunity	Plugin which contains multiple components for GIS data and coordinates to load and create photorealistic terrain from any part of the Earth.
Unity Asset Store	NA	Unity Technologies	Online marketplace for Unity users to sell project assets (artwork, code systems, audio, etc.) to each other.
Unity	2018.2.17f1	Unity Technologies	Cross-platform game engine for creation of 3-D games and simulations with API scripting in C#.
Windows	10	Microsoft	Operating system.

Video Links

D

The following Google Drive video link:

https://drive.google.com/drive/folders/1_zBol7ROfooj6nwx5ThbHuK_NP2aG5u3?usp=sharing

contains several MP4 videos, which demonstrate 3-D visualization of ships in a mixed-reality environment. The locations are the city of Trondheim, the river Nidelven and the fjord west of Trondheim.

Otter_mixed_reality_risk_index.mp4: Video showing the Maritime Robotics AS Otter USV moving up the Nidelven river in Trondheim. The Otter is simulated using a mathematical model of the vehicle (see Appendix B.3), which allows for evasive maneuvers to avoid collision.

USV_AISship_evasive_maneuver.mp4: Video showing a simulated USV approaching several AIS ships. The lines show the predicted motion of the AIS ships using the extended Kalman filter algorithm and the colors indicate the collision risk (green-yellow-red). This is a mixed-reality environment where the USV is simulated using a mathematical model of the vehicle (see Appendix B.3) and the AIS ships are overlaid the virtual world.

USV_in_waves.mp4: Video showing the wave-induced motions in heave, roll and pitch for a USV in waves using the closed-form hydrodynamic model of (Jensen et al., 2004), see Appendix B.4.

AIS_ship_overview.mp4: Bird perspective of AIS ships outside the city of Trondheim when changing the camera view.

Vehicle_Control_Station.mp4: Video showing the vehicle control station (see Fig. 1.3) for mixed-reality visualization of live AIS ships.

References

Automatic Identification System (2018). Wikipedia. Accessed 2018-11-15. **URL:** *https://en.wikipedia.org/wiki/Automatic_identification_system*

Balaji, S. and Murugaiyan, M. S. (2012). Waterfall vs. V-Model vs. Agile: A Comparative Study on SDLC. *International Journal of Information Technology and Business Management*, 2(1): 26–30.

Bhat, S. P. and Bernstein, D. S. (2000). A Topological Obstruction to Continuous Global Stabilization of Rotational Motion and the Unwinding Phenomenon. *Systems and Control Letters*, 39(1): 63–70. doi: 10.1016/S0167-6911(99)00090-0.

COLREGS (1972). International Regulations for Preventing Collisions at Sea - Articles of the Convention on the International Regulations for Preventing Collisions at Sea. Lloyd's Register.

Farrell, J. A. (2008). *Aided Navigation: GPS with High Rate Sensors*. McGraw-Hill.

Fossen, S., Bye, R. T., and Osen, O. (2018). Visualization and Collision Risk Assessment of Real Ships in a Mixed Reality Environment using Live Automatic Identification System (AIS) Data. In *Proc. of the 2nd IEEE European Conference on Computers & Computing (EECS'18)*, 20-22 December, Bern.

Fossen, S. and Fossen, T. I. (2018a). eXogenous Kalman Filter (XKF) for Visualization and Motion Prediction of Ships using Live Automatic Identification System (AIS) Data. *Modeling, Identification and Control*, 39(4): 233–244. doi: 10.4173/mic.2018.4.1.

Fossen, S. and Fossen, T. I. (2018b). Extended Kalman Filter Design and Motion Prediction of Ships using Live Automatic Identification System (AIS) Data. In *Proc. of the 2nd IEEE European Conference on Computers & Computing (EECS'18)*, 20-22 December, Bern.

Fossen, T. I. (2011). *Handbook of Marine Craft Hydrodynamics and Motion Control*. Wiley. doi: 10.1002/9781119994138.

Fu, H. F. H., Liu, S. L. S., and Sun, F. S. F. (2010). Ship Motion Prediction based on AGA-LSSVM. In *IEEE International Conference on Mechatronics and Automation (ICMA'10)*, pp. 202–206. doi: 10.1109/ICMA.2010.5589093.

Gelb, A. (1974). *Applied Optimal Estimation*. MIT Press.

Holsopple, J., Sudit, M., Nusinov, M., Liu, D. F., Du, H., and Yang, S. J. (2010). Enhancing Situation Awareness via Automated Situation Assessment. *IEEE Communications Magazine*, 48(3).

Jaskolski, K. (2017). Automatic Identification System (AIS) Dynamic Data Estimation Based on Discrete Kalman Filter (KF) Algorithm. *Scientific Journal of Polish Naval Academy*, 211(4): 71–87. doi: 10.5604/01.3001.0010.6747.

Jensen, J. J., Mansour, A. E., and Olsen, A. S. (2004). Estimation of Ship Motions using Closed-Form Expressions. *Ocean Engineering*, 31: 61–85.

Jiang, S., Jin, H., and Wei, F. (2013). LS-SVM Application for Ship Course Model Predictive Control. In *IEEE International Conference on Mechatronics and Automation (ICMA'13)*, pp. 1615–1619. doi: 10.1109/ICMA.2013.6618156.

Johansen, T. A. and Fossen, T. I. (2017). The eXogenous Kalman Filter (XKF). *International Journal of Control*, 90(2): 161–167. doi: 10.1080/00207179.2016.1172390.

Johansen, T. A., Perez, T., and Cristofaro, A. (2016). Ship Collision Avoidance and COLREGS Compliance using Simulation-Based Control Behavior Selection with Predictive Hazard Assessment. *IEEE Transactions on Intelligent Transportation Systems*, 17(4): 3407–3422.

Khalil, H. K. (2014). *Nonlinear Systems*. Pearson, 3rd edition.

Kongsberg (2018). Autonomous ship project, key facts about YARA Birkeland. Accessed 2018-11-18.
URL: *https://www.km.kongsberg.com*

Lin, Z., Yang, Q., Guo, Z., and Li, J. (2011). An Improved Autoregressive Method with Kalman Filtering Theory for Vessel: Motion Predication. *International Journal of Intelligent Systems*, 4(4): 11–18.

Mazzarella, F., Arguedas, V. F., and M.Vespe (2015). Knowledge-based Vessel Position Prediction using Historical AIS Data. In *Sensor Data Fusion: Trends, Solutions, Applications*, pp. 1–5.

Milgram, P. and Kishino, F. (1994). A Taxonomy of Mixed Reality Visual Displays. *IEICE Transactions on Information Systems*, E77-D(12).

MSQ (2018). Maritime Safety Queensland. Accessed 2018-11-27.
URL: *https://www.msq.qld.gov.au/Safety/Situational-awareness*

MSS (2004). Marine Systems Simulator. Accessed 2018-10-31.
URL: *https://github.com/cybergalactic/MSS*

NMEA (2018). Standard. Accessed 2018-10-10.
URL: *https://www.nmea.org/content/nmea*$_{s}tandards/nmea\_0183\_v\_410.asp$

Perera, L. P. and Soares, C. G. (2010). Ocean Vessel Trajectory Estimation and Prediction Based on Extended Kalman Filter. In *2nd International Conference on Adaptive and Self-adaptive Systems and Applications*, pp. 14–20.

Quora (2018). The Difference Between Virtual Reality, Augmented Reality And Mixed Reality. Accessed 2018-10-05.
URL: *https://www.forbes.com/sites/quora/2018/02/02/the-difference-between-virtual-reality-augmented-reality-and-mixed-reality/1aa88f932d07*

Ristic, B., Scala, B. L., Morelande, M., and Gordon, N. (2008). Statistical Analysis of Motion Patterns in AIS Data: Anomaly Detection and Motion Prediction. In *International conference on Information Fusion*, pp. 40–46.

Sapankevych, N. and Sankar, R. (2009). Time Series Prediction using Support Vector Machines: A Survey. *IEEE Computational Intelligence Magazine*, 4(2): 24–38. doi: 10.1109/MCI.2009.932254.

Sommerville, I. (2016). *Software Engineering*. Pearson Education Limited, 10th edition.

Triantafyllou, M. S. and Bodson, M. (1982). Real-Time Prediction of Marine Vessel Motions Using Kalman Filtering Techniques. In *Annual Offshore Technology Conference*.

Unity (2018). The Unity Game Engine. Accessed 2018-09-17.
URL: *https://unity3d.com*

US Coast Guard (2018). Navigation Centre. Accessed 2018-11-12. doi: 10.1016/0003-4916(63)90068-X.
URL: *https://www.navcen.uscg.gov/?pageName=AISMessages*

Xiao, Z., Ponnambalam, L., Fu, X., and Zhang, W. (2017). Maritime Traffic Probabilistic Forecasting Based on Vessels' Waterway Patterns and Motion Behaviors. *IEEE Transportation Intelligent Transportation Systems*, 18(11): 3122–3134. doi: 10.1109/TITS.2017.2681810.

Xu, X., Geng, X., and Wen., Y. (2016). Modeling of Ship Collision Risk Index Based on Complex Plane and Its Realization. *TRANSNAV, The International Journal on Marine Navigation and Safety of Sea Transportation*, 10(2): 251–256.

Yin, J. and Zou, Z. (2011). A Combined Modular Parametric and Non-parametric Method for Planar Ship Motion's On-line Prediction. In *Lecture Notes in Informatics in Control, Automation and Robotics*, volume 1, pp. 17–24.

Yin, J. C., Zou, Z. J., and Xu, F. (2013). On-line Prediction of Ship Roll Motion during Maneuvering using Sequential Learning RBF Neural Networks. *Ocean Engineering*, 61(139–147). doi: 10.1016/j.oceaneng.2013.01.005.

Yumori, I. (1981). Real Time Prediction of Ship Response to Ocean Waves Using Time Series Analysis. *Ocean*, 81(1082–1089).