Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Runar André Olsen

Sensor fusion of radar data with deep learning based detection and tracking of ships in camera images

Master's thesis in Cybernetics and Robotics
Supervisor: Edmund Førland Brekke
June 2019

NTNU
Norwegian University of
Science and Technology

Runar André Olsen

Sensor fusion of radar data with deep learning based detection and tracking of ships in camera images

Master's thesis in Cybernetics and Robotics
Supervisor: Edmund Førland Brekke; Arild Hepsø, Kenan Trnka
June 2019

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Dedicated to my wonderful and loving children,
which will forever be the cornerstones of my life

Preface

This thesis marks the conclusion of the author's 2 year Master's degree in Cybernetics and Robotics at Department of Engineering Cybernetics at Norwegian University of Science and Technology (NTNU). The thesis builds upon the work done by the author in the autumn semester as part of the final year specialization project course TTK4551 [1].

Throughout the thesis an implementation of sensor fusion between track associated targets from RADAR and detections and tracks in cameras are explained. The implementation is written in Python and uses the Robot Operating System (ROS) framework [2]. The author have created some ROS packages while others were freely available online. The backbones for detection (YOLOv3 [3]) and tracking (Re3 [4]) are not the author's work, but much work has been done adapting the modules for the given task. Re3 had no ROS node available and this is built by the author.

The RADAR tracker is courtesy of the Autosea project at NTNU, developed by PhD students [5][6]. The Integrated Probabilistic Data Association (IPDA) based tracker, which is performing all necessary handling of the RADAR data, is used to initialize tracks and further as an advanced sensor measurement for the RADAR in the following fusion implementation.

Most of the work and demonstrations have been done on data recorded in the autumn of 2018 as part of Autosea experiments [5]. RADAR, INS and AIS data was recorded in rosbags while images were recorded separately and stored to disk. The Linux driver for the Ladybug camera used had a bug which prevented recording of images directly in ROS. Time synchronization is therefore not optimal between the images and rostopics. NTNU have provided the Ladybug camera system as well as a powerful laptop which has been used to process the data and run the algorithms.

I would like to thank my supervisor Edmund F. Brekke for all support and guidance through this immense and challenging field of research. I would also like to thank my co-supervisors for support and help to facilitate the experiments.
Scheduling meetings this year have not always been easy because of my family situation, but they have been vital for pointing the project towards a working fusion algorithm.

Runar André Olsen

Trondheim, June 10, 2019

Abstract

Deep learning Convolutional Neural Network (CNN) based detection and tracking methods for images and videos are currently the highest performing methods available [7]. Much research is being invested in these methods, which are predicted to be vital in the upcoming autonomous revolution. Out at sea in a collision avoidance (COLAV) scenario cameras can provide valuable information to aid other sensors such as RADAR and LiDAR. The cameras have excellent angular resolution and update frequency, but lacks any information on depth or distance.

To reliably achieve robust and safe autonomous vessels it is necessary to have multiple sensors which reduce the likelihood of erroneous or missed detections. Using different types of sensors one can combine the best characteristics into a sensory system better than any single sensor could provide.

This thesis describes a modular sensor fusion pipeline, built using ROS, for combining RADAR tracks with image data captured using omnidirectional ($360°$) camera systems. The Autosea RADAR tracker used in this thesis performs all necessary handling of the RADAR data. It outputs position and uncertainty as well as an unique ID for each tracked target. All of this information is used to initialize the author's Extended Kalman Filter (EKF) based tracker which takes in measurements from the RADAR and camera. Received positions from the AIS system is also implemented as possible measurements for when they are available. Other sensors outputting measurements in Cartesian coordinates are easily added in the algorithm.

Image data in the direction of the detected vessel is processed through a CNN based object detector. The object detector is complimented by a CNN based object tracker, and the tracker is initialized from the detections made in the object detector. Measurements from both the tracker and detector is merged with the RADAR data in the EKF based sensor fusion algorithm.

The tests demonstrates that the EKF is able to track an already initiated track quite well using only camera data, especially if the ASV performs some maneuvers. The cameras further help reduce the uncertainty of the targets position from RADAR alone. In the tests done, the visual object tracker provide limited enhancement on the object detector due to serialization on GPU usage and imperfect image stabilization. Separate GPUs and more stringent synchronization between sensors will be necessary for better results.

The overall aim of the Autosea project and this thesis is a robust, real-time COLAV system to enhance safety at sea. There are no "one true" answer on how the problem should be solved and this thesis takes a practical and straightforward approach, demonstrating the possibilities in active-passive sensor fusion.

Sammendrag

De metodene med høyest ytelse på objektgjenkjennelse og -sporing i bilder og videoer er nå begge basert på dyp læring og konvolusjonelle nevrale nettverk CNN [7]. Mye forskning blir investert i disse metodene, som antas å være avgjørende i den kommende autonome revolusjonen. Ute til sjøs i et kollisjonsunngålese (COLAV) scenario kan kameraer gi verdifull informasjon ved å komplimentere andre sensorer som RADAR og LiDAR. Kameraene har utmerket vinkel-oppløsning og oppdateringsfrekvens, men mangler all informasjon om dybde eller avstand.

For å oppnå robuste og sikre autonome fartøyer er det nødvendig å ha flere sensorer som reduserer sannsynligheten for feilaktige eller manglende detekteringer. Ved å bruke forskjellige typer sensorer kan man kombinere de beste egenskapene til et sensorsystem bedre enn noen enkeltsensor kan gi.

Denne oppgaven beskriver en modulær sensorfusjonsmetode, bygd ved bruk av ROS, for å kombinere RADAR spor med bildedata sanket med omnidireksjonelle ($360°$) kamerasystemer. RADAR-trackeren fra Autosea prosjektet som brukes i denne oppgaven, utfører all nødvendig behandling av RADAR-dataene. Den gir ut posisjon og usikkerhet samt en unik ID for hvert sporet object. All denne informasjonen er brukt for å initialisere forfatterens Extended Kalman Filter (EKF)-baserte tracker som tar inn målinger fra RADAR og kamera. Mottatte posisjoner fra AIS systemet tas også inn som målinger hvis tilgjengelige. Andre sensorer som gir ut målinger i Kartesiske koordinater kan enkelt legges til i algoritmen.

Bildedata i retning av det detekterte fartøyet behandles gjennom en CNN-basert objektdetektor. Objektdetektoren er komplementert av en CNN-basert objektracker, og trackeren initialiseres fra deteksjonene som er gjort i objektdetektoren. Målinger fra både trackeren og detektoren slås sammen med RADAR-dataene i den EKF-baserte sensorfusjonsalgoritmen.

Testene demonstrerer at den EKF-baserte algoritmen klarer å følge et allerede initialisert spor ganske bra ved bare bruk av kameradata, spesielt hvis ASVen utfører noen manøvre. Kameraene hjelper videre til å redusere usikkerheten på det sporede objektet sammenlignet med RADAR alene. I testene som er gjort, gir den visuelle trackeren begrenset forbedring sammenlignet med objekt detektoren alene. Årsaken er serialisering på GPU bruk og svak bildestabilisering. Separat GPU til trackeren og strengere fokus på synkronisering mellom sensorer er nødvendig for gode resultater.

Det overordnede målet med Autosea-prosjektet og denne oppgaven er et robust, sanntids COLAV system for å øke sikkerheten til sjøs. Det er ikke noe enkelt korrekt svar på hvordan problemet skal løses og denne oppgaven tar en praktisk og ukomplisert tilnærming, som demonstrerer mulighetene i aktiv-passiv sensorfusjon.

Contents

Acronyms

Re^3 Real-time Recurrent Regression. v, vi, 4, 12, 19, 23, 24, 27, 30, 31, 35–37, 39, 40, 42, 45–49, 55–57, 61–66, 73–77, 79, 80, 84–88, 91–93, 107–112

1PPS 1 Pulse Per Second. 42, 53, 93

2D 2 Dimensional. 43

3D 3 Dimensional. 16, 43, 44

AI Artificial Intelligence. 1, 3, 19

AIS Automatic Identification System. v, 2, 5, 7, 33, 39, 48, 51, 64, 66, 78, 93

API Application Programming Interface. 12, 15, 16, 53

ASV Autonomous Surface Vessel. vi, 1–3, 24, 27, 34–36, 38, 40, 42, 45, 46, 51, 61, 66, 67, 78, 79, 88, 91, 99–102

BODY Body-fixed. 5–7, 10, 39, 42, 45

CNN Convolutional Neural Network. iii–v, 1, 10, 20–24, 27, 55, 61, 64, 91

COCO Common Objects in COntext. 21, 57

COLAV collision avoidance. iii, iv, 1, 2, 27, 28, 42, 89, 92

COR Center Of Rotation. 10

CPU Central Processing Unit. 21, 48, 55, 64

CT Coordinated Turn. 33, 93

CUDA Compute Unified Device Architecture. 21, 55

CV Constant Velocity. 35, 93

DCF Discriminative Correlation Filters. 24

ECEF Earth Centered Earth Fixed. 5, 6

EKF Extended Kalman Filter. iii, iv, vi, 3, 32–35, 39, 41, 47, 48, 64, 65, 88, 91–93

ENU East North Up. 6

EO Electro Optical. 2–4, 10, 39, 48

FM Frequency Modulated. 8

FMCW Frequency Modulated Continuous Wave. 8

FN False Negative. 30

FOV Field Of View. 10, 13, 15, 38, 44–46, 88, 93

FP False Positive. 29, 30

GES Globally Exponentially Stable. 32

GFLOPS Giga Floating Point operations per Second. 55, 56

GNSS Global Navigation Satellite System. 7, 8, 10, 36, 42, 53, 54, 93

GPU Graphics Processing Unit. 21–23, 36, 40, 48, 55–57, 61, 62, 64, 91

IMO International Maritime Organization. 7

IMU Inertial Measurement Unit. 8, 93

INS Inertial Navigation System. v, vi, 2, 8, 10, 24, 31, 36, 41, 42, 44, 47, 51, 64, 78, 88, 92, 93

IoU Intersection over Union. 3, 30, 31, 39, 47, 62, 63, 92

IPDA Integrated Probabilistic Data Association. i, 2, 29, 36, 41, 91

IR infrared. 2, 3

JIPDA Joint Integrated Probabilistic Data Association. 3, 29, 92, 93

JPDA Joint Probabilistic Data Association. 29

KF Kalman Filter. vi, 8, 31–34

LiDAR Light Detection And Ranging. 1–3, 7, 27, 39, 48, 89, 93

LOS Line of Sight. 30

LSTM Long Short-Term Memory. v, 20, 21, 23, 47, 61

ML Machine Learning. 19

NED North East Down. 5–7, 15, 34, 36, 39, 40, 45

NM Nautical Miles. 8

NMEA National Marine Electronics Association. 53

NTNU Norwegian University of Science and Technology. i, 2, 51

PDAF Probabilistic Data Association Filter. 29, 41

PDF Probability Density Function. v, 28

PF Particle Filter. 32

RADAR RAdio Detection And Ranging. v, vi, 1–4, 7–10, 12, 27, 28, 31, 33–37, 39–42, 44–49, 51, 56, 62, 64, 65, 68–76, 78, 80–88, 91–93

RGB Red Green Blue. 20

RNN Recurrent Neural Network. 20

ROS Robot Operating System. i, iii, vi, 12, 15, 16, 34–36, 40–42, 45, 46, 51, 53, 54, 89

RPAS Remotely Piloted Aircraft Systems. 1

RPM Rounds Per Minute. 8

RS-232 Electronic Industries Association (EIA) Recommended Standard 232. 53

RTK Real-Time Kinematics. 10

SLAM Simultaneous Localization And Mapping. 27

SNR Signal to Noise Ratio. 30

SSD Single Shot Detector. 21

TN True Negative. 30

TP True Positive. 29, 30

TTL Transistor–Transistor Logic. 53

UAV Unmanned Aerial Vehicle. 1

UKF Unscented Kalman Filter. 32

USB Universal Serial Bus. 51–53

VHF Very High Frequency. 7

VOT2018 The Visual Object Tracking Challenge. 24, 25, 27

WGS84 World Geodetic System of 1984. 5, 36

XKF eXogenous Kalman Filter. 32

YOLO You Only Look Once. v, vi, 4, 12, 19, 21–23, 27, 30, 31, 35–37, 39, 40, 42, 45–49, 55–66, 68, 71–74, 78, 82–85, 88, 91, 92, 103–106, 110–112

Chapter 1

Introduction

There is a large and increasing interest in autonomous vehicles, both on land, in the air and at sea. Heavy investments are being made to increase safety and reduce demand for personnel. The main investments and research has, especially in the past, been from the military where a reduction of risk to own personnel and civilians have been in focus, as well as more effective weapon systems. In contrast we now see a large private and commercial market for remote controlled and autonomous vehicles. Especially in the air where Remotely Piloted Aircraft Systems (RPAS) or Unmanned Aerial Vehicle (UAV) often called simply drones, have hit the mass market. The largest industry for autonomous vehicles are now the car industry where autonomous car research is a very active field and is receiving large funds. Also in the maritime industry we are seeing a large interest in automating certain shipping tasks as well as automated and autonomous ferries. The expensive and potentially dangerous systems in the commercial and military market have very high demands concerning safety and reliability.

A key technology for such autonomous vessels is a good understanding of the surrounding environment, which again opens up for a reliable and robust collision avoidance (COLAV) system. To achieve this high robustness, several sensors and advanced computer systems are necessary. Sensor fusion can help reduce the uncertainty that follows from any sensor measurement, and provide more robust sensory information than any standalone sensor. Different sensor fusion methods can therefore be valuable in all types of products containing several sensors. This applies to both high end military and commercial products as well as mass market products.

The new massive improvements done on the area of Artificial Intelligence (AI) in recent years, have opened up for practical use of camera systems as sensors for object detection and tracking. This thesis investigates the use of new state-of-the-art, deep learning based, object detection and tracking methods in image data together with classical sensors such as RADAR and Light Detection And Ranging (LiDAR).

1.1 Background and Motivation

This thesis builds upon the author's specialization project report "Deep learning based detection and tracking of ships in camera data for sensor fusion with radar" from the autumn semester of 2018 [1]. The report considers the use of deep learning CNN based object detetctors and -trackers for use onboard an ASV. An ASV is an autonomous marine surface vehicle, e.g. ship, boat, sailboat, kayak etc.

For autonomous vessels it is of utmost importance to have reliable and updated sensory information for both the vessel itself as well as the surroundings. In ships and boats the dynamics of the autonomous

vessel itself is for the most part a solved problem using state of the art solutions as of 2018 [8]. Several sensors are used to know the position, speed and attitude of the vessel with great confidence. These sensors might be fused together to support each other, for example Inertial Navigation System (INS), or be separated. Offshore ships such as diving vessels need to have several separated systems so that any type of sensor or system failure will not make the vessel suddenly start moving [9].

In contrast other vessels might have unexpected maneuvers and are not easily modelled with sufficient certainty over longer time steps. Usually the other vessel is not known and properties such as size, speed, maneuverability and vessel type may only be estimated by various sensors such as RAdio Detection And Ranging (RADAR), Light Detection And Ranging (LiDAR) and Automatic Identification System (AIS). Various optical sensors such as regular Electro Optical (EO) cameras detecting visible light (daylight) and infrared (IR) cameras give detailed information about the object, but need interpretation to be useful to computers.

To be able to reliably detect and track other vessels it is necessary to have a very good and accurate understanding of own position, speed and attitude. The same is true for information about timing and synchronization between sensors. The need for accurate sensors, and especially good synchronization, is demonstrated in the thesis. The data recorded under real-time scenarios on the ASV, and used throughout this thesis, is to a large degree affected by motion of the ASV.

The problem to be investigated in this thesis is sensor fusion of active and passive sensors, i.e. RADAR and digital daylight cameras. The task is part of a collaboration project called Autosea between NTNU, Maritime Robotics, DNV GL and Kongsberg. The Autosea project aims to solve some of the problems concerning sensor fusion and collision avoidance for autonomous surface vehicles.

Many types of sensors are possible to use in COLAV scenarios. The different sensors all have strengths and weaknesses and must be chosen and adapted for the relevant scenario. RADAR have long range and good reliability, but low update frequency and relatively little details. The cameras give detailed information for objects close by, but give very limited depth information. This is even true for stereo cameras when the objects are far away. If the camera could have been mounted high above the sea surface, on a large ship, the images would have given distance information based on the vertical angle to the tracked vessel. Another obvious limitation is that EO cameras rely on good light conditions to give usable data. Since the cameras are passive sensors they can have extremely high update frequency and are not affected by the limited range of for instance LiDAR. If needed a stabilized camera with a good zoom lens could provide good details at long ranges, this is however not exploited in the setup.

Fusion between LiDAR and camera have been the topic of the Master's thesis of Kamsvåg [10]. The Master's thesis by Helgesen [11] have sensor fusion between RADAR, LiDAR, IR and EO camera as topic. Both of which this thesis finds inspiration. LiDAR have greater details and higher update frequency, but have shorter range and is more affected by atmospheric conditions such as rain, snow or fog compared to RADAR. IR cameras have, under many circumstances, significant better signal to noise ratio and they work even at night compared to EO cameras. However, at the moment, IR cameras have much lower resolution and a much higher price tag.

1.2 Implementation

In this thesis we suggest a novel approach for sensor fusion using new state-of-the-art deep learning based methods for tracking and detection in camera data together with the classical active sensors such as RADAR. See figure 5.1 for an overview of the implementation. The fusion algorithm is built around the IPDA based RADAR tracker already developed in the Autosea project [5][6]. The RADAR tracker handles all processing of RADAR data including clustering of detections, land-detection removal and

tracking. The tracking includes initialization and termination of identified tracks as well as estimation of uncertainty.

The tracks initialized and tracked in the RADAR tracker are used to initialize and further as measurements in the author's EKF based sensor fusion pipeline. Image data which corresponds to the same target, based on the angle, is extracted and fed to the image detector and tracker. The output from the detector is further used to initialize the image tracker and the output from both modules are taken in as measurements to the sensor fusion pipeline.

The advantage of using an image tracker is that one has continuous position estimates for the tracked object compared to a detector which may or may not detect the object at each frame. The trackers are generally also faster than detectors and have lower computational demands. This may be utilized to provide better than real-time tracking speed for the image data. The trackers will however have lower confidence about the tracked object, and may drift away from the object if not monitored. A track-before-detect inspired method is therefore implemented on the visual tracker together with several underlying restrictions.

Since the visual modules does not provide uncertainty information regarding position, classical track-to-track fusion of estimates is difficult [12][13]. In this thesis a novel and, based on the information available, original method for combining passive and active sensor information is described. It is based on the Intersection over Union (IoU) [14] method for evaluating how well overlapping data such as bounding boxes align.

The Artificial Intelligence (AI) methods used are demonstrated to be robust against changes in lighting conditions, appearance and size [4][15]. The underlying deep learning networks can furthermore be optimized on the types of boats and ships the ASV will encounter in the area. This training to optimize deep learning networks are referred to as transfer learning and will be detailed in chapter 3.

The suggested pipeline is shown in figure 5.1 and will be presented throughout the thesis.

1.3 Previous work and inspiration

Hermann et al. [16] have described a tracking system incorporating RADAR and camera data for a maritime multiple obstacle tracker. They have demonstrated an significant improvement of the tracking performance by fusion of RADAR and camera. The obstacle detection in image data uses an effective and simple filtering and thresholding of the raw image. They also extract the horizon using filtered canny edge detections.

Elkins et al. [17] proposes a framework for autonomous maritime navigation (AMN) using several different sensors including RADAR, LiDAR and omnidirectional camera. They provide insight into the core ideas for fusion. Although the AMN project have performed several successful demonstrations they conclude that there are still big limitations of the system.

Kamsvåg [10] describes fusion of passive and active sensors, i.e. camera images and LiDAR. The implementation did not use any image tracker, but proposed to fuse the measurements from an image detector (Faster R-CNN) with the clustered LiDAR measurements directly using a Joint Integrated Probabilistic Data Association (JIPDA) based tracker. Kamsvåg did not get good detection performance from the camera data for distances beyond about 20m, and did neither complete the fusion by using the camera measurements to update the target state.

Helgesen [11] describing sensor fusion between RADAR, LiDAR, IR camera and EO camera use a similar JIPDA based tracker as Kamsvåg. The data set used for the experiments was recorded from a slightly higher stationary point on land and are not directly transferable to this thesis. Helgesen did not

find that the EO cameras gave the expected positive effect in the sensor fusion. The RADAR data was, as expected, very reliable for regular boats, but had difficulties detecting smaller vessels such as kayaks. The RADAR used, which is the same as the one used in this thesis, did not provide good separation between objects close to each other.

1.4 Thesis Outline

The rest of the thesis will be structured in the following matter:

- Chapter 2 is used to provide more detailed information about the different sensors and processes. What types of sensors are available and some theory behind them.
- Chapter 3 gives an introduction to the YOLO detector and the Re^3 tracker as well as the underlying deep learning methods used. New and promising methods are also introduced.
- Chapter 4 builds up the background for target detection and tracking using both passive and active sensors. The chapter also introduces sensor fusion and data association together with details of the implemented pipeline.
- Chapter 5 explains the suggested tracking pipeline for sensor fusion with omnidirectional (360°) camera and RADAR. Implementation details on both the visual detector and the visual tracker is explained together with an overview of the Autosea tracker.
- Chapter 6 is used to explain the experiments and tests that have been done to verify the possibilities and limitations of the methods and modules in the suggested pipeline. Experiment details and challenges are explained.
- Chapter 7 includes an overview of the status of the pipeline. It gathers the main takeaways in the thesis and also suggests future work that has not been completed within this project.
- Appendix A includes complimentary results and data from the tests and examples throughout the thesis.

Chapter 2

Sensors

In modern vessels there are many sensors which give valuable information to the helmsman and potential guidance and navigation systems. Most sensors are standalone, but advanced sensors may utilize sensor fusion for increased accuracy and robustness. If communication of data such as AIS is installed, position, speed and heading can also be transmitted to other vessels.

The sensors can be classified into different categories based on their properties and their usage. Firstly the sensors can be divided between active and passive sensors, where passive sensors only observe the current state, while active sensors include transmitters. Secondly one can divide the sensors into whether they observe the state of the own vessel or if they observe the surroundings. To reliably observe, detect and understand the surrounding state it is of upmost importance to know one's own state accurately.

2.1 Reference Frames

The different sensors can have measurements and outputs referenced to different reference frames. Transformations between different frames are needed to combine sensors. The most common frames used for surface vessels are [8]:

ECEF Earth Centered Earth Fixed (ECEF) where the center of the earth is the reference. This is commonly used in maritime navigation where World Geodetic System of 1984 (WGS84) is used and coordinates are given in degrees as Latitude (north/south) and Longitude (east/west). The zero point for Latitude is the equator while the zero point for Longitude is at the Greenwich Royal Observatory in London. Elevation is given referenced to the WGS84 ellipsoid of the mean sea level across the globe. Local Geoide models which are much more detailed are used if the height information is important.

NED North East Down (NED) is a local coordinate system which can be considered a flat tangent plane on the face of the earth. It might also be referenced to the WGS84 ellipsoid height, but the origin will depend on the chosen coordinate system.

BODY The Body-fixed (BODY) coordinate frame is fixed to the vessel or the sensor and moves with the vessel. The Body-fixed (BODY) coordinates are given as Forward-Starboard-Down which is comparable to North East Down (NED) with a transformation. Each sensor will have its own BODY frame and must be aligned with the vessel BODY frame for further transformation.

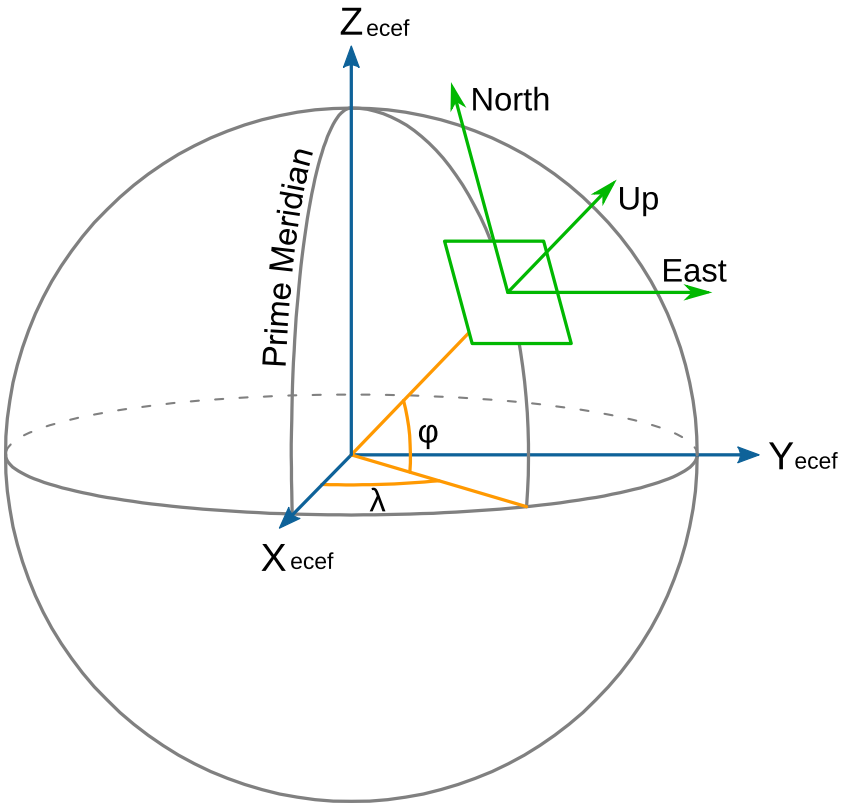


Figure 2.1: East North Up (ENU) is similar to NED but is rotated so that up is positive. Here shown compared to Earth Centered Earth Fixed (ECEF). Wikimedia Commons.

Rotation and rigid body transformations are efficiently calculated using matrices belong to special distance preserving Lie groups called Special Orthogonal Group 3 $SO(3)$ and Special Euclidean Group 3 $SE(3)$. They have the useful property that the inverse of the matrix is it's transpose, which is much easier to compute.

$$SO(3) = \{\mathbf{R} \in \mathbb{R}^{3\times3} \mid \mathbf{R}\mathbf{R}^T = 1, det\mathbf{R} = 1\}$$
$$SE(3) = \left\{ \mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \in \mathbb{R}^{4\times4} \;\middle|\; \mathbf{R} = SO(3), \mathbf{t} \in \mathbb{R}^{3\times3} \right\}$$

The principal rotations ϕ, θ, ψ around the single axis x, y, z are given in the designated coordinate system as:

$$\mathbf{R}_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi & -s\phi \\ 0 & s\phi & c\phi \end{bmatrix}, \mathbf{R}_y(\theta) = \begin{bmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{bmatrix}, \mathbf{R}_z(\psi) = \begin{bmatrix} c\psi & -s\psi & 0 \\ s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

x, y, z indicates North,East,Down in NED and Forward,Starboard,Down in BODY. s and c is short for Sine() and Cosine() respectively. ϕ, θ and ψ are used for the angles Roll, Pitch and Yaw/Heading.

Rotations from BODY to NED can be performed using the following rotation matrix which is composed of all 3 principal rotations (x-y-z). Other variants of the rotation matrix is used in other areas such as

robotics where (z-x-z) is common. Superscript indicated the expressed coordinate system, n indicates NED and b indicates BODY. Subscript indicates the coordinate system it is referenced from.

$$\mathbf{R}_b^n(\Phi) = \mathbf{R}_{x/b}^n(\phi)\mathbf{R}_{y/b}^n(\theta)\mathbf{R}_{z/b}^n(\psi) \tag{2.2}$$

$$\mathbf{R}_b^n(\Phi) = \begin{bmatrix} c\psi c\theta & -s\psi c\phi + c\psi s\theta s\phi & s\psi s\phi + c\psi c\phi s\theta \\ s\psi c\theta & c\psi c\phi + s\phi s\theta s\psi & -c\psi s\theta + s\theta s\psi c\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \tag{2.3}$$

With the correct rotation $\mathbf{R}$ and translation $\mathbf{t}$, any point p in the BODY coordinate frame can be transformed to the NED coordinate frame.

$$p^n = \mathbf{T}_b^n p^b$$

2.2 Automatic Identification System (AIS)

An important safety feature that has been introduced to the maritime industry in recent years is the AIS system. The AIS relies on Global Navigation Satellite System (GNSS) position and sometimes also heading on board the other vessel and Very High Frequency (VHF) radio for transmitting and receiving the information between the vessels.

The AIS provides the position, speed and heading of the vessel which can be used together with other sensors as a "ground truth" provided that the AIS is correctly set up. International Maritime Organization (IMO) defines the AIS standard [18] and requires the AIS message transmitted, depending on vessel type and current speed, at a constant rate between 2 seconds and 3 minutes. The delay between the received GNSS position of the transmitting ship and time of transmission/reception of the AIS message is unknown, but relatively small ($<$ 1 second). This uncertainty in time as well as unknown accuracy of the transmitted signal is the biggest issue with using the AIS as a measurement input.

All ships above 500 gross tonnage and all passenger ships of any size should have installed AIS [19]. Most new boats also install AIS as this is a cheap and effective safety feature, ensuring visibility to others. Smaller ships and other objects at sea may not have AIS installed or it can be turned off or be inoperative and one can therefore not rely solely on the system.

2.3 Active Sensors

Active sensors include a transmitter which sends out a signal that is reflected by the surrounding environment and received again by the sensor. Common active sensors are RADAR and LiDAR which radiate electro-magnetic signals in the form of radio and light respectively. Also the many variants of echosounders used for measuring water depth etc. are active, using sound-waves. The same principle is used in air for close range measurements using ultrasound.

2.3.1 RAdio Detection And Ranging (RADAR)

The RADAR technology dates back to the early 1900s, but was improved significantly during World War 2. In the maritime sector RADAR have been common for decades giving great improvement in navigation and safety at night and in poor visibility conditions. The RADAR consists of an transmitter and receiver antenna which is rotating with a set speed. Older systems used the same antenna for transmitting and receiving. They transmitted powerful short pulses and waited for the return (pulse radar).

Modern RADAR uses at least one receiver and one transmitter antenna such that a much weaker, and safer, continuous signal can be transmitted and received. A Frequency Modulated (FM) signal gives a much more accurate and reliable detection than pure pulses. Maritime RADAR is designed so that the transmitted signal is narrow in the horizontal plane, but wide in the vertical, radiating a radio signal with a vertically fan-shaped beam. This arrange for a good resolution horizontally while relatively unaffected by vessel motion. The transmitted signal gets reflected by objects such as boats or land and is picked up by the receiver antenna. The distance between the RADAR and the object is measured as half the time of flight for light in air

$$r = \frac{c \cdot t}{2} \tag{2.4}$$

where c is the speed of light, r is the distance to the object and t is the time passed until the transmitted signal has returned. The strength of the return indicates the reflectance of the object, waves and rain will for instance give low return signal strength and much can be filtered out. There are functions within the RADAR system for adjusting range and other settings as well as some filtering.

The "Simrad Broadband 4G™ RADAR" used in the experiments utilizes modern Frequency Modulated Continuous Wave (FMCW) technology which give continuous data coverage and very safe operation with low radiation [20]. There are two separate antennas which rotates together, where one is transmitting radio waves in the X-band (8-12 GHz) and one is receiving the return signals. The range setting for the RADAR controls the rotational speed of the antennas. The maximum range for this RADAR is 32 Nautical Miles (NM) (1 NM = 1852 metres), with a rotational speed of 24 Rounds Per Minute (RPM) = 0,4 Hz. The maximum rotational speed is 48 RPM = 0,8 Hz, with a minimum range setting of only 50 meters. Low range settings gives best range resolution while the horizontal angular resolution of the antenna is fixed at 5,2°. The vertical angular resolution is 25°. The relatively wide horizontal resolution might give problems where close objects are smeared together and not possible to separate in the data. Smaller objects some distance away will also appear larger than they are in the data. Another limitation using RADAR is that it is hard to reliably detect objects very close due to high levels of noise. The Simrad RADAR has acceptable performance beyond 50 meters, based on the testing done throughout the Autosea project. The data output from this RADAR have limited amplitude information so that further filtering of the data is difficult.

2.4 Passive Sensors

Passive sensors do not contain any transmitter and only senses the environment via changes in the received signals. This may be electro-magnetic waves, such as radio-waves or light, sound, wind, temperature, gravity, magnetic fields etc.

2.4.1 Inertial Navigation System (INS)

The Inertial Navigation System (INS) is a state of the art sensor package used to measure and estimate the current state of the vessel. It provides position and attitude measurements to be used by other systems and sensors with a high output rate. The INS consists of an Inertial Measurement Unit (IMU) with built in accelerometers and rate gyros in 3 axis. Sometimes also 3 axis magnetometers are used to provide reference to magnetic north. The position and heading output from the IMU is corrected by a GNSS system with two antennas and receivers. This is referred to as GNSS aided navigation [8] and is a sensor fusion algorithm based on nonlinear Kalman Filter (KF) that is running within the INS.

Figure 2.2: Simrad Broadband 4G maritime RADAR

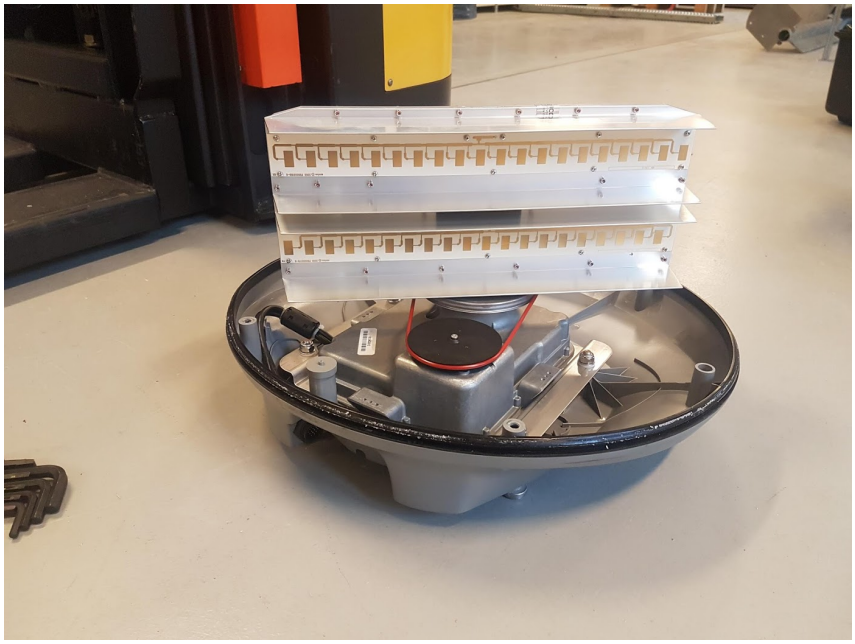


Figure 2.3: Simrad RADAR with protective dome removed

The GNSS primary receiver have Real-Time Kinematics (RTK) capabilities, indicating that received correction signals can be used in real-time to get very accurate position estimates.

The INS used in the experiments is the ”Kongsberg Seapath 330+” which have a state-of-the-art performance. See table 2.1.

The entire INS system is calibrated so that the sensors are exactly known relative to each other and preferably the Center Of Rotation (COR) of the vessel. The INS can therefore be considered as one system giving out the motion and attitude of a single vessel/BODY reference point. Throughout the thesis the Seapath INS is considered as a perfect sensor, and no uncertainty from own pose and position is taken into account.

- Heading accuracy 0.04° RMS (4 m baseline)
- Roll and pitch accuracy 0.008° RMS for ±5° amplitude
- Heave accuracy (delayed signal) 2 cm or 2% whichever is highest
- Position accuracy RTK (X and Y) 1 cm + 1 ppm RMS
- Position accuracy RTK (Z) 2 cm + 1 ppm RMS
- Velocity accuracy 0.03 m/s (RMS)

Table 2.1: Performance of the Kongsberg Seapath 330+ INS

2.4.2 Digital Daylight Camera

Most cameras today use digital EO sensor arrays with several Megapixels. The Ladybug camera system described below have a total of 6 cameras, each with a resolution of 2448 x 2048 pixels (5 Megapixels), covering the 360°horizon plus upwards.

2.4.3 Omnidirectional Cameras

Omnidirectional camera systems are cameras that have a total Field Of View (FOV) covering the entire 360°horizon. They are designed in several different ways using only one or multiple cameras. Certain types of lenses, such as wide FOV dioptric cameras, commonly known as wide angle fisheye, and catadioptric cameras (camera and mirror systems) can provide 360 degree coverage using only one lens [21][22]. The images from fisheye lenses are heavily distorted and the resolution of the single camera is spread around the entire image. Most of the image will also be of the sky directly above, which in this setup is uninteresting. Fisheye single lens cameras are therefore unsuited for fusion with RADAR.

The camera setup using omnidirectional, convex mirror above the lens removes the problem with most of the image containing the sky, and by using the correct shape of the mirror the distortion can be low [23][22]. This model requires very precise and high quality mirrors for good results. The simplicity of using only one sensor gives uncomplicated sensor models. However the limitations of using only one sensor gives a relatively low resolution in the interesting parts of the image. The image also have a distinct different appearance from normal perspective cameras as can be seen in figure 2.4. This might give problems when feeding the image to a CNN for detections and would require significant training effort.

Another possibility for using only one camera is to use a rotating camera. This mimics how the RADAR functions. However the two sensors have different workings and a camera works best when relatively stationary, especially in low light conditions. This can affect the image quality negatively. The acquisition with this rotating camera setup will also be quite slow. Part of the reason for fusing RADAR and camera is to have a faster sensor to help with reliable tracking of objects.

Figure 2.4: Fisheye camera and hyperbolic mirror camera compared. Wikimedia Commons and Columbia University CAVE project.

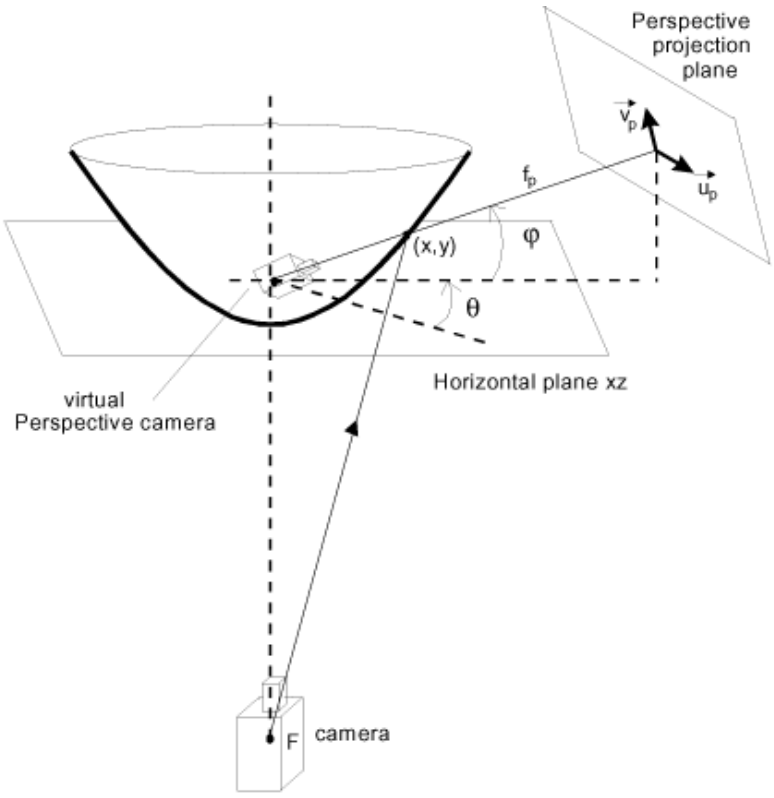


Figure 2.5: Hyperbolic mirror camera principle [23]

Due to all the limitations of the above mentioned omnidirectional cameras, the best camera type in this setup is polydioptric cameras, where the full 360°coverage is realized using multiple overlapping cameras [22][24]. And especially because the tracked object using RADAR is usually some distance away, effectively reducing the visibility of the objects, the better resolution from the multiple cameras are valuable. The exact camera system used in this report is the ”Ladybug 5+” from ”FLIR® Machine Vision”.

Figure 2.6: The FLIR Ladybug 5+ camera system

2.4.4 Camera Calibration

As can be seen from figure 2.8 there are several percent overlap between the pictures from the Ladybug camera. This is vital to perform stitching between the images, to create one large panoramic image, as well as track objects translating from one image into the next.

There are, however, three reasons for not stitching the images together; Firstly the Real-time Recurrent Regression (Re^3) image tracker and the You Only Look Once (YOLO) detector, which will be described in chapter 3, perform best on images that have somewhat quadratic shape as the images are converted into a fixed size before fed through the network [4] [3]. Furthermore the stitching comes with a high computational cost, reducing the resources available. Also the stitching might create strange artifacts that may negatively impact the robustness of the detector. Also the Ladybug Linux Application Programming Interface (API) does not support stitching out of the box.

The Ladybug camera system is built using a robust aluminium housing. The 6 cameras come with factory calibrated parameters that can be used during processing. These parameters are unfortunately considered proprietary by FLIR and they are not available to the user. To be able to calibrate using FLIR parameters one must use the Ladybug API, see table 2.2, which is not supported by ROS. The pictures are output as 6 single images from the Ladybug ROS node. They are uncalibrated and have a large barrel distortion as can be seen from images 2.8, 2.11 and 2.12.

Table 2.2 indicates how another software could use the Ladybug API for calibration. To convert from pixel location to the needed 3D ray, a transformation must be performed using the camera matrices 2.12 through 2.21 [25] [26].

The Ladybug 5+ uses the Sony IMX264 image sensor (5 MP), with a resolution of 2048 × 2448 for each camera. FLIR are not willing to provide detailed information about the camera system, including FOV of each camera or lens model used. The parameters for distortion as well as the details of FOV in each direction can however be calculated from calibration. The angle of view along the vertical and horizontal axis of each camera with an Equisolid fisheye lens is calculated as follows:

$$\alpha = 2 \arcsin \frac{d}{2f} \tag{2.5}$$

$$\alpha_x = 2 \arcsin \frac{7.0656}{2 \cdot 4.4} = 106.82° \tag{2.6}$$

$$\alpha_y = 2 \arcsin \frac{8.4456}{2 \cdot 4.4} = 147.37° \tag{2.7}$$

Where d is the sensor size in mm (height or width) and f is the focal length in mm [27]. It is the Equisolid fisheye model that is used throughout the thesis.

The most common lens types, the Stereographic fisheye and Perspective lens as well as the Equidistant fisheye lens is calculated as follows:

$$\alpha = 2 \arctan \frac{d}{2f} \tag{2.8}$$

$$\alpha = \frac{d}{f} \tag{2.9}$$

Both of these lens models gave however smaller FOV than what was observed in the images and verified using aerial photo maps and landmarks visible in the image. The pure landmark test indicated that the FOV in the horizontal direction was slightly above 100°. The common Perspective and Stereographic fisheye lens model did not match the observed results as shown below. They are shown only for reference and are not used in any computations.

$$\alpha_x = 2 \arctan \frac{7.0656}{2 \cdot 4.4} = 77.52° \tag{2.10}$$

$$\alpha_y = 2 \arctan \frac{8.4456}{2 \cdot 4.4} = 87.65° \tag{2.11}$$

The cameras have wide angle of view as can be seen in eqation 2.6 and 2.7, and introduce much barrel distortion which must be counteracted. The rectified images can be modeled using the standard perspective camera model (pinhole camera) [25].

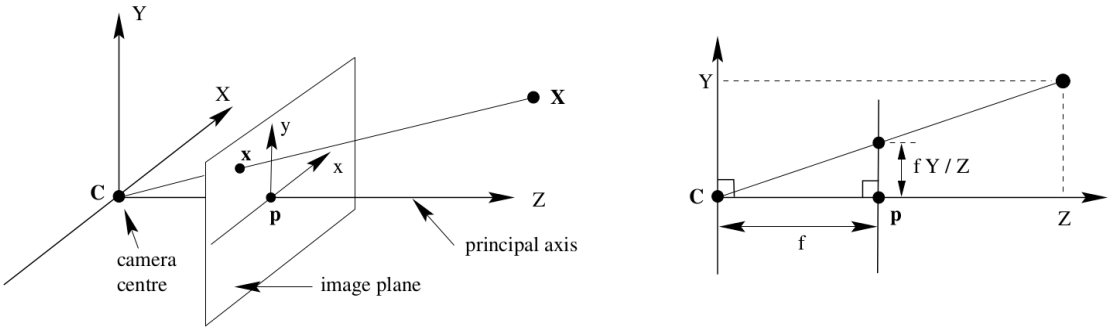


Figure 2.7: The pinhole camera model

$$\mathbf{P} = \mathbf{K}\left[\mathbf{R}|\mathbf{t}\right] \tag{2.12}$$

$$\mathbf{K} = \begin{bmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.13}$$

$$\left[\mathbf{R}|\mathbf{t}\right] = \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} & t_1 \\ r_{2,1} & r_{2,2} & r_{2,3} & t_2 \\ r_{3,1} & r_{3,2} & r_{3,3} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.14}$$

The $\mathbf{P}$ matrix is the camera matrix composed of the Intrinsic and Extrinsic matrices. The camera matrix describes the mapping of the pinhole camera from 3D points in the world frame to 2D points in the image plane.
$\mathbf{K}$ is called the Intrinsic matrix or the camera calibration matrix and transforms the 3D camera coordinates to 2D homogeneous image coordinates. The f_i parameters are the focal lengths, the x_0 and y_0 are the center point offset of the sensor (roughly the center of the sensor) and the s parameter is a skew value for non-rectangular pixels which is usually 0.

$\left[\mathbf{R}|\mathbf{t}\right]$ is a transformation matrix $SE(3)$ called the Extrinsic matrix or the view matrix and describes the camera's location and pose in the world frame. It contains a 3D rotation and a translation expressed in homogeneous coordinates.

$$\mathbf{u} = \mathbf{K}\hat{\mathbf{x}} \tag{2.15}$$

$$\mathbf{u} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{2.16}$$

$$\hat{\mathbf{x}} = \begin{bmatrix} x_{corrected} \\ y_{corrected} \\ 1 \end{bmatrix} = \begin{bmatrix} (1 + k_1 r^2 + k_2 r^4 + k_3 r^6)x \\ (1 + k_1 r^2 + k_2 r^4 + k_3 r^6)y \\ 1 \end{bmatrix} \tag{2.17}$$

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.18}$$

$$r = \sqrt{(x_\mathrm{d} - x_0)^2 + (y_\mathrm{d} - y_0)^2} \tag{2.19}$$

Where $\mathbf{u}$ is the pixel coordinates in the normalized image plane, $\mathbf{x}$ is real coordinates in mm given in the camera coordinate system, k_i is radial distortion coefficients and r is the distance from the image/distor-

tion center to the distorted coordinate x_d, y_d.

$$\mathbf{x} = \begin{bmatrix} \mathbf{R}|\mathbf{t} \end{bmatrix} \mathbf{U} \tag{2.20}$$

$$\mathbf{U} = \begin{bmatrix} X^n \\ Y^n \\ Z^n \\ 1 \end{bmatrix} \tag{2.21}$$

$\mathbf{U}$ is homogeneous world coordinates given in NED.

Since the calibration parameters from the Ladybug API is unavailabe to ROS the calibration must be manually performed using a known checkerboard pattern or similar which is supported by software such as ROS, OpenCV or Matlab. See figure 2.9. Note that other calibration parameters than barrel distortion is not shown in equation 2.17, but can also be calibrated.

Figure 2.8: The 5 images that form the panoramic view from the Ladybug camera

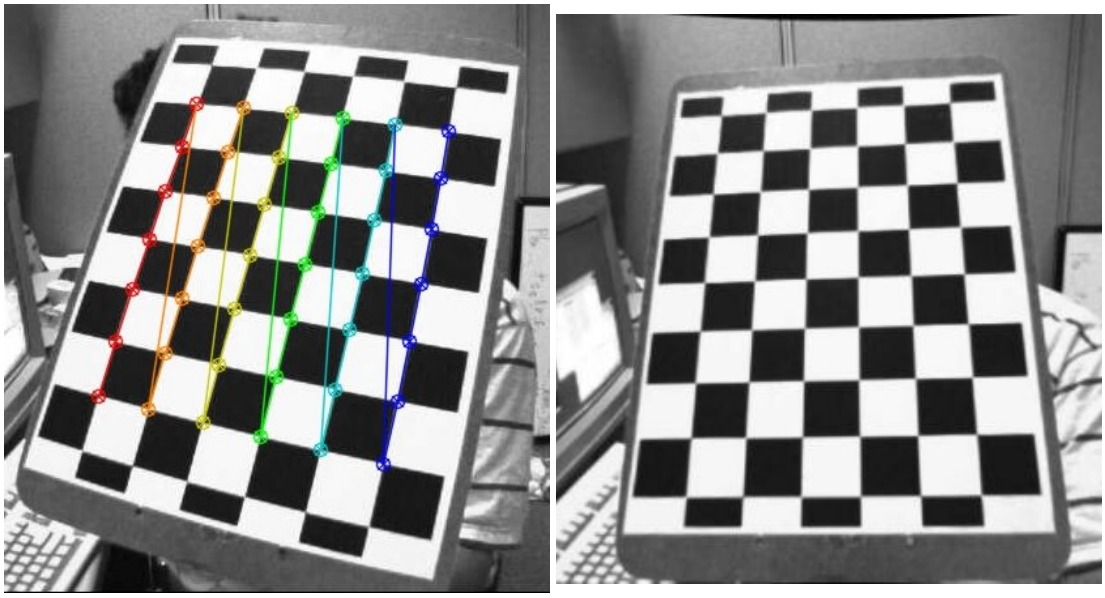


Figure 2.9: Calibration of barrel distortion using OpenCV. [28]

To find the distortion coefficients it is useful to use the built in calibration routine in OpenCV [28] and a large number of images taken at different distances and angles with respect to the camera. Rotations of the checkerboard pattern is also recommended.

Since the FOV of the Ladybug cameras is in the outermost region of where the Stereographic model is valid, the best calibration routine would be the Fisheye calibration. Unfortunately OpenCV currently

have a bug making use of that model difficult, and ROS does not have a separate routine for Fisheye cameras.

The results from calibration of a random set of 90 images from a pool of 380 images gave average results of:

$$\mathbf{K} = \begin{bmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1600 & 0 & 1024 \\ 0 & 1600 & 1232 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.22}$$

$$\mathbf{D} = \begin{bmatrix} k_1 & k_2 & p_1 & p_2 & k_3 \end{bmatrix} = \begin{bmatrix} -0.38 & 0.16 & -0.0001 & -0.00075 & -0.033 \end{bmatrix} \tag{2.23}$$

Here $\mathbf{K}$ is the Camera intrinsic matrix and $\mathbf{D}$ is the vector with distortion coefficients. The parameters k is counteracting barrel distortion, while p is counteracting tangential distortion. Tangential distortion is caused by the lens not being aligned parallel to the image plane. The cameras used are of high quality and removing the p parameters show no visible change. The p parameters are therefore not used in the correction as they are just as likely due to imperfect calibration routines. The same calibration parameters are used for all 5 cameras covering the horizon. Optimally the calibration should be done specifically for each camera and the cameras should be calibrated against each other as well. This optimization is left for future improvements.

The images coming from the Ladybug are undistorted using the OpenCV method ”cv2.getOptimalNewCameraMatrix” and then ”cv2.undistort”.
The image is clipped to remove the black parts which appear along the axis on the edges of the rectified image. See image 2.10.

Ladybug API

If one have the possibility to use the Ladybug API the procedure below can be used. To convert a pixel location in a raw image to a 3 Dimensional (3D) ray in the Ladybug Coordinate System the following steps should be taken using several of the Ladybug API functions [25]:

1. Obtain the focal length for the appropriate camera using:
   ladybugGetCameraUnitFocalLength()
2. Obtain the image center for camera using:
   ladybugGetCameraUnitImageCenter()
3. Obtain 6D extrinsics vector (Euler angles and translation) for the camera using:
   ladybugGetCameraUnitExtrinsics()
4. Rectify 2D pixel location using:
   ladybugRectifyPixel()
5. Find the (u,v) pixel coordinate for this rectified image location.
6. Transform the rectified 2D pixel location into a 3D ray within the local camera coordinate system.
7. Transform the local 3D ray to a 3D ray in the Ladybug Coordinate System.

Table 2.2: Using the Ladybug API to convert a pixel coordinate to a 3D ray

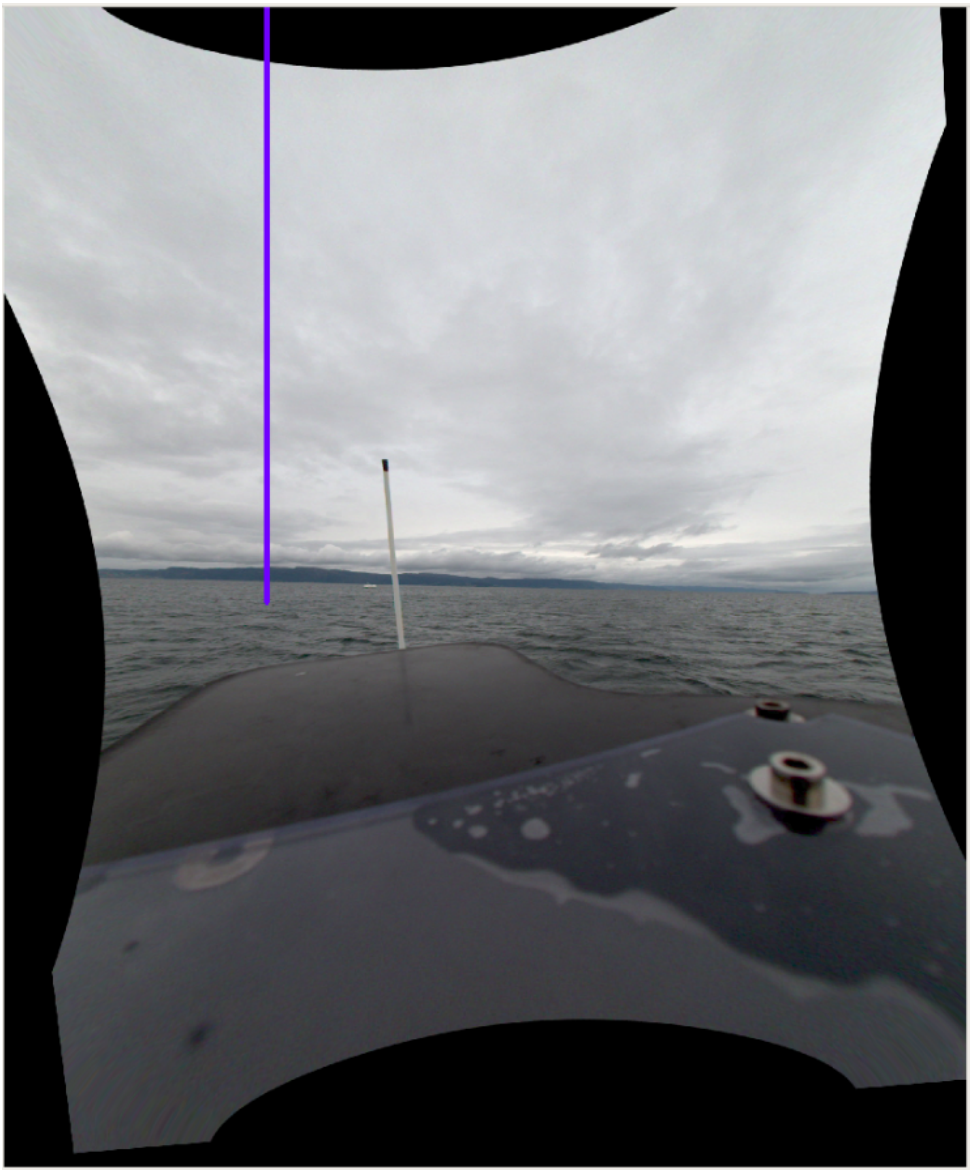


Figure 2.10: Example of the rectified image before cropping

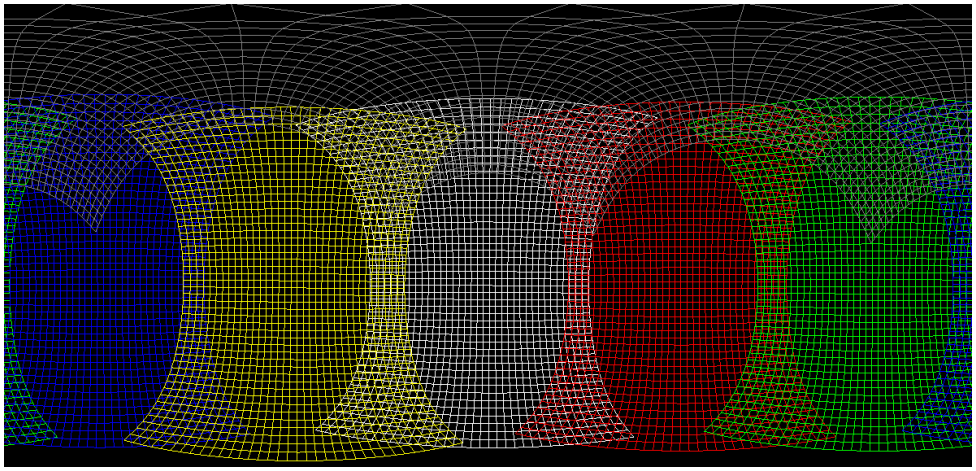


Figure 2.11: 2D polygon mesh of stitched Ladybug images [26]

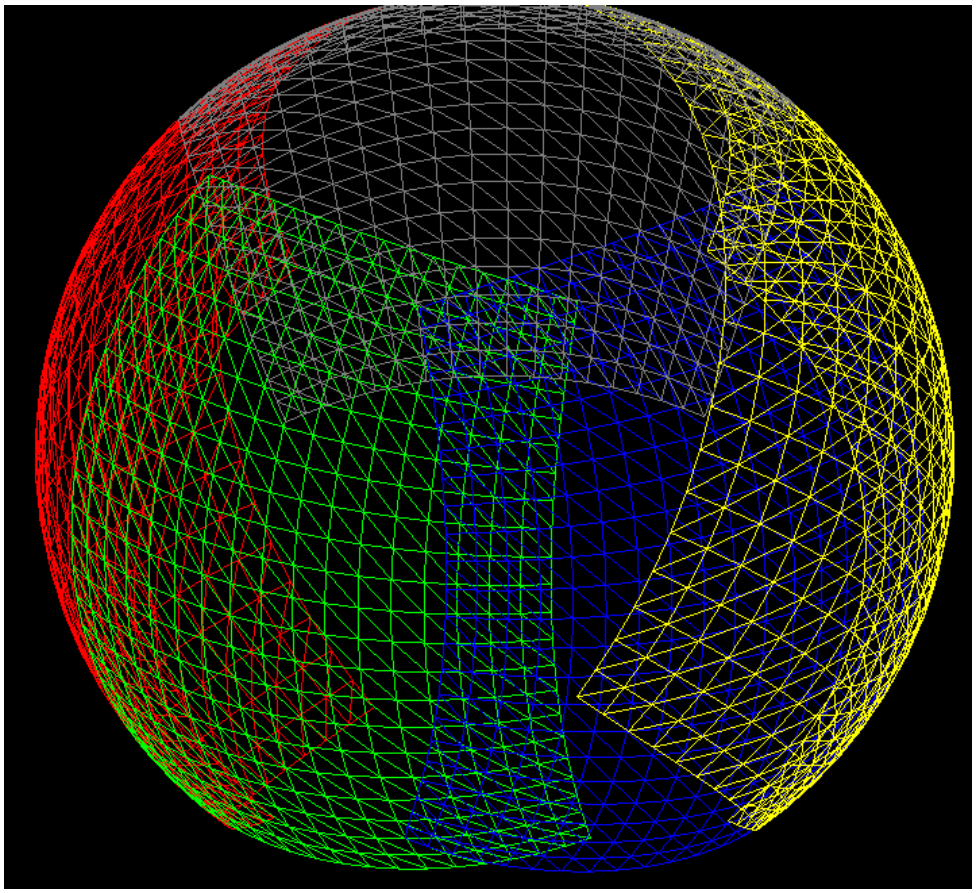


Figure 2.12: 3D polygon mesh of stitched Ladybug images [26]

Chapter 3

Deep Learning Methods

Deep learning is a modern Artificial Intelligence (AI) and Machine Learning (ML) approach especially used for information extraction from complex structures such as images, video and speech. It is based on a layered data structure containing artificial neurons, where each layer extracts the most valuable information, and passes this on to the next layer. The output of the neurons are weighted through some type of non-linear function to avoid the whole network becoming a linear combination. The network can be trained so that the information extracted in each layer is optimized for the given task. This is commonly referred to as backpropagation, where the weights (gain) and bias for input to the neurons are updated. One can also finetune a previously trained network by retraining, and updating the parameters, only in the last few final layers. This is called transfer learning. Backpropagation will not be further detailed here. For a good reference on deep learning methods we can refer to [29] and [30]. Transfer learning is however a method that is suitable for retraining the YOLOv3 and Re^3 networks as described below. Doing this one can also limit or adapt the number and type of objects the detector should look for.

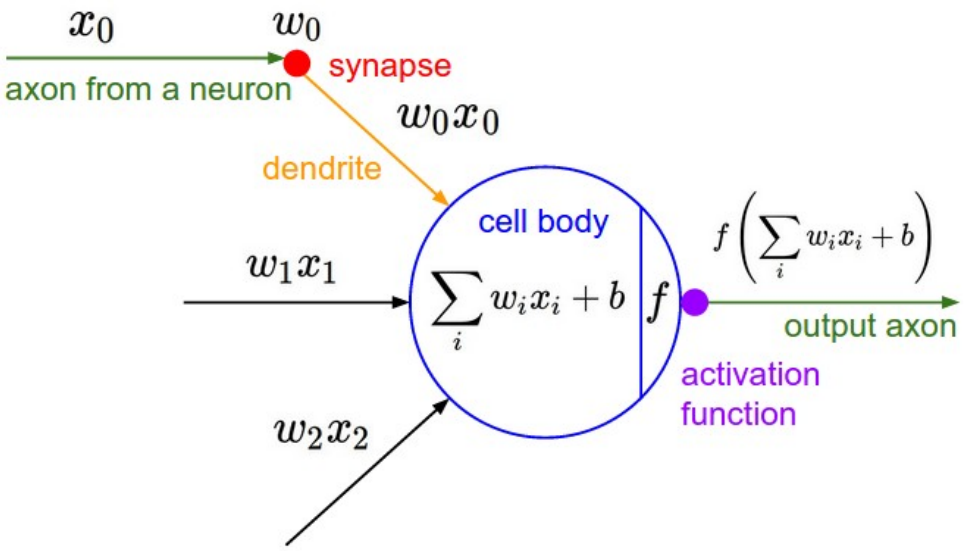


Figure 3.1: An illustration of the typical Neuron model used in fully connected neural networks

3.1 Convolutional Neural Network (CNN)

CNNs are a special type of neural networks used primarily where the input is an image or a video (series of images). The spatial matrix style of images makes them ideal for reducing the size through convolution as separate parts of an image does not directly affect other parts.

The input image has a size of $(n \times m \times c)$, with $c = 3$ channels (Red Green Blue (RGB)) or $c = 1$ channel if grayscale. After the convolution layer the size of the image is changed to $((n-s) \times (m-s) \times f)$ where f is the number of different filters and s is the number of pixels the filter kernel is translated (stride). If pooling is used the size is even further reduced, while keeping the most interesting features extracted by the filters. However, recent findings may lead the way for pooling layers to be replaced by convolutional layers with larger strides [31][32].
For each convolution step, the spatial size is changed so that the image matrix is transformed from a large, flat "box" to a small, tall "box". Finally the output of the last convolutional layer is fed to a classical fully connected layer for classification. A large percentage of the total number of parameters are within this last fully connected layer. The convolutional layers utilize parameter sharing so that the number of parameters depend on the kernel size rather than the image size, dramatically reducing the number of parameters that must be optimized. Newer network architectures therefore try to limit the use of fully connected layers [31][32].

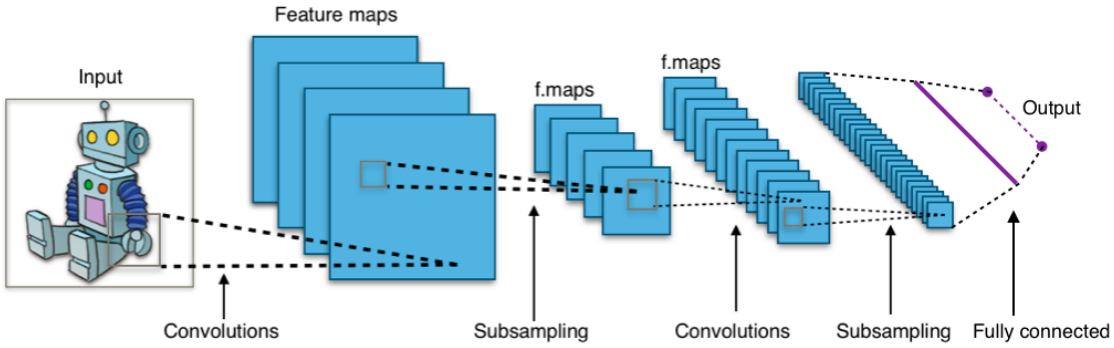


Figure 3.2: A typical CNN structure. Wikimedia Commons

3.2 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) is a special type of Recurrent Neural Network (RNN) capable of learning long-term dependencies.

In a regular CNN the network does not retain any information from previous timesteps, and performs the same computations every time. To be able to utilize the previous data, this must be stored and fed as input to the next computation, or looped in the data structure. This is referred to as a RNN. In contrast to standard RNN the LSTM network has a data line transferring previous information that is not the output of the cell. This data line or cell state, C_t is affected by 3 neural layers in the cell, and it also influences the output of the cell. The cell state is also very effective in reducing the problem of vanishing gradient in back propagation, preventing learning, during training.

The first layer is called the "forget gate layer" and takes the previous output h_{t-1} combined with the current input X_t and scales the data in C_t by the output of the Sigmoid function σ which ranges from 0 to 1.

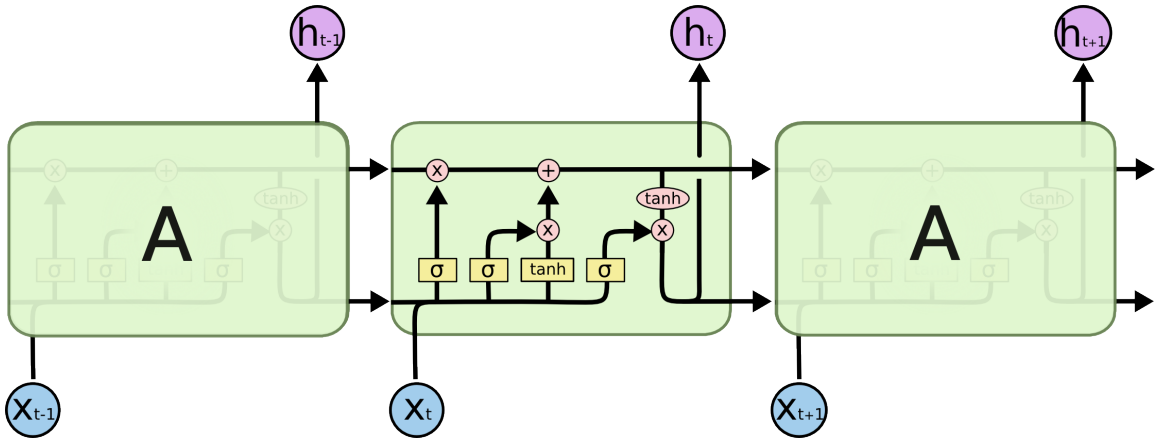


Figure 3.3: The LSTM chain over 3 timesteps, notice the 4 gates affecting the information flow

The next two layers, the Sigmoid "input gate layer" and the hyperbolic tangent (tanh) layer, adds information to the cell state C_t.
Finally the output h_t is a multiplication of the last Sigmoid layer and a tanh weighting of C_t.

The LSTM networks can similarly to other deep learning networks be trained using annotated data to improve the output predictions. A LSTM network can therefore effectively retain important information from previous images and ignore information which is not useful due to occlusion for instance. The network can because of this predict robustly over time.

3.3 You Only Look Once (YOLO) v3 detector

YOLO is a state-of-the-art object detection algorithm capable of running in real-time even for large images on a powerful Graphics Processing Unit (GPU) [3][33]. The algorithm is built using "Darknet" which is an open source neural network framework written in C and Compute Unified Device Architecture (CUDA) so that it is capable of computing both using a Central Processing Unit (CPU) and a GPU. YOLOv3 uses a version of "Darknet" with 53 layers, see figure 3.4. There are many versions of YOLO and tiny-yolo uses for instance only 15 layers [34]. This is considerably faster, but also much less reliable when detecting objects. Especially small objects and objects with a different than "normal" look.

The detector layers are added to the output of the image processing CNN. As can be seen from figure 3.5 the regular YOLOv3 detector performs detection on 3 different scales and does an upsampling of the images between the layers, similar to a feature pyramid network. This makes version 3 better at detecting small objects than the previous verions. The detector is fast due to the fact that it only extracts information from the image one single time (You Only Look Once). The detections from each 3D tensor output, for all the 3 layers are combined to make the total output, and only detections with a certainty above some threshold are accepted. The detector used in the tests is trained on the Common Objects in COntext (COCO) dataset which contains 80 different object types [35][3]. Compared to other high end detectors, such as Faster Single Shot Detector (SSD) and RetinaNET, YOLOv3 is faster and just as robust [7]. It is slightly weaker in adjusting the bounding box to perfectly match the ground truth, also referred to as mean Average Precision (mAP).

3.3.1 Current Object Detection development

New deep learning based object detectors are continuously performing better as research finds better and more efficient methods. The current top performers with respect to mean Average Precision is the FlowNet 2.0 [36], TridentNet [37] and CenterNet [38] which all approach an AP of 50.
The ”Objects As Points” algorithm using CenterNet-ResNet 18 as model is about 3 times as fast as YOLOv3, 142 fps compared to 45 fps on a NVidia Titan Xp GPU [39]. CenterNet uses a CNN which produces a heatmap with detected objects. The ”hottest” points on this heatmap coincides with object centers and the bounding boxes are built around the center point.

An interesting aspect of this centerpoint detection framework combined with the active sensor target tracking, is that no bounding boxes need to be created and no image tracker need to be used if the detector is fast enough and reliable enough. The active sensor tracker outputs a single centerpoint for the target and CenterNet centerpoint could potentially be used directly for sensor fusion. This would reduce the complexity of the algorithm and reduce the demands for high power processing equipment.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 3.4: The Darknet-53 CNN structure of YOLOv3

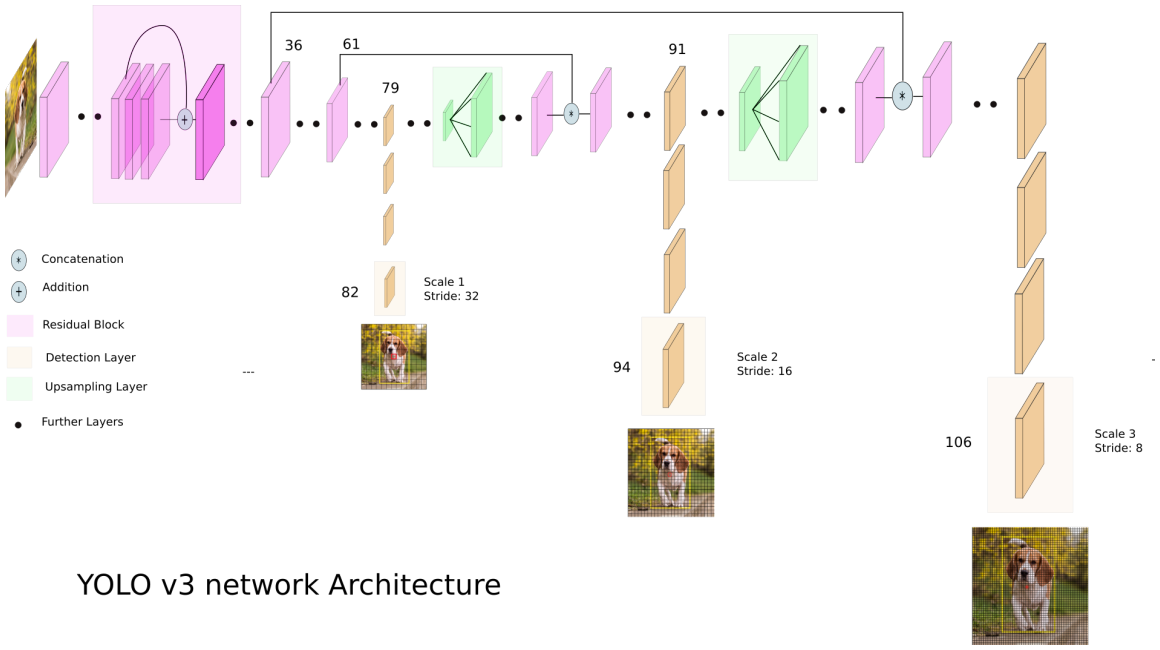


Figure 3.5: The feature extractor of YOLOv3 on the Darknet-53 CNN

3.4 Real-time Recurrent Regression (Re^3) tracker

The Re^3 tracker is the first successful hybrid tracker utilizing both a pretrained deep learning based offline tracker and an online tracker capable of learning features of the tracked object real-time [4]. It is a generic tracker which is trained on the "Imagenet" dataset containing almost 22.000 object types and more than 14 million images in total [40]. The model is trained on very generic data and therefore can track almost anything. It is shown to be robust against changes in appearance of the tracked object and very fast, giving real-time performance even without a GPU.

The tracking pipeline, as shown in figure 3.6, consists of convolutional layers to embed the object appearance, recurrent layers to remember appearance and motion information, and a regression layer to output the location of the object. The tracker takes in a cropped part of both the current image and the previous image that is twice the size of the previously predicted bounding box and feeds both through separate CNNs. This makes the network able to fully separate out the differences between the images before they are concatenated.

The recurrent layers are built using a two layer LSTM with 1024 units/neurons each, followed by a 2048 unit fully connected layer for predicting the 4 corners of the bounding box. The two layer LSTM is likely able to capture more complex object transformations and remember longer term relationships than the single layer LSTM [4].

The tracker is capable of tracking multiple objects simultaneously in real-time. One must however be cautious when it comes to camera motion, as the network would only be able to track objects translating less than half the bounding box size between frames. Tests for increasing padding of the bounding box have been performed by the author, but the size is hard-coded and involves much of the network. A larger box also required more processing and reduced the tracking speed of the network.

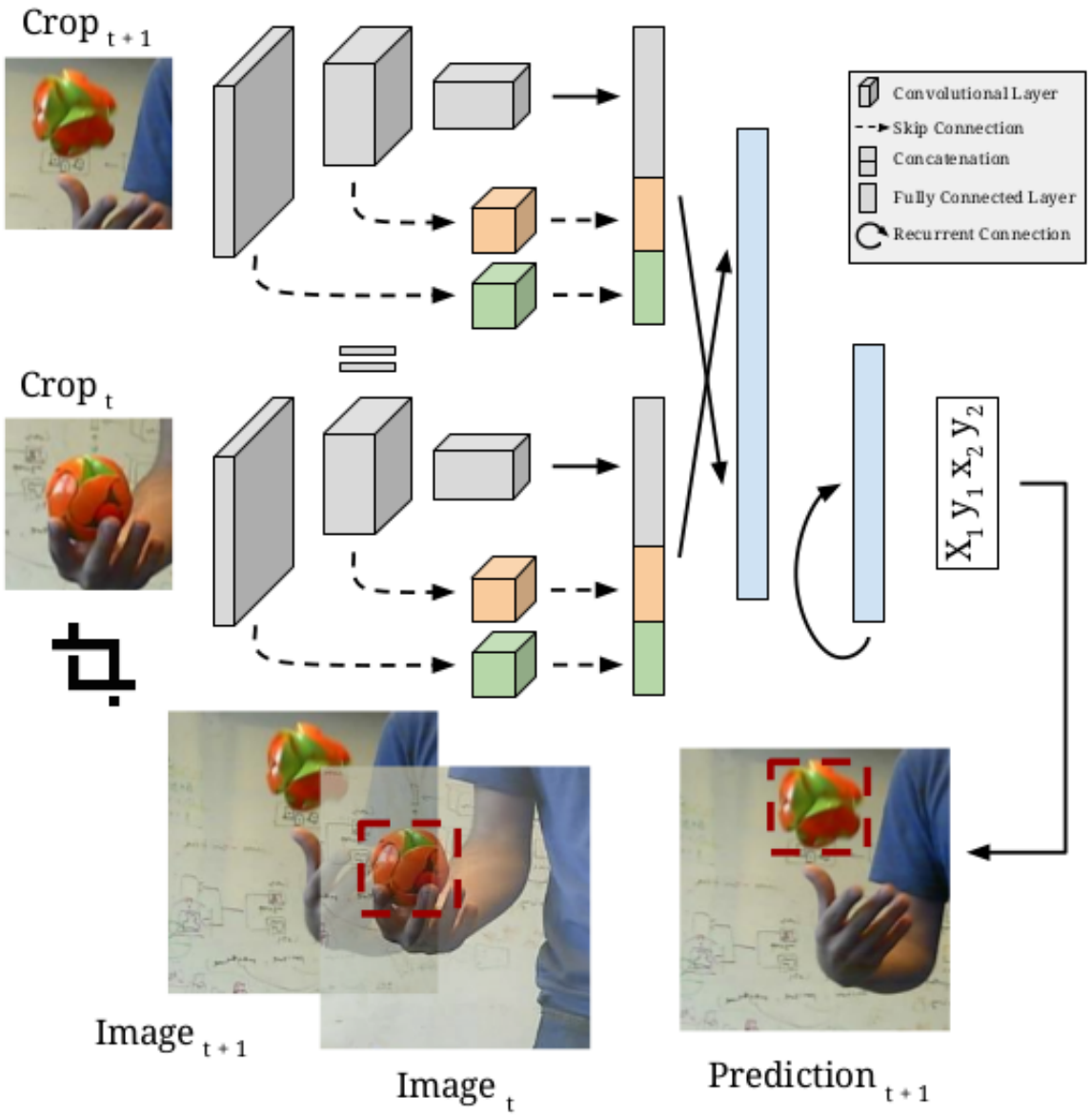


Figure 3.6: Overview of the structure of Re^3

3.4.1 New Visual tracking Methods

The Re^3 tracker is definitively most robust when used in a stationary camera setting. When used on the recordings from the ASV, even though the image is stabilized, most of the image will experience some motion between frames. To overcome this issue it helps to enforce an even stricter handling of timing and INS attitude. It will anyway not be possible to stabilize the image 100% without performing stabilization directly in the image. The problem arising from the vessel dynamics are difficult to handle in a realtime and computationally cheap manner. Improved classical methods using optical flow [36][41] might help further align the images prior to tracking. This is however not trivial to combine with the other methods, and they add further processing costs.

As can be seen in the results of The Visual Object Tracking Challenge (VOT2018) [7], all the best performing visual trackers are CNN based. The previous state-of-the-art visual trackers based on Discriminative Correlation Filters (DCF) are outperformed by the new CNN based trackers, and in real-time applications the Siamese Network based trackers come out on top.

3.4.2 Siamese Networks

Siamese Networks are built up using two or more identical networks in parallel. Often one of the output vectors are precomputed, for instance from a template, thus forming a baseline the other output vector is compared against. This works in a similar way to classical template matching for object recognition. The template matching together with online and offline training of the deep learning architecture gives a tracking performance which is state-of-the-art in terms of robustness against changes of appearance, lighting conditions, size etc. Substantial online training takes time and decrease the real-time tracking ability [42]. A compromise between real-time performance and good online training, to adapt to changes, must therefore be made. Siamese trackers use metric learning at their core [15], and the goal is to find a vector representation (learn an embedding space) that can maximize the interclass correlation between different objects and minimize the intraclass correlation for the same object. They are pre-trained on highly general semantic segmentet datasets to learn general object classifications. The Siamese networks are promising in tracking applications due to their balanced accuracy and speed. Recently the DaSiamRPN network won the The Visual Object Tracking Challenge (VOT2018) real-time challenge [43][15]. This network is trained to be distractor-aware so that it does not start tracking nearby and/or similar objects. Another and even newer Siamese based tracker showing good potential and an impressive tracking speed of 120 fps is the SPM-Tracker (Series-Parallel Matching for Real-Time Visual Object Tracking) [44]. It performs an effective coarse matching prior to, and combined with, a fine matching. This helps the model track using only a single image scale.

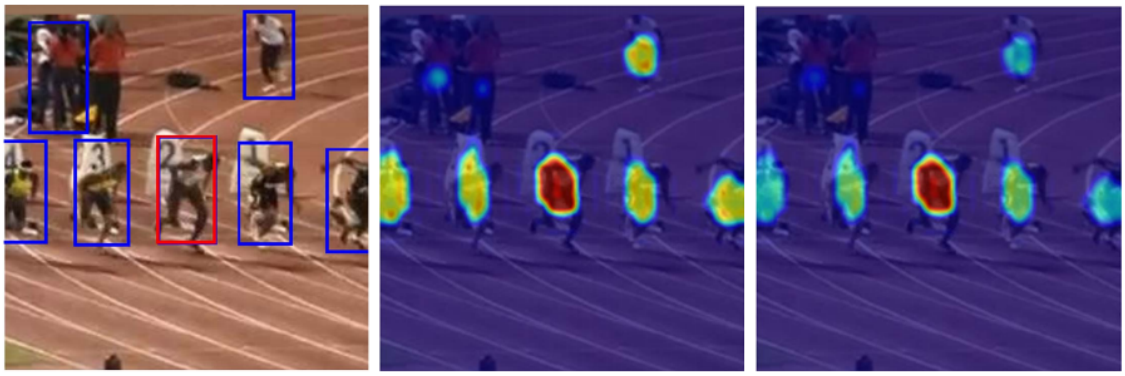


Figure 3.7: Distractor-aware Siamese Network tracker, DaSiamRPN

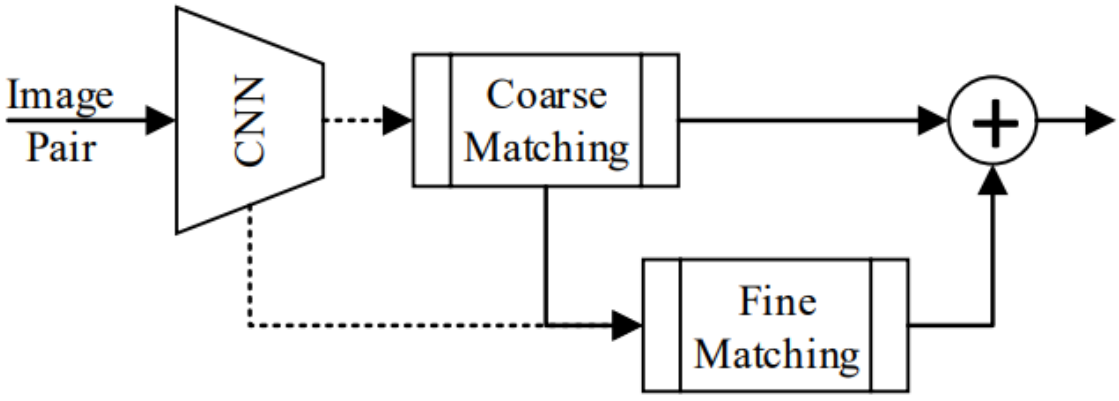


Figure 3.8: Basis of the Series-Parallel Matching tracker

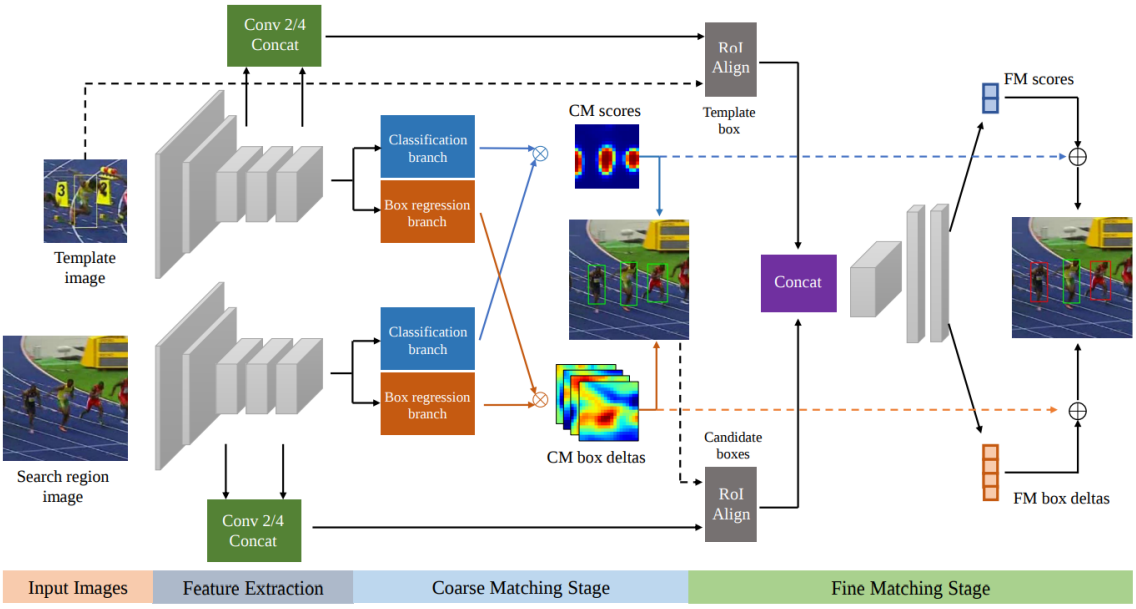


Figure 3.9: Series-Parallel Matching for Real-Time Visual Object Tracking

Chapter 4

Target Detection and Tracking

New and improved sensors are constantly being developed. This indicates that a fusion algorithm for real life COLAV scenarios should be flexible enough to import new sensors when they are released to marked. This thesis builds a starting point for fusing many different sensors outputting either measurements directly in Cartesian coordinates or as angle measurements from the sensor.

In this thesis we suggest to use a video (series of images) tracker for improved fusion performance of the passive camera sensors with the active RADAR sensor. We suggest to run a deep learning detection algorithm to find boats or other interesting objects and use this to initialize and update the tracker.

The YOLO version 3 detector is used in the thesis as this was state-of-the-art in terms of speed and robustness when the project was started [3]. The Re^3 tracker is used in this thesis as it is a generic tracker and is demonstrated to be robust and fast [4]. The last few months have shown further improved object detection methods in the Visual Object Tracking (VOT) Challenge [7]. Other trackers have also emerged in the last few months which show promising results. The best trackers in the VOT2018 all use CNN architecture and some are based on Siamese networks. The Re^3 tracker have a comparable approach to the tracking problem as the Siamese trackers.

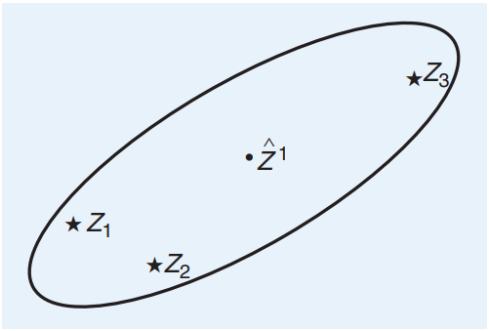
One of the huge benefits with using active sensors for target detection and tracking is that they are capable of measuring both the angle and distance to an object. Passive sensors will need to triangulate to calculate the distance to an object. This will within the task of tracking at sea usually not be feasible, or one must make assumptions such that the object is stationary, or at least slow moving compared to own vessel. Dept or distance information is therefore usually only reliably gathered using active sensors such as RADAR and LiDAR.

Passive sensors on the other hand do not have any transmitters and therefore do not interfere with any other instruments on board, this makes them more flexible in where they can be mounted. Cameras are the potentially fastest of the sensors considered here, practically only limited by the computing power for processing the images. As mentioned in chapter 2, do passive sensors not give depth or range information directly. It is possible to mount the camera high above the surface or use a stereo camera setup with a long baseline between the cameras on a large vessel, but this is not feasible on a small ASV. There exists methods for estimating depth from single cameras, as used extensively in for example Simultaneous Localization And Mapping (SLAM). These methods are however primarily suitable where the surroundings are stationary. The sea and objects we wish to track are to the contrary dynamic of nature.

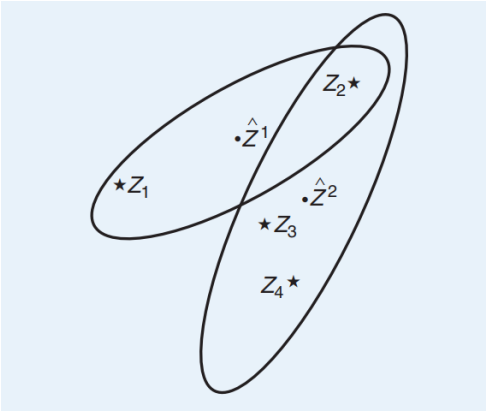
4.1 Probability

The likelihood that an event will occur.
Probability is the basis of sensor fusion in one form or the other and is vital in multi object tracking. In the world of imperfect sensors, no measurement is always guaranteed to be correct. A notion of uncertainty and probability is therefore important. The different sensors have different properties. They may for instance be prone to false detections, have much clutter and noise or have very low, but reliable, detection rate. No existing sensor is perfect.

It is important to take into account that every detection might or might not be the target we are seeking. In a COLAV scenario we can use some physical laws to help, so that the area in which we are looking for the target is limited to an area around the estimated position, a validation region. The detections in this region are candidates for the correct target. At most one of them are correct, but they might also all be clutter or other targets. The notion of missing detections are therefore important as well. In a RADAR measurement another problem might arise; that two or more targets are presented as one and the same detection. If this detection is then only used for one of the tracks it might give drift and higher uncertainty to the other track(s).



(a) The validation region around the estimated position $\hat{Z}^1$ with three detections Z_1, Z_2 and Z_3 [45]



(b) The overlapping validation region around two estimated positions $\hat{Z}$ with four detections Z [45]

4.1.1 Probability Density Function (PDF)

The Probability Density Function (PDF) is a distribution describing the probability of any random variable within the specified range of possible values, for instance the validation region. It is usually conditioned so that the sum of the distribution is equal to 1 (100%).

$$Pr(a \leq x \leq b) = \int_a^b p(x)dx = 1 \tag{4.1}$$

The Normal or Gaussian distribution as shown in figure 4.2 is common in natural processes. The PDF of the Normal distribution is given by

$$f(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{4.2}$$

where μ is the mean of the distribution and σ^2 is the variance of the data. The variance is given by the standard deviation σ squared, where the standard deviation $\pm\sigma$ contains 68.27% of the data and $\pm 2\sigma$ contains 95% of the data.

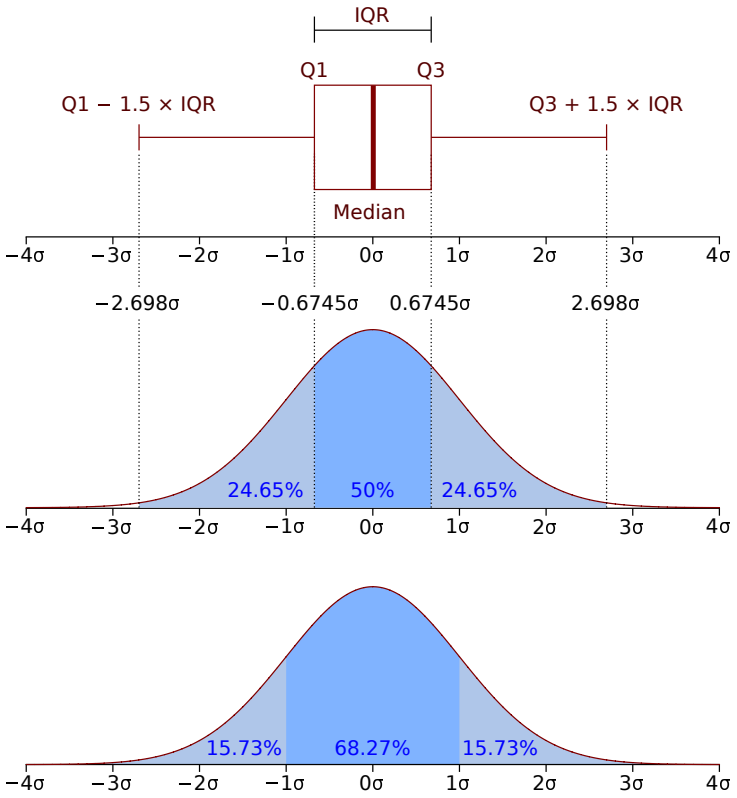


Figure 4.2: Boxplot and probability density function of a normal distribution $N(0, \sigma^2)$. Wikimedia Commons

Given that the sum of the distribution is constant, a smaller variance σ^2 from more accurate estimates or sensors will give a higher and narrower peak in the Normal distribution, giving lower uncertainty.

4.2 Data Association and Sensor Fusion

As long as there is only one object being tracked the problem formulation is relatively straightforward and one can use the probability of a correct detection, a True Positive (TP) in the further processing. It is important that the hypotheses consider the possibility that any detection does not belong to the tracked object. Detections might be a False Positive (FP), belong to another track or be new objects. The association methods used in Probabilistic Data Association Filter (PDAF) and IPDA etc. builds on these single target tracking problems, where for each target at most one of them is the actual target we want to track. In the PDAF any other measurement is considered noise/clutter even if this might be another vessel or track. All the measurements are however considered and a weighted average used to choose which measurement, if any, is used to update the track [46]. The IPDA is a continued development of the PDAF and introduce uncertainty about the existence of the track itself. Notice that these single-target methods can be used to track several targets, but the tracks are treated separately.

When multiple objects are being tracked simultaneously the complexity of the problem grows. It is not enough to match any detection to one track, but one must also match which detection belongs to which track. These problem formulations are sought solved using multi target tracking algorithms such as Joint Probabilistic Data Association (JPDA) and JIPDA. Both of which are further developments of the single target tracking models [46].

4.2.1 Target detection in noisy data

Data association is the process of associating measurements with inherent uncertainty to known prior tracks [46]. As described in the probability section the main problems that arise in data association is whether or not a detection is a true target, and the correct target. For each sensor a twofold problem arises: Is the detection a true object, a TP, or is it an error, a FP. If there is no detection we have the same conundrum; does no detection mean that there is no object, a True Negative (TN), or did we just not detect an object that really is there, a False Negative (FN). The classes can be gathered in a confusion matrix, splitting up in true/false and positive/negative as shown in figure 4.3. If the sensors are capable of

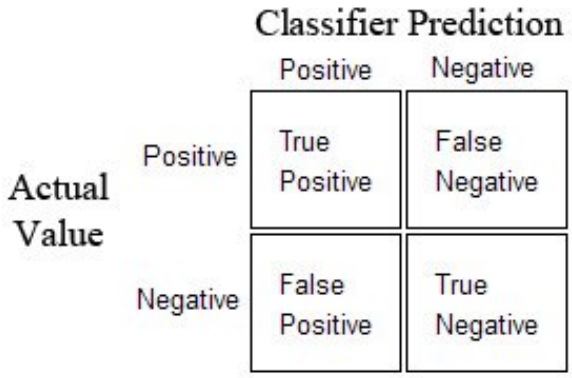


Figure 4.3: Example of the simplest confusion matrix

detecting several different objects the same problem appears for all of the classes, effectively increasing the size of the confusion matrix. When several sensors are used the big problem is whether or not the the detections arise from the same target and if they are all TP.

The probability of each of the four classes are dependant of the sensor used and the environment being observed. If no vessels are in the area we would like all of the detections to be TN. This might however not be the case in a real setting where noise and other factors could trigger erroneous detections. Erroneous detections due to for instance low Signal to Noise Ratio (SNR), clutter or multipath introduce noise in the estimates and must be taken care of in a suitable manner.

4.2.2 The Assignment Problem

The Kuhn–Munkres algorithm or ”Hungarian algorithm” is one way to solve the assignment problem [47]. It finds the optimal solution in $O(N^3)$ (polynomial) time, where N is the size of the largest size of the corresponding matrix. See figure 4.5 for an example of the matrix. Other methods include the ”Auction algorithm” [48], successive shortest path algorithms and Djikstra's algorithm [49].

In the context of target tracking for vessels in open waters, we will most of the time have separation between the different measurements and the track assignment will be relatively straight forward. However once there is any overlap between more than one measurement the task is harder. For instance in the camera sensor, or any other sensor which is limited by Line of Sight (LOS) detections, occlusion might blend two objects so that the furthermost one is lost.

The Jaccard index, also known as Intersection over Union (IoU), as shown in figure 4.4, is used here as a measure for the likelihood of one measurement belonging to one particular track. The reason for using the IoU compared to other statistical or probabilistic methods are due to simplicity as both the Re^3 tracker and the YOLO detector outputs similar bounding boxes [4][3]. The bounding boxes are easy to compare using IoU which directly outputs a measure of overlap between 0 and 1 [14].

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}, \quad \in \mathbb{R} \quad | \quad [0 \leq J(A,B) \leq 1] \tag{4.3}$$

A and B are bounding boxes from the Re^3 tracker and the YOLO detector respectively.

The RADAR does not provide any bounding box, only an angle which together with the output from the INS and the calculated horizon, might give a center point. This center point can however be considered as a bounding box with size zero. If this center point is padded, so that it has a size, it can be tested using IoU similarly to YOLO. This is implemented in the algorithm as a way to initialize and test the Re^3 tracker in lack of detections from YOLO. In future improvements the polygon which is output from the RADAR tracker can be used to set the dimension for the RADAR bounding box, but currently the size is determined by the distance to the vessel.

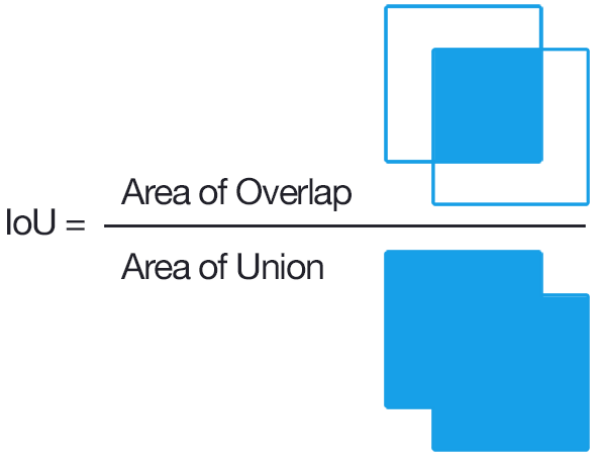


Figure 4.4: Intersection over Union visualized. Wikimedia Commons

See figure 4.5 for an illustration on how the matrix with 4 targets and 4 measurements could look like using the Kuhn-Munkres algorithm.

4.2.3 Kalman Filter (KF)

The KF is a linar state estimator that has built up a good track record since it was developed by Rudolf Kalman in 1960 [50]. The algorithm is used widespread in many applications and was early used and

-	A	B	C	D
a	0	0	0.2	**0.7**
b	**0.8**	0	0	0.1
c	0.1	0.2	**0.8**	0
d	0	**0.9**	0	0

$$BestSolution$$
$$(A, b) = 0.8$$
$$(B, d) = 0.9$$
$$(C, c) = 0.8$$
$$(D, a) = 0.7$$

Figure 4.5: The assignment problem illustrated with targets A,B,C,D, measurements a,b,c,d and IoU as values. Higher is better.

made famous in the Apollo space program [51]. The KF is proven to be the optimal estimator when the model is linear and known, with a Gaussian distribution, and the noise affecting the model is white. The covariance of the noise processes must also be known for optimality [52]. The model needs to be linear for the standard KF to be a feasible method. This world is inherently non-linear indicating that the KF often is infeasible or at least sub-optimal. Some non-linear problems might be modelled in such a way that it can be robustly linearized and then solved with the KF. Other non-linear problems cannot be solved using the standard KF and other methods must be used.

The discrete KF is a recursive algorithm that iteratively computes the estimated next state and the state covariance. The algorithm uses the (linearized) state and observation matrices F,G,H with the previous state $\hat{x}_{k-1}$ and measurements z_k as well as the previous state covariance P_{k-1} and the covariances for the process and measurement noise Q,R. The Kalman gain K is a correction term weighing the trust in the estimate versus the measurement and is used in calculating the new state estimate. Both the state and measurements of the real system is modeled with white noise affecting it; w,v.

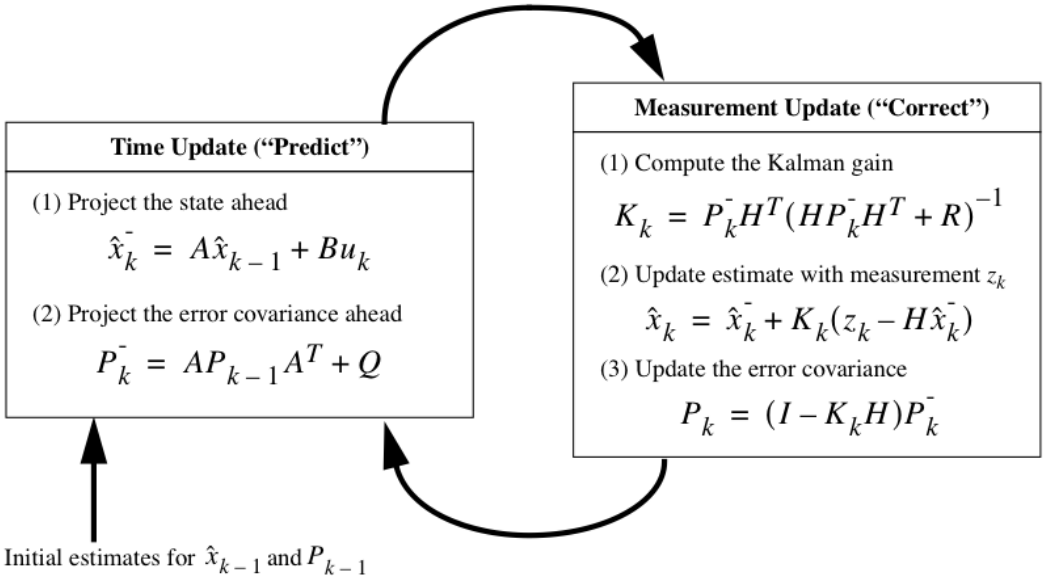


Figure 4.6: The iterative intuition of the Kalman Filter (KF). tex.stackexchange.com
(A and B are used in the figure instead of F and G respectively)

4.2.4 Extended Kalman Filter (EKF)

To solve non-linear problems which the basic KF is unable to estimate reliably, one must use other tools. The most common solution is the EKF which linearizes the state transition and observation models around an estimate of the current state. The models need to be differentiable functions, but not necessarily linear.

The EKF replaces the linear state and observation matrices with the Jacobian (partial derivatives) of their non-linear related functions. See table in figure 4.7. The EKF can diverge if the reference for which the linearization takes place is not good. Due to this it is imperative to have a good model of the system in question. Other non-linear methods which are much used include the Unscented Kalman Filter (UKF) and the Particle Filter (PF). New developments as seen in for instance the eXogenous Kalman Filter (XKF) opens up further use by combining a Globally Exponentially Stable (GES) observer and a linearized KF [53].

For further documentation about the inner workings of the KF and its many extensions, please see for example [8] [52][12][53] and [46].

The Extended Kalman Filter (EKF) algorithm

Non-linear System Description

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})\mathbf{x} + \mathbf{E}w$$

Observer Description

$$\hat{\mathbf{x}}_k = \mathbf{f}(\hat{\mathbf{x}}_{k-1})\hat{\mathbf{x}}_{k-1}$$
$$\mathbf{z}_k = \mathbf{h}(\hat{\mathbf{x}}_k, u_k)\hat{\mathbf{x}}_k$$

Jacobians

$$\mathbf{F}_k = \frac{\delta \mathbf{f}}{\delta \mathbf{x}}|_{\hat{\mathbf{x}}_{k-1}}$$
$$\mathbf{H}_k = \frac{\delta \mathbf{h}}{\delta \mathbf{x}}|_{\hat{\mathbf{x}}_k}$$

Time Update

$$\hat{\mathbf{x}}_k^- = \mathbf{f}(\hat{\mathbf{x}}_{k-1})\hat{\mathbf{x}}_{k-1}$$
$$\mathbf{P}_k^- = \mathbf{F}_k\mathbf{P}_{k-1}\mathbf{F}_k^T + \mathbf{Q}$$

Measurement Update

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^T(\mathbf{H}\mathbf{P}_k^-\mathbf{H}_k^T + \mathbf{R})^{-1}$$
$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_k^-)$$
$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^-$$

Figure 4.7: Overview of the Extended Kalman Filter (EKF). The time update and measurement update is recursive.

4.3 Fusion algorithm

The Extended Kalman Filter (EKF) is the foundation for the proposed fusion algorithm. It estimates the targets positions using the constant velocity model (white noise acceleration model) based on measurements from the RADAR tracker and detections in the cameras. The algorithm can take in any measurement from any source outputting the targets position in Cartesian coordinates. For each sensor the measurement noise parameters must be specified nevertheless. The state transition model is dependant of the vessel tracked, where slow moving vessels with low maneuverability can be estimated more robustly, whereas high speed and high maneuverability vessels must be estimated with sufficient uncertainty to adapt to rapid changes. Since the vessel being tracked is unknown, it is assumed that it is following a straight path at constant speed and have relatively low maneuverability. In future improvements one might use the AIS information received and adapt the model to different ship types. Other methods using for instance the Coordinated Turn (CT) model [46] might adapt better to changes in the vessel track.

The state vector is given using Cartesian coordinates and their velocities, in Northing and Easting.

$$\mathbf{x}_k = \begin{bmatrix} N \\ E \\ v_N \\ v_E \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \end{bmatrix} \tag{4.4}$$

The tracked vessel initial state $\mathbf{x}(0)$ is initialized using the position and velocity from the Autosea RADAR tracker ROS topic $/radar/estimates/$. The posterior state output is used as input.

$$\mathbf{x}(0) = \begin{bmatrix} N(0) \\ E(0) \\ v_N(0) \\ v_E(0) \end{bmatrix} = \begin{bmatrix} x_{posterior} \\ y_{posterior} \\ \dot{x}_{posterior} \\ \dot{y}_{posterior} \end{bmatrix} \tag{4.5}$$

The constant velocity state transition model is linear but is presented as a function for use in the EKF. T is the timestep between iterations in the tracking algorithm. The pipeline used runs at 50 Hz which indicates $T = 0.02$.

$$\mathbf{f}(\mathbf{x}_k) = \begin{bmatrix} x_k = x_{k-1} + T\dot{x}_{k-1} \\ y_k = y_{k-1} + T\dot{y}_{k-1} \\ \dot{x}_k = \dot{x}_{k-1} \\ \dot{y}_k = \dot{y}_{k-1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x}_{k-1} \tag{4.6}$$

The observation/measurement model for the camera is non-linear and is the reason we can not use the regular Kalman Filter (KF). The calculation of angle between estimated target position x_k y_k and ASV own position x_0 y_0 uses the inverse tangens function. The observation model for the RADAR from the Autosea tracker is simply the position given by the first two states in the state vector:

$$\mathbf{h}_{radar}(\mathbf{x}_k) = \begin{bmatrix} x_k \\ y_k \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \mathbf{x}_k \tag{4.7}$$

$$\mathbf{h}_{camera}(\mathbf{x}_k) = \begin{bmatrix} \phi_k = \arctan(\frac{y_k - y_0}{x_k - x_0}) \end{bmatrix} \tag{4.8}$$

The measurement vector from the RADAR is given by position in Cartesian coordinates. While the measurement vector from the camera is a scalar value of the polar angle (ϕ) between the tracked vessel and the ASV tracking vessel, given in NED.

$$\mathbf{z}_{radar_k} = \begin{bmatrix} x_{radar_k} \\ y_{radar_k} \end{bmatrix} \tag{4.9}$$

$$\mathbf{z}_{camera_k} = \begin{bmatrix} \phi_{camera_k} \end{bmatrix} \tag{4.10}$$

The initial covariance matrix $\mathbf{P}(0)$ is initialized using the posterior covariance for position and velocity output from the Autosea RADAR tracker. The correlations are ignored as of now.

$$\mathbf{P}(0) = \begin{bmatrix} var_x & cor_{xy} & 0 & 0 \\ cor_{xy} & var_y & 0 & 0 \\ 0 & 0 & var_{\dot{x}} & cor_{\dot{x}\dot{y}} \\ 0 & 0 & cor_{\dot{x}\dot{y}} & var_{\dot{y}} \end{bmatrix} \tag{4.11}$$

The process uncertainty matrix Q is given below. It is based on the standard Constant Velocity (CV) uncertainty model.

$$\mathbf{Q} = \sigma_{cv}^2 \begin{bmatrix} \frac{T^4}{4} & 0 & \frac{T^3}{2} & 0 \\ 0 & \frac{T^4}{4} & 0 & \frac{T^3}{2} \\ \frac{T^3}{2} & 0 & T^2 & 0 \\ 0 & \frac{T^3}{2} & 0 & T^2 \end{bmatrix} \tag{4.12}$$

Where T is the timestep between iterations. σ_{cv} is a gain dependent of the vessel in question. A small gain indicates high confidence in the process model. This can also be seen as a stable and non-manouvering vessel. Tracking high dynamic vessels must use higher gains.

$$\mathbf{R}_{radar} = \begin{bmatrix} var_x & cor_{xy} \\ cor_{xy} & var_y \end{bmatrix} \tag{4.13}$$

$$\mathbf{R}_{camera} = \begin{bmatrix} \sigma_c^2 \end{bmatrix} \tag{4.14}$$

The values in the $\mathbf{R}_{radar}$ matrix is given by the position covariance in the posterior output from the Autosea RADAR tracker. The covariance received is quite dynamic and would be in the range of a few meters up to above 100 meters, depending on the dynamics of both the tracked vessel and the ASV as well as distance and other disturbances.
$\mathbf{R}_{camera}$ (given by σ_c) is the estimated noise/uncertainty in the camera detections. The RADAR measurements are given in Cartesian coordinates and the camera detections are given as angles in polar coordinates. $\mathbf{R}_{camera}$ is also depending on whether the measurement originate from the YOLO detector or the Re^3 tracker. The detector is given highest confidence. See table 4.1.

A method inspired by [8] for manipulating the Kalman gain matrix K based on the measurement available gave promising results, but was ultimately abandoned since the state model was unable to adequately incorporate the non-linearities using this method. High σ_{cv} values of more than 2 was necessary for keeping the sensitivity to new measurements acceptable.

The values chosen after a multitude of tuning runs are given in table 4.1. The values are chosen as a middle way between a stable estimated trajectory and sensitivity to measurement input.

T	Adaptive, actual iteration time used
σ_{cv}	0.1
σ_c detector	0.01
σ_c tracker	0.02

Table 4.1: Parameters used in the EKF tracking.

4.4 Other modules used

The detection and tracking pipeline is built up modularly using the Robot Operating System (ROS) framework. This makes it relatively easy to replace different modules when better versions are developed. It also makes it possible to run different parts of the system on separate or dedicated hardware. The most

hardware/processing intensive module used in this pipeline is the YOLOv3 image detector which relies heavily on a powerful GPU. The Re^3 tracker is also magnitudes faster using GPU processing power. Both of these modules should therefore advantageously be run on dedicated GPU hardware.

4.4.1 Kongsberg Seapath Inertial Navigation System (INS)

The ASV pose is given from the INS as a pose vector with 3D position and 3D attitude given in a quaternion representation.

$$\mathbf{X}_{ins}(t) = \begin{bmatrix} x_0(t) \\ y_0(t) \\ z_0(t) \\ q_1(t) \\ q_2(t) \\ q_3(t) \\ q_4(t) \end{bmatrix} \tag{4.15}$$

In the $/seapath/pose$ ROS topic the position is given in NED frame with Munkholmen island as origin and with the WGS84 ellipsoid as height reference. The attitude in quaternions are compuationally efficient and avoids potential singularities that could affect Euler angle representations. For further use the angles are however converted to Euler angles (Roll, Pitch and Heading). The height, $z(t)$, or any heave is also not used as the vessels operate on the 2D ocean surface, and any height motion would be negligible to the used sensor outputs.
The INS have a high output rate of 100 Hz and the INS GNSS-based timestamp is used to timetag and synchronise the ROS network.

4.4.2 Autosea RADAR tracker

The Autosea tracker is a full functioning RADAR tracker based on the IPDA filter. It performs collection and clustering of RADAR-spokes together with filtering and tracking of these.

The raw RADAR data is output as a RADAR spoke with an array of 1024 return amplitudes $\mathbf{A}$ equally spaced out with $\Delta|\bar{\mathbf{r}}|$ along one ray of azimuth ϕ out to maximum range setting. See figure 4.8 for an illustration of a RADAR spoke.

$$z_{radar-spoke}(t) = \begin{bmatrix} \phi(t) \\ \Delta|\bar{\mathbf{r}}|(t) \\ \mathbf{A}(t) \end{bmatrix} \tag{4.16}$$

The RADAR data is pre-processed in the Autosea ROS library and only the detections of potential vessels remain. For the list of tracked objects an array $\mathbf{p}_{centroid} = [x, y, z]^T$ of the centroids and a matrix $\mathbf{P}_{polygon} = [[x1, y1, z1]^T, ..., [xi, yi, zi]^T]$ containing the points of the associated polygon is output. The output from the RADAR tracker is given in Cartesian NED coordinates, also with the Munkholmen island as origo.

$$z_{radar-estimates}(t) = \begin{bmatrix} \mathbf{p}_{centroid}(t) \\ \mathbf{P}_{polygon}(t) \end{bmatrix} \tag{4.17}$$

The tracker also output estimated velocity and uncertainty in both position and velocity, as well as covariances between them, for each tracked object. The inner workings of the tracker is not further described here. See papers from Kufoalor and Wilthil for details [5][54][6].

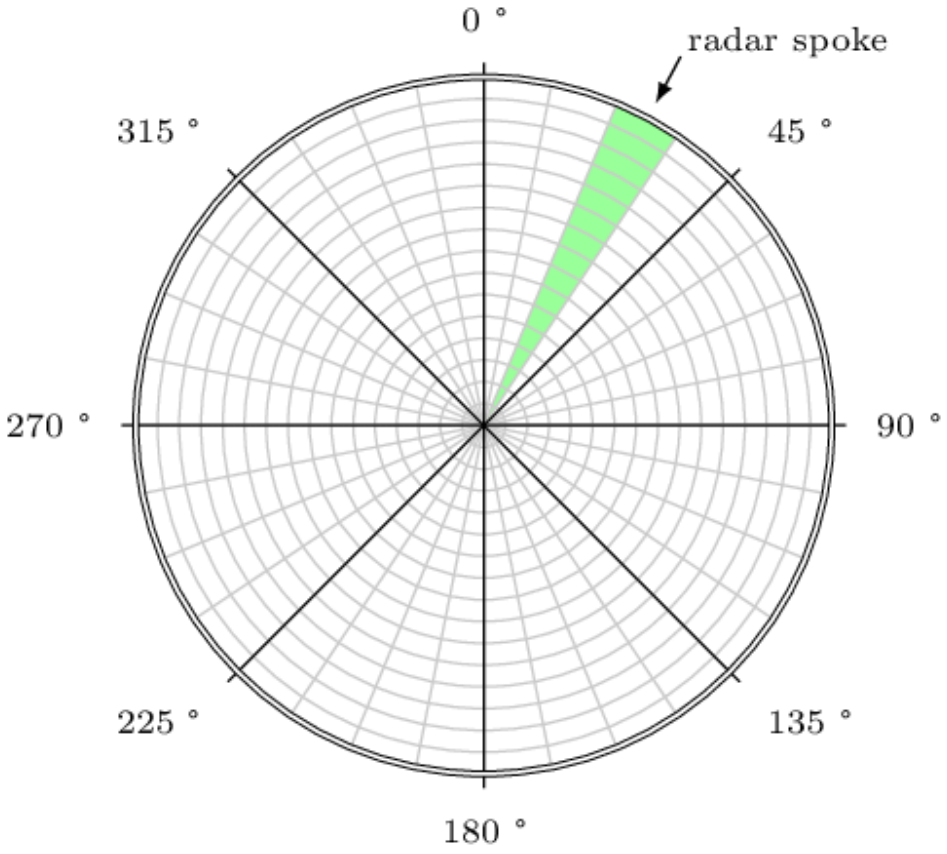


Figure 4.8: An illustration of a RADAR spoke, with cells split along a ray of fixed angular resolution

4.4.3 Image detections

The image detector outputs the coordinates of the bounding boxes created by the YOLO algorithm around detected objects.

$$z_{detector}(t) = \begin{bmatrix} \mathbf{P'}_{\mathbf{detector}BBox}(t) \\ \beta_a(t) \\ \sigma_a(t) \end{bmatrix} \tag{4.18}$$

The coordinates are given in image plane $\mathbf{p'} = [u, v]^T$ which can be used directly by the image tracker. The confidence score $\sigma_a(t)$ of the bounding box, indicating how certain the algorithm is about the detected object type $\beta_a(t)$, is also provided.

$$z_{tracker}(t) = \left[\mathbf{P'}_{\mathbf{tracker}BBox}(t)\right] \tag{4.19}$$

The image tracker outputs the coordinates of the bounding boxes continuously recomputed by the Re^3 algorithm around the tracked objects. The coordinates are again given in the image plane $\mathbf{p'} = [u, v]^T$.

Based on the pixel coordinate and the viewing angle and FOV of the camera in question, the angle between the ASV and the tracked vessel is calculated in the sensor fusion algorithm.

Chapter 5

Tracking Pipeline

The described pipeline in figure 5.1 focuses on modularity, ease of implementation, speed and accuracy. In the demonstrated setup described below, we use an Extended Kalman Filter (EKF) to merge measurements and estimate the tracks for several tracked vessels using RADAR and EO Daylight cameras. The tracks are initialized from the track ID, position, speed and covariance output by the Autosea RADAR tracker for each target.

Since the output from neither the YOLOv3 [3] visual object detector nor the Re^3 [4] image tracker includes any covariance or other uncertainty information regarding position, classical fusion of estimates is difficult [12]. Both the YOLO visual object detector and the Re^3 visual object tracker outputs bounding boxes, containing the detected and tracked vessel respectively. YOLOv3 provide indication of certainty for the detected object type, but no indication of accuracy of the bounding box position. Re^3 has no indication of certainty for neither tracked object type nor position of the bounding box.

To provide a mean of comparing the RADAR with the two visual modules we propose to create another bounding box in the image around the horizon and in the direction of the RADAR detection. This bounding box is padded to a size depending on the range to the detected object. In future improvements it can also be based on the size given by the polygon output together with the centroid of the tracked vessel from the Autosea RADAR tracker.

Given that all detections now have a bounding box, they can be compared using the Intersection over Union (IoU) [14], see figure 4.4. The calculated IoU value will then be used to initialize or update the Re^3 visual object tracker based on detections in the YOLOv3 visual object detector or the RADAR detections from the Autosea RADAR tracker.

Based on available information, this comparison method using IoU, between camera and RADAR or other passive or active sensors, is an original idea.

The proposed pipeline does not try to estimate the tracked objects position from camera observations only. The EKF is therefore not initialized before a tracked target is received from the Autosea RADAR tracker. After the track is initialized in the EKF, the subsequent outputs by the RADAR tracker is considered as advanced measurements including updated uncertainty. Based on the initialized track position, the angle to the other vessel is calculated, first in NED and rotated to a BODY fixed angle. The angle is further used to extract the correct camera and pixel location from where a part of the image is cropped out. This cropped image is next used in the image detector and tracker modules. Other measurements in Cartesian coordinates, and given a known or estimated uncertainty, like LiDAR or AIS can easily be combined and fused in the EKF-based algorithm.

The pipeline using deep learning based methods for both the image detector and the image/video tracker provides flexible possibilities for training both methods on the concrete data that is likely to be encountered in the area of interest. Boats around the world have quite large changes in appearance and one might not need to reliably detect unlikely boat shapes at a distance. Due to the module based structure of the pipeline it is possible to replace any of the modules if better versions become available. As of time of writing both the Re^3 tracker and the YOLOv3 detector are state-of-the-art in terms of speed and reliability. The RADAR tracker from the Autosea project may also be updated and the physical RADAR be changed as long as the output from the library remains the same.

Another possibility that would be interesting to pursue is to fuse the convolutional networks in the Re^3 tracker and the YOLOv3 detector, or similar modules. This would reduce the need for processing the images multiple times, and should give rise to increased detection and tracking speeds. It would also circumvent the problem of serialization of processes due to lack of multithreading possibilities in the GPU. However this requires algorithms currently unavailable and this option is not given further attention in the thesis.

5.1 The Autosea RADAR tracker

The RADAR tracker devoloped in the Autosea project by PhD students is a robust standalone package performing all necessary processing of the RADAR data. The Autosea repository is written primarily in Python and uses the ROS framework for interfacing with other sensors and actuators. The tracker utilizes the position and heading of the ASV together with the raw RADAR data output as RADAR spokes, see equation 4.16.

The following description is not meant to describe the tracker in detail, just an overview of the functionality. Under the hood the tracker performs clustering of close together detections and conversion from polar coordinates to NED coordinates. The detections are then compared to a map database from the Norwegian mapping authorities and land detections are removed. The remaining clusters are initialized

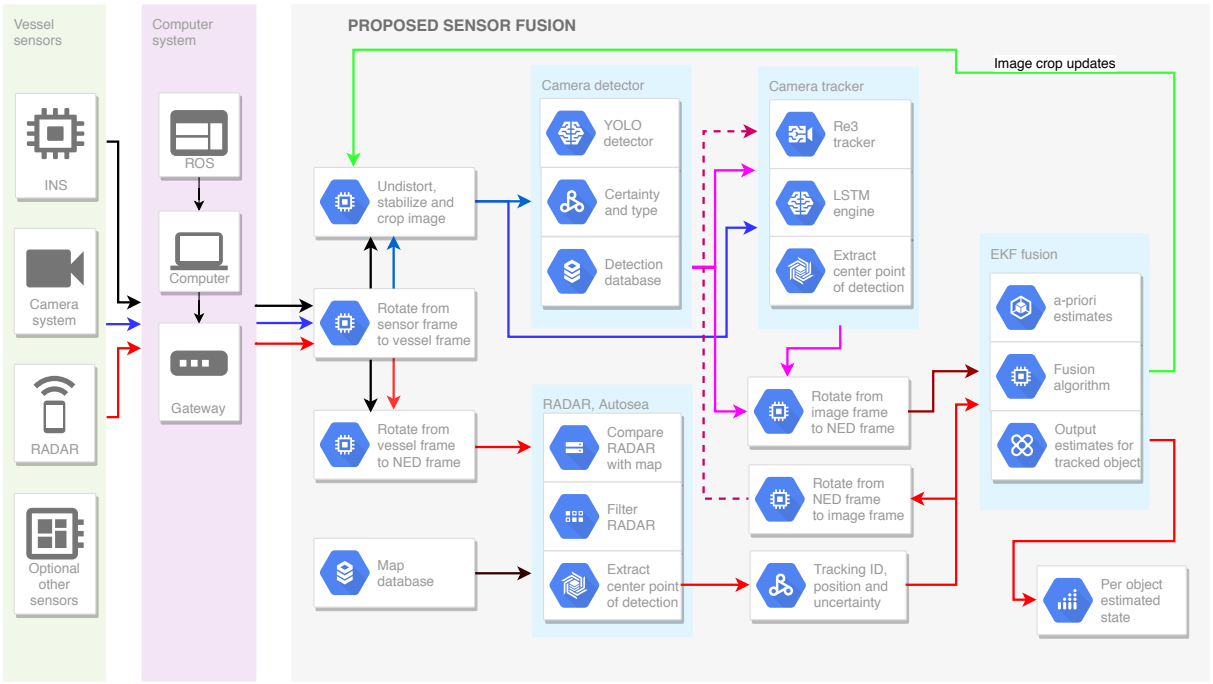


Figure 5.1: Sensor Fusion pipeline

as targets, using the Integrated Probabilistic Data Association (IPDA) filter or the PDAF method for associating measurements to tracks. The track initialization and track termination is handled within the tracker algorithm and these tracks are used further in the EKF fusion algorithm proposed here.

The clustered detections, not filtered out as land and initialized as trackes, are output using ROS as rostopics with a cluster center point and a polygon enclosing each object. The output also includes a unique ID per track and the estimated speed of the tracked object as well as the uncertainty of the track, both in position and speed.

The IPDA filter [55] is a further development to the PDAF[45][56]. The properties of these association methods will not be further detailed in this report, but their implementation can be studied in the papers by Kufoalor et al. [54][5] and Wilthil et al. [6].

The RADAR data used for the experiments in this thesis are collected during a real-time tracking with the IPDA based tracker in the autumn of 2018.

5.2 Synchronization

Good synchronization is key to good data quality. If the exact moment which data is collected is not known this introduces an unknown uncertainty. The cause and affect of such poor synchronization could be different for each sensor. A RADAR would have the wrong angle information giving large position error on targets. The camera would not have the correct image at the given time and the INS would provide wrong attitude. An example of delayed image data under stabilization is shown in figure 5.2. The lack of synchronization between the INS and the camera transforms the image with the true horizon substantially below the center of the image (red line). As can also be seen, the blue vertical line indicating the tracked vessel does not match the vessel in the image.

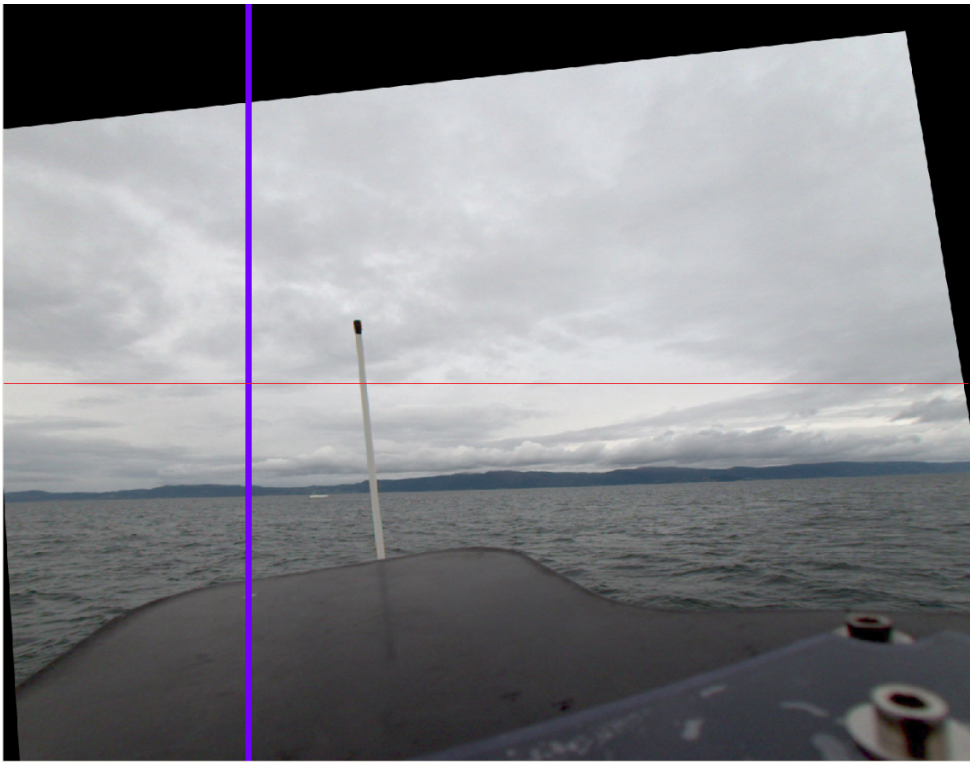


Figure 5.2: An example of poorly synchronized image and INS

On the system used for the tests all sensory data is time tagged using GNSS time from the INS and the accurate time when the data was collected is embedded in the data available from the ROS node. The Ladybug camera system used also support 1 Pulse Per Second (1PPS) which is an extremely accurate electrical pulse output from the Seapath INS. This pulse is used to further synchronize the time string given as NMEA standard messages such as GPGGA or GPZDA. These messages give different information about GNSS status from the INS as well as time. Different messages give out position, heading, attitude and so forth.

The output from the Autosea RADAR tracker is considered to have no time delay even if the RADAR measurements which it builds on have such. The reason for ignoring these temporal correlations is that the posterior output of the Autosea tracker is already estimated for the current timestamp. How the Autosea tracker handles the uncertainty of received RADAR data is not detailed here.

A better handling of synchronization is needed for a real-world COLAV system where it is critical that data is received and processed using the correct time information. Some form of verification of good synchronization must therefore be enforced to have a reliable and robust COLAV system. Predominant use of 1PPS with GNSS time strings are sensible.

5.3 Image Stabilization

The Ladybug camera system is interfaced into ROS using a library developed by Autoware [57] and the 5 relevant images are easily extracted from the ROS node. Each camera is output on separate ros-topics. The raw images output from the Ladybug camera are uncalibrated and are first rectified using the calibration parameters found from calibration [28]. See equations 2.12 through 2.23.

Using the accurate INS system onboard the ASV, the calibrated and rectified camera images can be rotated and transformed into a projection such that the horizon is always a straight line in the middle of the image. This helps minimize the further errors when the image or parts of the image is fed to the YOLO detector and the Re^3 tracker. Stabilizing the image in such a way dramatically reduces the number of pixels a visible boat will translate if the ASV is subject to rough seas or motion due to maneuvers or other. Ideally only heading changes and forward speed of the ASV as well as motion of the tracked vessel will be affecting the image around the horizon.

The stabilization of the image requires several transformations and rotations as well as a projection of the image into the new frame. Firstly the common Rotation matrix constructed of the three principal rotations, roll (ϕ), pitch (θ) and heading relative to the vessel (ψ). See equation 2.1 and 2.3. The matrix operations are done in Homogeneous coordinates and the matrix is therefore expanded to 4 dimensions. This matrix must be transformed so that the motion measured in vessel BODY frame is mapped to the image frame using a transformation matrix $\mathbf{T}_b^i$.

$$\mathbf{R}(\Phi) = \mathbf{R}(\phi)\mathbf{R}(\theta)\mathbf{R}(\psi) = \begin{bmatrix} c\psi c\theta & -s\psi c\phi + c\psi s\theta s\phi & s\psi s\phi + c\psi c\phi s\theta & 0 \\ s\psi c\theta & c\psi c\phi + s\phi s\theta s\psi & -c\psi s\theta + s\theta s\psi c\phi & 0 \\ -s\theta & c\theta s\phi & c\theta c\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.1}$$

$$\Phi = \begin{bmatrix} \phi \\ \theta \\ \psi \\ 1 \end{bmatrix} \tag{5.2}$$

$$\mathbf{T}_b^i = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.3}$$

$$\mathbf{G}(\Phi) = \mathbf{T}_b^i \mathbf{R}(\Phi) \mathbf{T}_i^b = \mathbf{T}_b^i \mathbf{R}(\Phi) {\mathbf{T}_b^i}^T \tag{5.4}$$

$$\tag{5.5}$$

The pixel coordinates in 2 Dimensional (2D) image space must be mapped to 3D coordinates and back again. This is done using projection matrices $\mathbf{A_1}$ and $\mathbf{A_2}$

$$\mathbf{A_1} = \begin{bmatrix} 1 & 0 & -w/2 \\ 0 & 1 & -h/2 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \tag{5.6}$$

$$\mathbf{A_2} = \begin{bmatrix} f & 0 & w/2 & 0 \\ 0 & f & h/2 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{5.7}$$

Where f is the focal length of the camera in pixels and w, h is the image width and height in pixels.

To align the center of the image with the horizon and also rotate the adjusted image back into the heading angle of the image plane, another pair of rotation and translation matrices are needed.

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.8}$$

$$\mathbf{R}_y, i(\psi) = \begin{bmatrix} c-\psi & 0 & s-\psi & 0 \\ 0 & 1 & 0 & 0 \\ -s-\psi & 0 & c-\psi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.9}$$

Here $\mathbf{R}_y, i$ takes the negative heading angle relative to the vessel body to rotate the image back around the image y-axis. The translation values dx, dy and dz are given in pixels. dx and dz are zero in this setup, while dy is changing based on the heading angle of the camera relative to the vessel frame and the roll and pitch values. The number of pixels are calculated as a function of the angles.

$$dy = \frac{\phi}{\alpha}\sin(\psi) - \frac{\theta}{\alpha}\cos(\psi) \tag{5.10}$$

$$\alpha = \frac{FOV}{h} \tag{5.11}$$

ψ is here the heading angle of the camera in question relative to the vessel body frame. ϕ and θ is roll and pitch respectively. All angles given in radians. α is a factor constant given by the FOV and height of the image in pixels.

The total equation for stabilizing the image based on the roll and pich values reported by the vessel's INS system is given as:

$$\mathbf{H} = \mathbf{A_2}\mathbf{T}\mathbf{R}_y, i(\psi)\mathbf{G}(\Phi)\mathbf{A_1} \tag{5.12}$$

The images are transformed and stabilized using the matrix $\mathbf{H}$ and the OpenCV method "cv2.warpPerspective" which performs a 3D transformation of the matrix-like image-data.

Figure 5.3: An example of a stabilized image

5.4 Image extraction around RADAR detections

To be able to detect vessels and limit the necessary processing of the images and keep the pipeline running in realtime, only the part around the horizon and in the direction of RADAR detections are used

as input to the YOLO detector and Re^3 tracker. The horizon is the center of the stabilized image and is easily extracted. The RADAR detection is converted to relative angle in NED coordinates between the tracked vessel and the ASV, and further to an angle and pixel location in body frame. The crop size could be determined by the range to the tracked target and also be dynamic to adapt to different vessel sizes. For now it is however fixed at 1/6 the height and width of the corrected and stabilized image.

Figure 5.4: An example of a cropped image around the horizon, with the Re^3 tracker tracking the target

Which camera the image is cropped from is based on the angle to the detection in BODY frame. The forward looking camera (camera 0) is primary. An angle beyond $\pm 40°$ is cropped from the next camera (camera 1 or 4 respectively). Camera 2 or 3 are chosen if the angle is above $\pm 112°$, and the shift between them happens at $\pm 180°$. See figure 5.5 and 2.8 for an overview of the overlapping FOV from the 5 cameras. Each camera is separated by 72°, (360°/5 cameras).

5.5 Image Detections using YOLOv3

A ROS node built to support YOLO was used during the tests [33]. This gives a flexibility of multi-threading so that the main camera tracking node is not occupied while the detections are performed. The YOLOv3 detector is out of the box trained for detecting 80 objects, among these are boats. It is this pretrained network that is used throughout the tests. During these tests the YOLOv3 detector found the target boat in the cropped image at a distance covering as little as 5×10 pixels, or 0.2% of the height and 0.5% of the width of the raw image. The wide FOV and resolution of the Ladybug cameras makes it possible to detect a 10 meter long boat about 500 meters away in the cropped image. Larger vessels will be detectable further out, whilst small boats, kayaks etc will need to be closer. Cameras with smaller FOV or higher resolution might give even longer ranges. Lighting conditions, vessel type, visibility and backgrounds all play a big role in the detection performance.

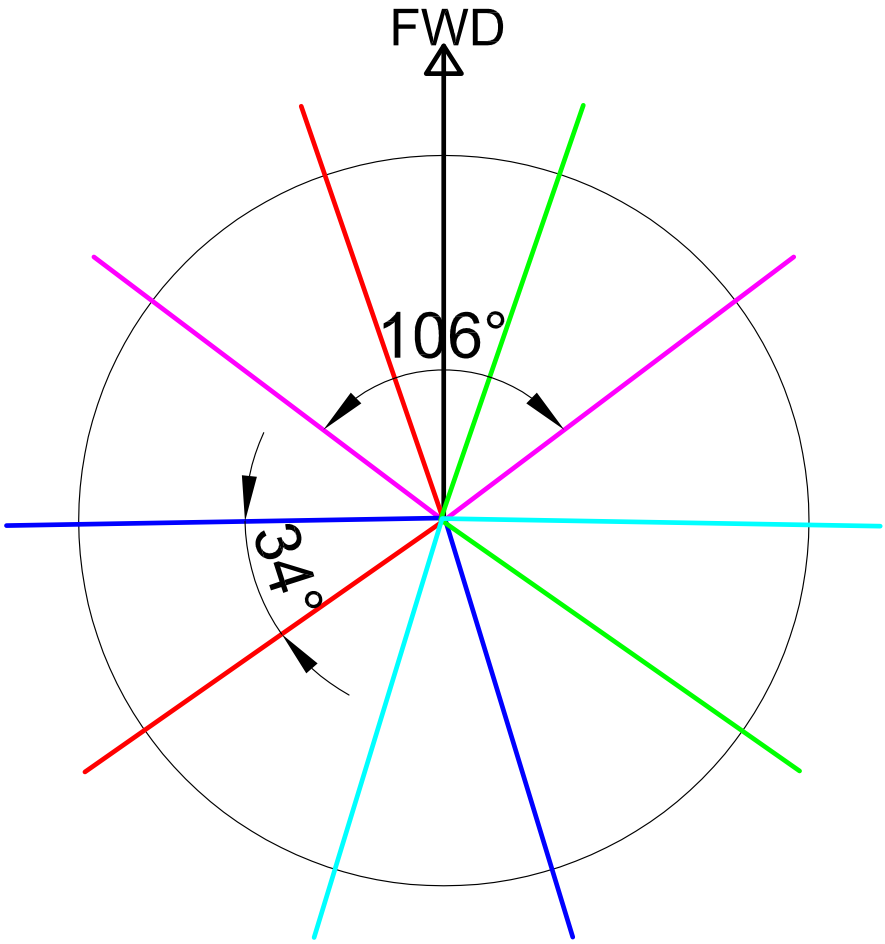


Figure 5.5: The overlapping FOV of the 5 cameras in the Ladybug 5+ camera covering the horizon. Camera 0 (purple) is looking forward, followed clockwise by camera 1 (green) etc.

The detector outputs a nested array containing arrays of bounding boxes for all detected objects. The array includes object type and certainty as well as the corners of the bounding box. See vector in 4.18. Since the detector and the Re^3 tracker works om the same images, only an adjustment for attitude changes in the ASV in the time taken for processing the image is performed. The bounding boxes that contain boats are then output on a ROS topic for the other modules to use. The detector processes each image separately and has no recurrent network or memory. The output changes therefore between image frames. This could include the order of detections if several objects are found.

5.6 Image Tracking using Re^3

For each object to be tracked the image tracker takes in a bounding box array with the 4 corners and a unique name as parameters. The bounding box is updated and refined on every image frame by the Re^3 tracker. The tracker will normally be initialized using the bounding boxes from the YOLOv3 detector. This is the most robust method described in this thesis for using the Re^3 tracker and is the primary method.

However if no detections are made from the YOLO detector, or perhaps if that module is not used, the Re^3 tracker can be initialized by the RADAR detections. It is assumed that the RADAR-only detections

is some distance away. The bounding box would then be initialized around the horizon in the direction of the RADAR detection. The bounding box would be set to a quadratic size depending on the distance between the vessels. The center of the stabilized and cropped image is the horizon and angle towards the EKF estimated position of the vessel. The RADAR detection should therefore be very close and the small delta angle between RADAR and estimate is calculated as the center point for the padded RADAR "bounding box". Provided that the stabilization from the INS is sufficiently good and the target have good visibility, the method initialize Re^3 robustly. Since the tracker is pretrained on a large dataset of objects, it starts to follow features in the image that stands out, such as boats on the sea, even if there is no detector present. A warning should be mentioned here; should the tracker be started around the wrong target, and especially before the EKF is stable, this could lead to divergence and large errors in the estimates.

Given that the visually tracked object may not be verified by a detector or, even if detected, not be the correct object, one cannot put too much emphasis on the output of the tracker alone. This might give rise to an overconfident state transition model which would not sufficiently take updates from other, less frequent measurements/sensors. The track from Re^3 is therefore not accepted as a measurement into the EKF before it is validated by the YOLO detector or the RADAR. The track-before-detect method inspired by developments in RADAR [58], is here used to only rely on the visual tracker when the situation is sufficiently stable. It is critical to avoid feedback from erroneous targets through the high output-rate Re^3 tracker.

To mitigate some of the uncertainty from using the image tracker, several limitations have been put in place. This applies to initializations from both the object detector and the RADAR.
Firstly the tracker is not used if the distance to the target is too large, currently set to 600 meters. Secondly the tracker terminates if no updates are received within a specified time period, currently tested between 20 and 100 received image frames (2-10 seconds). Also the angle corresponding to the center of the output bounding box, must match either the RADAR detection or the estimate from the EKF. This limit is currently set to $\pm 4°$. To further reduce possible erroneous detections due to drift of the bounding box, the bounding box is deleted should the horizon not lie within the bounding box. This coincides with a bounding box not around the center of the image height. A good stabilization of the image is therefore important. To further catch drift, the bounding box is also deleted should it touch any edge of the cropped image or if it grows larger than half the size of the cropped image in either direction.

This option to use the RADAR to initialize and confirm the tracker is shown as a dotted line to the Re^3 tracker in figure 5.1. In a later implementation the initialization can be evolved into using the polygon from the Autosea tracker as size of the bounding box. It is now implemented such that the distance to the target is used to adjust the size of the bounding box. The closer the target vessel is, the larger the bounding box.

After the tracker is initialized and is tracking the target, new detections from the YOLOv3 detector and RADAR are used to monitor and verify the tracking performance. This is calculated using IoU [14], which gives a measure for how good the two bounding boxes align. The IoU value is used to evaluate whether or not the tracker is performing sufficiently good. It is preferable to not uncritically reinitialize the tracker every time a new detection is made due to the LSTM network structure adapting to the scene. An IoU value below a certain threshold would be reason to reinitialize the tracker bounding box. The IoU threshold is currently set to 0.5. A special case is where the IoU is zero as this indicates no overlap. This might indicate that the tracker has lost the vessel, an erroneous detection by YOLO or it could be a new/different object to be tracked. Currently this is handled as if the tracker is wrong, but more advanced handling could bring further improved tracking performance, especially in a multi-target tracking scenario.

The Re^3 image tracker is primarily written to accept one tracked object, but it is also adapted to accept

more tracked objects. In a crowded harbour environment the multi-object tracking might be critical, however for such a scenario a LiDAR sensor might be better suited than the RADAR to complement the camera. At the moment the algorithm accepts maximum one object to track in each window cropped from the RADAR detections.

The tracker needs a continuous feed of stable cropped images from the same angle. Therefore, to track several objects around the vessel, one instance of the Re^3 tracker must be started per object. If multi-target tracking in Re^3 is implemented one instance per cropped image would suffice.

For now the algorithm takes in at the most one target per camera. Giving a maximum of 5 tracked objects. The Re^3 tracker utilizes 1350 MB of GPU memory (RAM) per instance, and the memory available might limit the number of simultaneous trackers. On the Dell laptop used for the tests, see table 6.4, a maximum of two Re^3 trackers can run simultaneously with the YOLOv3 detector. Alternatively the Re^3 tracker can use the CPU only, but the performance is substantially reduced. On powerful hardware this might however still be enough to achieve realtime performance.

5.7 Tracking of vessels using active RADAR and passive cameras

Using the algorithm explained in section 4.3 and shown in figure 5.1, the targets found by the Autosea RADAR tracker is tracked and fused by the Author's EKF pipeline. The position, speed and uncertainty published by the Autosea tracker speeds up initialization of the EKF by providing a reliable initial state. When further measurements are received as Cartesian coordinates from RADAR or AIS, or as angle from camera detections these are used to update the estimates within the EKF.

The EKF parameters are tuned as well as time and available data permitted, but still indicate much room for improvements. The RADAR is considered to have a perfect output as the covariance in position from the Autosea tracker is used directly as the RADAR R matrix.

Measurements output by a LiDAR tracker as clustered centroids, similar to how the Autosea RADAR tracker publishes targets, can be implemented. Importing measurements in Polar coordinates with angle and range directly has not been attempted, but should be feasible as long as the necessary precaution to handle non-linearities and measurement uncertainty is handled.

Throughout the tests described in chapter 6, only RADAR and EO cameras are used as sensor input. The AIS have not been used as input to the algorithm, and is only shown in the plots as a possible Ground Truth. The AIS have previously been implemented and other tests show further increase in the accuracy of the algorithm. A possible slight delay can be observed in the AIS data, and in future it might be advantageous to use more non-linear models with high confidence for across track accuracy and less confidence along track due to possible time delays.

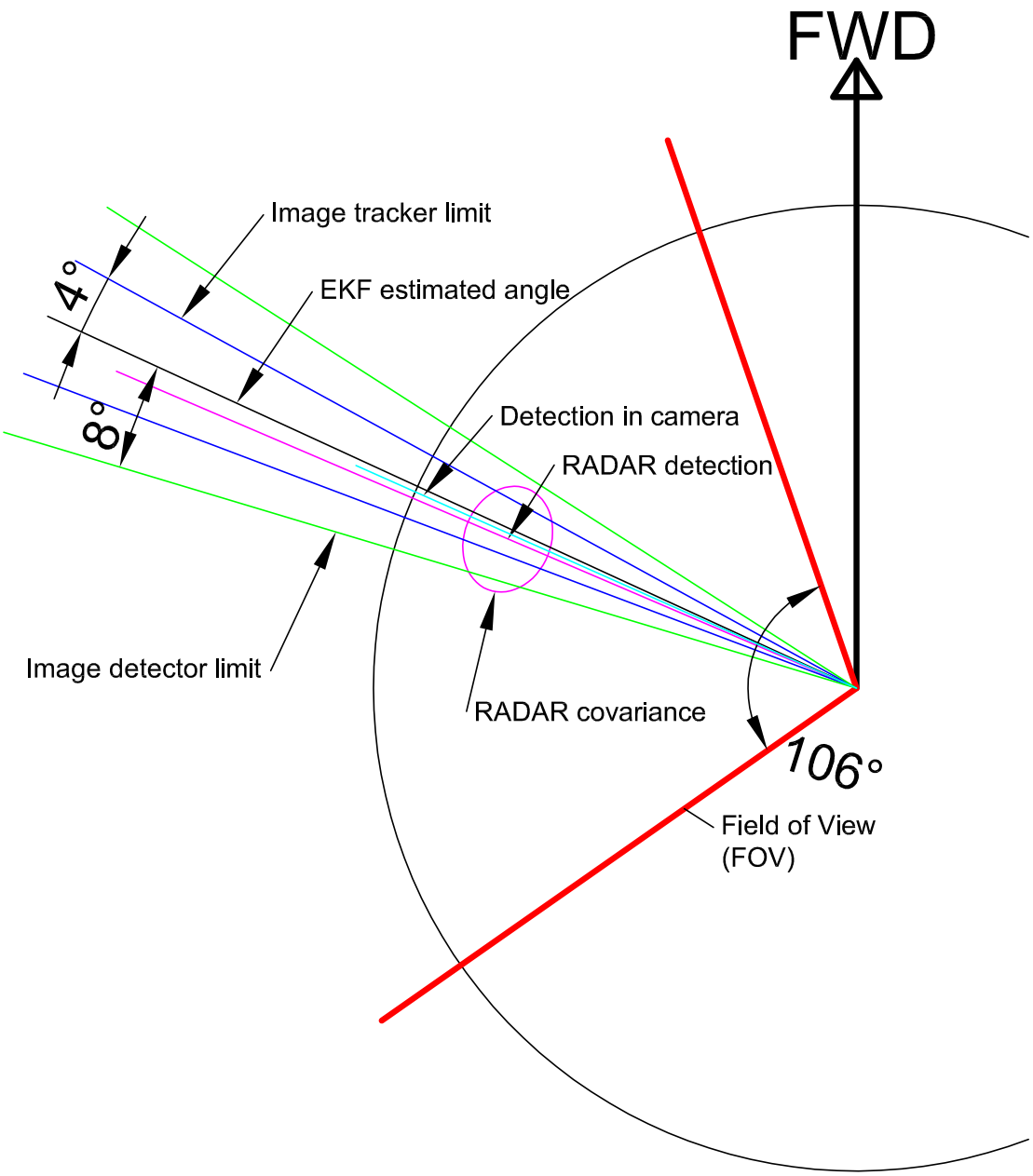


Figure 5.6: Combined detections with RADAR and the Ladybug camera using Re^3 tracker and YOLOv3 detector

Chapter 6

Tests and Results

Most of the tests done in this thesis are performed on data recorded close to the island of Munkholmen, north of Trondheim, during an experiment in the Autosea project in the last week of september 2018, by several PhD students at NTNU together with the author. The Maritime Robotics ASV ready boat ”Telemetron”, see figure 6.1, was used as platform for the experiments. One ASV from Kongsberg, ”Ocean Space Drone 1” as seen in figure 6.1, was used as a target. On the first day also the tug ”Munkholmen II” from Trondheim harbor was used as a target, but no image data was recorded that day.

During the experiments, data from the RADAR, INS and AIS systems were collected and stored in rosbags using ROS. Also the Autosea RADAR tracker was utilized, and its output stored during the experiments, and it is these tracks the further sensor fusion is performed on. The image data was unfortunately impossible to collect and store in ROS at that time due to a bug in the Linux driver for the Ladybug camera from FLIR. The image data was therefore recorded on a Windows laptop using the provided LadybugCapPro software. The recordings are later exported as images with separate folders for each camera, and with a timestamp in the filename.

The Ladybug camera system was mounted on top of the RADAR using a custom standoff/bracket made from plastic, which Maritime Robotics had used previously for other sensors. Similar to what is seen in figure 6.1.

The experiments was arranged such that the target was following tracks with fixed heading and speed. Several algorithms and scenarios were tested out for autonomous collision avoidance. Telemetron would then autonomously perform several maneuvers based on the relative angle between the target’s course and Telemetron’s course. Different scenarios was introduced including Head-on, Overtaking, Crossing from starboard and Crossing from port. Virtual fixed obstacles was also used to safely test out difficult scenarios close to land. See the article from Kufoalor et al. [5] for more details about the experiments.

6.1 Interfacing the Ladybug

The Ladybug 5+ camera is interfaced to the recording computer using Universal Serial Bus (USB) version 3. The camera delivers a massive data-rate of up to 400 MB/s, primarily limited by the USB3 bandwidth. It is therefore not feasible to store long time series of image data at high frame-rate. Image data was recorded until the storage capacity on the laptop was full. 340 GB of image data was recorded with a frame-rate of 5 and 10 frames per second. After export to images a total of 40.000 images was available on each of the 6 cameras.

Figure 6.1: ”Telemetron” (top) and ”Ocean Space Drone 1” used in the experiments

Note that for possible future recording, the images should be recorded with ”full height” setting. The ”half height” setting cropped the image in width (strangely enough) due to the fact that the image was exported with a 90 degree rotation. The post processing of rotating and scaling 200.000 images took a while. The setting reduced the data-rate and required storage capacity, but negatively impacted the image quality after scaling due to missing pixel information.

6.1.1 Time-Synchronization

The first task that was completed was to get the time synchronization up and running. This was done as part of the author’s specialization project [1]. The Ladybug 5+ camera has an auxiliary port used for power and interfacing with the camera. Data is output over a shielded USB 3 cable. Previous experiments

had shown that USB3 was interfering with GNSS reception, but probably due to the shielded design of the Ladybug this was not an issue in the tests. The auxiliary port has input for GNSS and time data using the common National Marine Electronics Association (NMEA) sentences as well as a dedicated signal wire for 1PPS. The usage of time strings together with 1PPS is common in areas where accurate time-stamping of data is necessary such as in bathymetric data acquisition. The NMEA data is commonly output from a navigational sensor system, such as the Kongsberg Seapath which is used in this experiments, as Electronic Industries Association (EIA) Recommended Standard 232 (RS-232) compatible signals. The RS-232 standard defines voltage levels which are not compatible with the Ladybug interface which only supports Transistor–Transistor Logic (TTL) voltage levels. A converter module had to be implemented between the Seapath and the Ladybug. This was implemented using a RS232-to-TTL converter board which Maritime Robotics had in store. Power to the converter was taken from a USB port on the Seapath as the board required +5 Volts and ground (GND) externally. The reason for using the USB as power was to remove any problems with different ground potentials that could potentially destroy the circuits. Initially power was tried extracted from the RS-232 port, but this only provided +3 Volts which was to low.

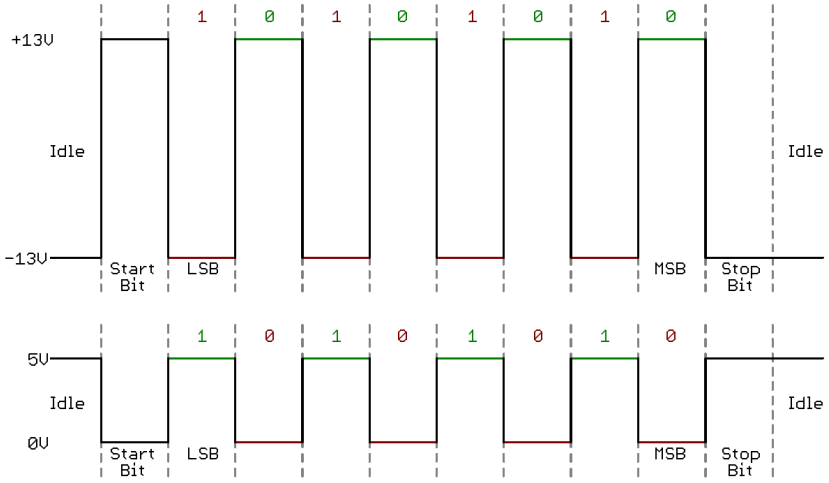


Figure 6.2: Comparison between RS-232 (top) and TTL voltage levels for logic

6.1.2 Ladybug ROS node

There exists a ROS package for the Ladybug camera system from another open-source project called Autoware [57]. This driver had been updated to support the Ladybug 5+ by ”Robot Perception & Navigation Group” from the University of Delaware.
The images are output as one node with 6 separate ROS topics, one for each of the 6 cameras.

The ROS node relies on the Ladybug Linux API/driver which is supplied from FLIR Machine Vision. The supplied driver did however not work on any of the PCs we tried up until the experiment. Support from FLIR was unable to find the root cause of the problem at the time. The issue have since been resolved, and the ROS package verified to work well. The issue was due to a missing ”Documents” folder in Ubuntu, and the folder was missing because the Ubuntu installation was with Norwegian language.

6.1.3 Using Ladybug images from disk

Due to the problem with recording the Ladybug images using ROS the images must be used by the sensor fusion algorithm based on the timestamp in the image name. The name is on the form:

ladybug_18408811_20181122_085508_ColorProcessed_000007_Cam0_215311_104-0000.jpg

ladybug	Name
18408811	Camera serial number
20181122	Date; year, month, day
085508	Time; hour, minute, second
ColorProcessed	Info
000007	Running number with same timestamp
Cam0	Camera number 0-5
215311	Running number since camera started
104	8 bit second parameter
0000	Possibly a millisecond value
.jpg	File type

The interpretation of the filename has been been done by the author and may not be accurate. It is important to notice that the Time stamp is kept constant in the file names on one entire recording session. FLIR Machine Vision was not willing to provide interpretation of the file name parameters.

The code snippet below shows how the filename in the variable stamp is extracted and converted to a ROS readable format. This should match the output from the Ladybug ROS node and the sensor fusion node should therefore work without alterations on realtime imagedata.

Even though every effort was made to synchronize the camera with GNSS time, the output from this code is 51 frames or about 5,1 seconds before the ROS time on the test-sets used here. On other files the difference in time between ROS and the Ladybug is different. This constant is simply subtracted from the array index when extracting images from disk based on their filename. If this difference is originating from the camera or ROS is not known. The synchronisation is therefore not perfect, but as close as possible without having all the data streaming in realtime or saved together in a rosbag. A better filename handling could possibly help as well.

```
def imageNametoTimestamp(self, stamp):
    day = int(stamp[23:25])
    month = int(stamp[21:23])
    year = int(stamp[17:21])
    hour = int(stamp[26:28])
    minute = int(stamp[28:30])
    second = int(stamp[30:32])
    running = int(stamp[60:66])
    milli = int(stamp[72:75])
    imagetime = datetime(year, month, day, hour, minute,
        second, second*1000)
    if self.firstimage == None:
        self.firstimage = running
        self.firstimagetimestamp = time.mktime(imagetime.timetuple())
    else:
        self.newimagetimestamp = time.mktime(imagetime.timetuple())
    self.imagetimestamp = self.firstimagetimestamp +
        (running+(milli/1000)-self.firstimage)/10.
    if self.newimagetimestamp > self.imagetimestamp:
```

```
20          self.imagetimestamp = self.newimagetimestamp
21          print('Timestamp updated')
```

6.2 Computer hardware

A challenging part initially, was the large number of prerequisites that was needed to make the networks run on a GPU with the required architecture and performance. Initially the tests had been performed using a Lenovo laptop with only an integrated Intel GPU, see table 6.2, which gave very poor performance on the YOLO detector. Each image processing would take roughly one minute. Since the performance of the YOLO detector was substantially below par another laptop was tried, see table 6.3, which had quite good specifications even though it was about 5 years old and in a bad shape physically. This laptop have a Nvidia GPU with support for CUDA which is used in Nvidia GPUs to access the GPU processing power. It is commonly known as a GPGPU (General-Purpose computation on Graphics Processing Units) and is similar to OpenCL, but more optimized for Nvidia cards. A fresh installation of Ubuntu 16.04 LTS (Xenial Xerus) was installed on a partition on the SSD (Solid State Disk) which is by far the fastest of the two available disks on the machine.

As can be seen from table 6.2 and 6.3 there is an 540 fold increase in the speed of the YOLO detector when using the GTX 860M GPU compared to the CPU on the Lenovo Yoga. The Re^3 tracker has a more conservative increase, but the 8 fold increase in speed is still massive. Without CUDA the Re^3 tracker on the Multicom machine ran with 9 frames per second.

The Dell laptop used during the thesis outperforms, on paper, most other portable computers currently available. The performance was initially substantially lower than was expected, providing lower performance than the Multicom laptop. This luckily turned out to be due to a driver issue. The new Dell laptop have roughly twice the practical performance compared to the Multicom one, see table 6.4. Mostly due to the GPU performance. Table 6.1 shows a comparison of processing power between the Nvidia GeForce 860M and the Nvidia Quadro P3200 Mobile. Giga Floating Point operations per Second (GFLOPS) are used as a measure for processing power. The GPU dedicated memory need to be large enough to load the large CNN models simultaneously.

To be able to utilize the complete powers of both the full size YOLOv3 detector and the Re^3 tracker it is needed at least 3 GB of GPU memory (RAM). The 2 GB GPU memory on the Multicom laptop limited the performance to either use the full size YOLOv3 together with the CPU only version of Re^3, or use the GPU version of Re^3 together with the smaller YOLOv3-tiny detector. The latter is shown in Appendix figures A.9 and A.10.

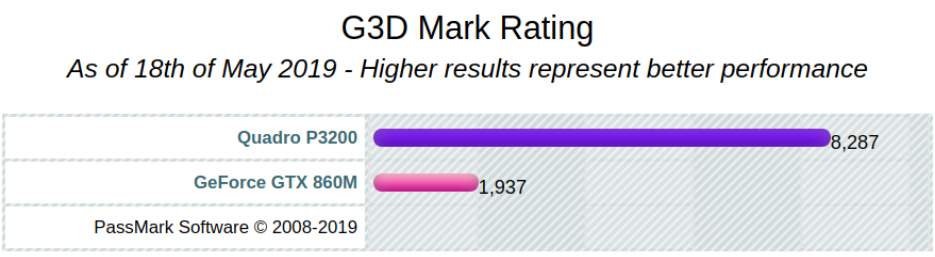


Figure 6.3: GPU 3D Mark performance of the Dell and Multicom laptops. PassMark Sodtware

GeForce 860M	2 GB memory	1389 GFLOPS
Quadro P3200 Mobile	6 GB memory	5300 GFLOPS

Table 6.1: Comparison between the GPU in the Multicom and Dell laptops

Lenovo	**YOGA 520**
CPU:	Intel Core i5 7200U 2,5GHz
GPU:	Intel HD Graphics 620
RAM:	8 GB
Storage:	256 GB SSD
YOLOv3:	1 frame per minute
Re3:	7 frames per second

Table 6.2: Laptop used for initial experiments

Multicom	**Xishan W230S**
CPU:	Intel Core i7 4710MQ 2,5GHz
GPU:	Nvidia GeForce GTX 860M
RAM:	16 GB
Storage:	240 GB SSD + 1 TB HDD
YOLOv3:	9.9 frames per second
YOLOv3-tiny:	78 frames per second
Re3:	64 frames per second

Table 6.3: Laptop used for experiments in specialization project report

DELL	**Precision 7530**
CPU:	Intel Core i9 8950HK 2,9GHz
GPU:	Nvidia Quadro P3200 Mobile
RAM:	32 GB
Storage:	1 TB SSD
YOLOv3:	26 frames per second
YOLOv3-tiny:	128 frames per second
Re3:	115 frames per second

Table 6.4: New DELL laptop used during the Master's Thesis

Combined Performance in the tracking pipeline

DELL	**Precision 7530**
YOLOv3:	16 frames per second
Re3:	38 frames per second

Table 6.5: Combined performance when using both the YOLOv3 detector and the Re^3 tracker together with the RADAR fusion pipeline

6.3 Verification of the YOLOv3 Detector

The YOLO visual object detector comes in many variants with different size, speed and detection capabilities. The full size YOLOv3 and YOLOv3-tiny detectors with standard networks and weights have been tested on several images of boats, as well as the experiment images, to verify the performance. Most of these tests were performed during the author's specialization project report [1]. The network is out of the box trained using the COCO dataset which contains 80 different object classes. The network is therefore not only able to detect and classify boats. As can be seen in figure 6.6 the regular YOLO detector have much higher capabilities compared to the tiny version.

The tests indicate that YOLOv3 is quite accurate for this type of application. However it might detect boats as other similar looking objects when the distance to the vessel is large. Similar in this context means "surfboard" and "frisbee", due to the distinct visual appearance of the target boat "Ocean Space Drone 1", resembling a white cylinder- or disk-like object. Erroneous detections in other parts of the image is rare except when the target boat is far away and the background is cluttered by Trondheim city skyline. These wrong detections have a low certainty for the object type from YOLOv3.

Using the standard YOLOv3 net and weights the detector successfully detected boats and ships in a large range of scales with different appearance. It was also able to find multiple boats in the same image. To detect boats as far away as possible the raw image was cropped into tiles 1/6 of the original width. The cropped images was taken from the raw image around the horizon and fed through the YOLOv3 network. On the cropped images the YOLOv3 algorithm manage to detect boats that are about 500 meters away and only occupy 5×10 pixels. That is 0.2% and 0.5% of the height and width of the raw image. See figure 6.5.

Without cropping the image the detector is unsuccessful in detecting smaller boats far away, as can be seen in figure 6.4. The reason for the lack of detection is due to downsizing of the raw image within the detector. Every image is converted to a fixed size before the convolutional network starts the processing [3].

The combination of detection process in cropped images along the horizon towards detected vessels as well as the original uncropped images would provide the best detection and classification performance. This method would however be time- and processing intensive and might not be able to have sufficient real-time performance.

Using the smaller YOLOv3-tiny network the detector only detected the biggest boats and was unsuccessful in finding the more exotic boat shapes. The tiny version of the detector is significantly faster, see table 6.4 and 6.3, but the lower robustness limits the practical use. The full size YOLOv3 is anyway capable of running in realtime on a powerful GPU as can be seen in table 6.4.

By training the networks to only search for boats it is possible that the tiny version would be significantly better. It would still be much less capable of finding small boats though, compared to the full size YOLOv3.

In Appendix figures A.4, A.5 and A.6 the performance of the full size YOLOv3 and YOLOv3-tiny are evaluated on the same video as the Re^3 tracker. It is clear that the robustness of the tiny detector is severely lower than the full size detector. The full size detector finds the boats reliably and also correctly detects birds and people. The tiny version incorrectly detects for instance rocks as elephants and cows, and is much less robust in detecting the boats overall.

Figure 6.4: Examples of the performance of the YOLOv3 detector on the raw Ladybug images. The target vessel ”Ocean Space Drone 1” is too far away and too small to be detected

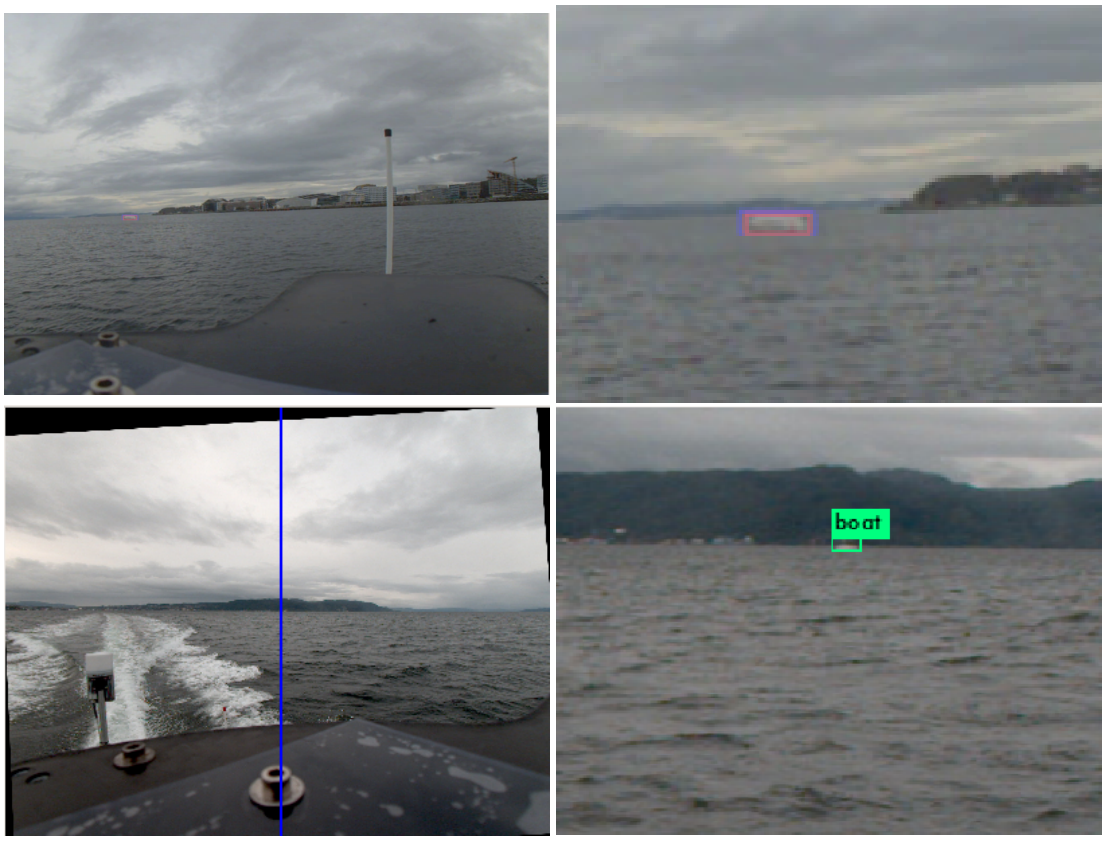
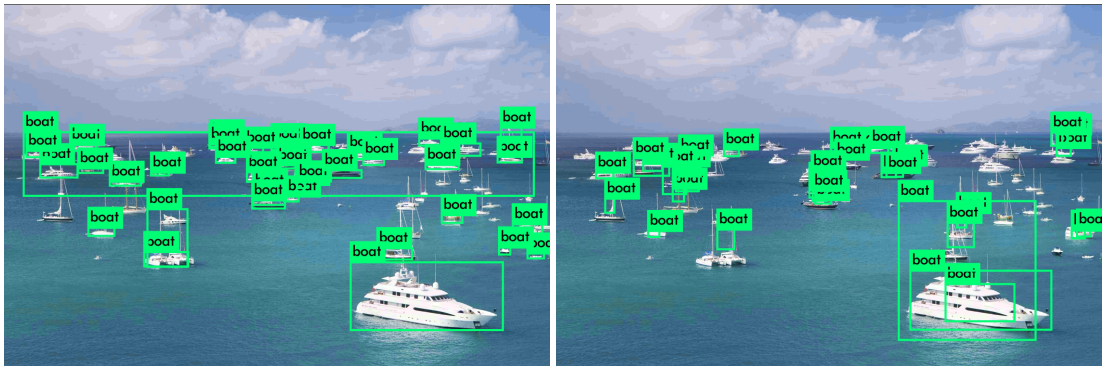
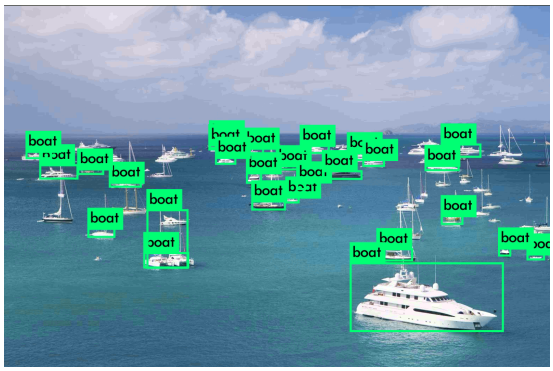


Figure 6.5: By tiling the original raw Ladybug image in such a way that we have several images along the horizon, the YOLOv3 detector is able to detect the target vessel even at a large distance.

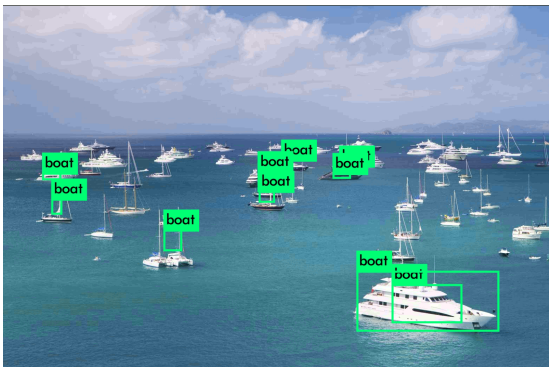


(a) YOLOv3 Accepted certainty: 0.15

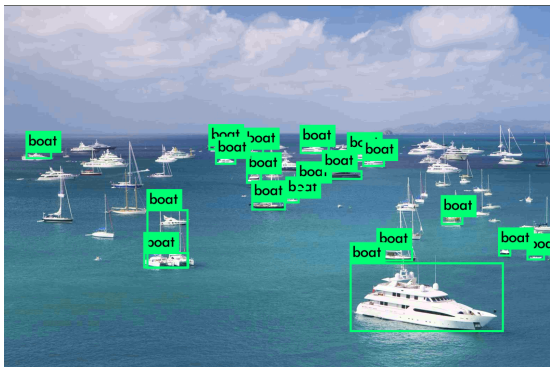
(b) YOLOv3-tiny Accepted certainty: 0.1



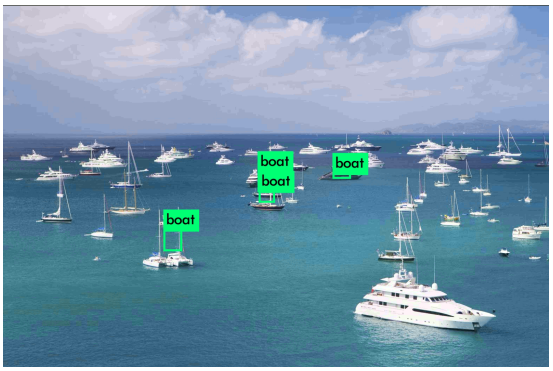
(c) YOLOv3 Accepted certainty: 0.2



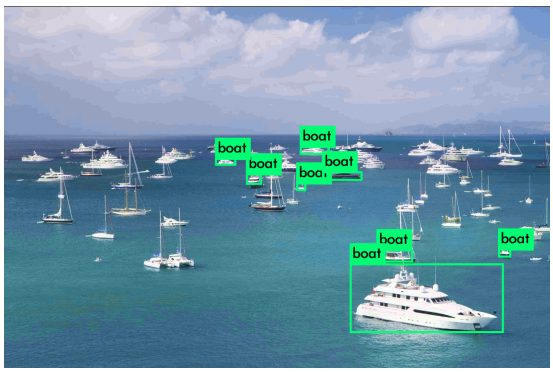
(d) YOLOv3-tiny Accepted certainty: 0.2



(e) YOLOv3 Accepted certainty: 0.3



(f) YOLOv3-tiny Accepted certainty: 0.3



(g) YOLOv3 Accepted certainty: 0.5

(h) YOLOv3-tiny Accepted certainty: 0.5

Figure 6.6: Performance of regular YOLOv3 (left) and YOLOv3-tiny (right)

6.4 Verification of the Re^3 Tracker

Using several different videos of boats at sea found on Youtube as well as the images captured during the experiment, the performance of the Re^3 tracker have been tested. These test were also to a large degree done throughout the author's specialization project report [1]. In these and other tests the bounding boxes were manually drawn around the boat to initialize the tracker. After initialization the tracker was left unaided.

The Re^3 tracker has demonstrated impressive performance in tracking boats with large changes in visibility and appearance. The bounding box is however adapting to the visible part of the boat, and may not expand sufficiently out again after occlusion, or it may expand too much if there is not enough contrast between the background and the vessel. See the images in Appendix figure A.8 and A.7 for examples of the unaided tracking performance.

The new image is compared to the previous image padded to a size twice that of the bounding box. These cropped images are furthermore warped to a fixed size of 227x227 pixels within the Re^3 tracking algorithm to match the underlying CNN structure [4]. Both the padding and the network size is hard-coded and difficult to change. The LSTM track length can however be altered, and a length of 64 frames (compared to 32 originally) before resetting the LSTM state seem to be optimal for the tracking performance.

Should motion of the ASV affect the image such that the bounding box shifts more than the size of the tracked object between frames, tracking is lost due to the object being outside the cropped image tested by the network. Changes of more than half the bounding box size negatively effects the tracking performance. Small and distant objects are therefore much more exposed to track loss caused by vessel motion and poor stabilization.

Notice that the tracker does not terminate on its own if the tracked object translates out of the image or if track is lost. Depending on the scene, the tracked bounding box might shrink or grow and might drift away. Handling of such occurrences must be done outside the tracker.

In Appendix figure A.7 "Telemetron" does rather aggressive maneuvers without the tracker loosing the target. The target is quite far away as it is not very visible in the image. Still the tracker performs well. The bounding box robustly locks on to the "Ocean Space Drone 1" target when it is as little as 10×30 pixels, comparable to 200m away. This is further improved when cropping the raw image to a smaller size around the object. Beyond 300m the tracker anyway struggles to track the barely visible boat as motion of the ASV and imperfect stabilization moves the image to much between frames.

6.5 Combination of YOLO Detector and Re^3 Tracker

The combination of YOLO and Re^3 shows promising results, but also limitations. Due to limited GPU memory in the initial tests on the Multicom computer, see table 6.3, it was only possible to run the tiny version, YOLOv3-tiny, together with the Re^3 tracker. Even though YOLOv3-tiny shows poor performance in Appendix figure A.6 it provides updated bounding boxes that are robust enough for the tracker.

In the later tests described below, using the Dell laptop; see table 6.4 and 6.5, the full size YOLOv3 detector have been used together with Re^3.

The main takeaway from the combination tests are that the tracked object is reliably tracked if it is sufficiently visible and not too much affected by motion or poor image stabilization. The tracker anyway must be monitored and verified by the detector to terminate lost tracks or update the bounding box should

it not match the target adequately. Tuning of parameters for survival of tracks and termination criterias must be performed for the applicable problem. See section 5.6.

6.5.1 Single Object Tracking in Re^3

Single object tracking is quite straight forward with the setup where the YOLO detector initializes and updates the Re^3 tracker. Problems arise, however, if the detector find more than one boat in the image, as the output from YOLO is sorted by object confidence. This will, if not handled, cause the bounding box to be reinitialized if a new and more confident detection is made within the image. This is due to the use of the Jaccard index, IoU, which output a zero value when no overlap between tracker and detector. A value between 0 and 1 indicates overlap and likely the same object for both modules.

This naive handling would be useless for aiding another sensor as it would potentially create much noise and erroneous detections. Handling of tracks must therefore be more stringently enforced. This is most easily done by narrow regions where proposals are only accepted, or better handling of multiple objects. The tests described in section 6.6 does not utilize multi-object tracking in each image, but enforces strict requirements for the location and accuracy of the bounding box.

6.5.2 Multi Object Tracking in Re^3

The multi-object tracking problem is non-trivial and several methods are available depending on the problem at hand. In multi-object tracking using the YOLO detector and the Re^3 tracker, the number of initialized tracks grows continuously if new detections are not matched with existing tracks.

The Kuhn-Munkres algorithm [47] is used here to match the detections with existing tracks. See figure 4.5. If more detections than tracks are found, a new track is initialized for the new detection bounding box. If the IoU value between the bounding box from detections and the bounding boxes from existing tracks are below a given value the tracker is updated with the new detection. Otherwise the tracker is left to continue tracking. The value is currently set to 0.5. If the IoU value is zero it indicates that the detection is new, or that the tracker has lost the desired object and the track will be terminated. Termination of tracks are currently implemented based on time passed and location of the bounding box within the cropped image. The time limit must be set quite low as only the detector have any reliable evaluation of object type and location.

6.6 Active-Passive Sensor Fusion; RADAR and camera

Due to the single target nature of the experiment data, no multi-target method is utilized on the Re^3 tracker in these tests. The targets are far away and only one vessel is anyway visible within the cropped image. It is however implemented tracking of multiple RADAR tracks using one separate Re^3 tracker per RADAR initialized track. A maximum of two GPU-powered Re^3 trackers can be used simultaneously on the Dell laptop together with YOLOv3, table 6.4.

In the complete tests using the Re^3 tracker together with the YOLOv3 detector and the sensor fusion algorithm, the overall performance unfortunately drops slightly compared to the standalone tests. This is due to several factors, most prominently the imperfect stabilization of the cameras, but also a visible performance drop when YOLO uses the GPU resources together with Re^3.

To overcome these limitations, and for a visual tracker to really be useful compared to only using an object detector, a better stabilization must be performed. It is likely that most of the issue arises from

Figure 6.7: If the chosen IoU value for updating the tracker is chosen too low one might end up with overlapping bounding boxes on the same object. The tiny detector had trouble locking onto the whole boat.

Figure 6.8: Examples of the multi-target performance of the combined YOLOv3-tiny detector and the Re^3 tracker. The pruning of old tracks is not optimized.

poor time synchronization, as the Kongsberg Seapath 330+ is very accurate. Table 2.1. Another method that will aid performance is to run each deep learning CNN based module on its own GPU. As it is now only one process can utilize the GPU processing simultaneously. The modules, even though they run separate threads and in parallel on the CPU, are running in series on the GPU.

The tracks from by the Autosea RADAR tracker is used to initialize the author's EKF-based tracker. The following detections output from the RADAR tracker is used together with the covariance information from the RADAR tracker as measurements for the EKF. These measurements are used together with the INS attitude and position to extract cropped images containing the tracked vessel. These cropped images are further processed in the YOLOv3 detector and the Re^3 tracker. The full process is described in chapter 4 and 5.

6.7 Fusion pipeline test results

Results from tests on some of the experiment data is provided below. The plots show the estimated position and covariance ellipses from the EKF-based tracker together with received data for RADAR and AIS. The AIS is not used in any of the calculations and only shown as "Ground Truth" for evaluating the performance. The RADAR data received from the Autosea RADAR tracker includes position and covariance information which is plotted as ellipses. The plotted position for detections from the YOLOv3 detector and the Re^3 tracker is not directly used in the computations as only the angle is input to the EKF. The positions plotted are based on the detected angle together with the EKF estimated range/distance.

Different tests are performed evaluating tracking performance compared to the AIS track as Ground Truth. Every evaluation test is impossible to present here, but a representative selection is provided in the following sections. Tests have been done with fusion of the following data:

- RADAR only
- RADAR and YOLOv3 detections
- RADAR, YOLOv3 and combined Re^3 tracks
- RADAR and Re^3 tracks initialized by RADAR detections

No changes has been made to the parameters in the EKF between the presented tests with different sensors. The algorithm runs however faster when some modules are not used and this might affect the results slightly. The detections for YOLO changes between runs and the same is to a certain degree true for the Re^3 tracker. Some variation will therefore be present in the data.

Figure 6.9: Target tracked using Re^3 in left image. In the image on the right the vertical lines indicate estimated position by the EKF (blue), RADAR detection (red), and camera detection (green). Slight delay might be visible. Test 1

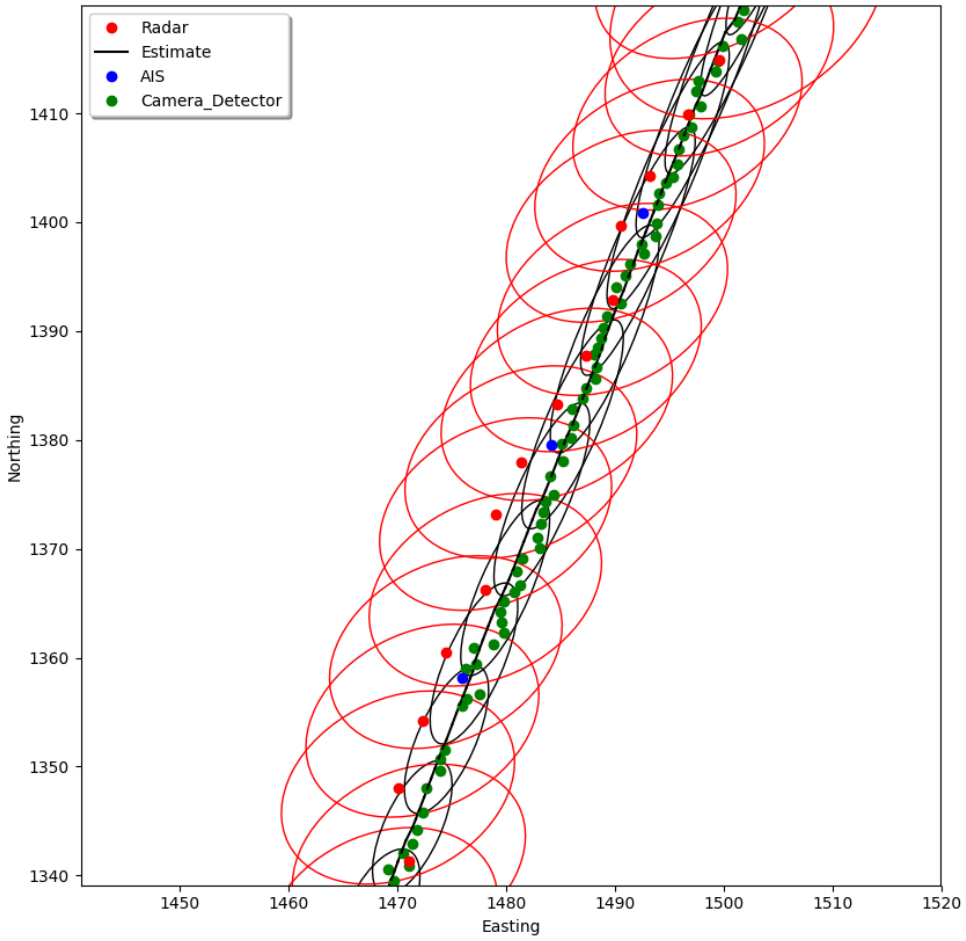


Figure 6.10: Example of improved tracking performance and reduced uncertainty by fusion of RADAR and YOLOv3 detector.

6.7.1 Test 1, approaching tracked vessel

In this example the ASV is following the target at higher speed. There are only small variations in angle between the vessels. All detections of the target are in camera 0 (forward looking).

Average maximum offtrack error

RADAR only:	4 meters
RADAR + YOLOv3:	2 meters
RADAR + YOLOv3 + Re3:	2.5 meters
RADAR + Re3:	2.5 meters

Best maximum offtrack error

RADAR only:	4 meters
RADAR + YOLOv3:	1.5 meters
RADAR + YOLOv3 + Re3:	1.8 meters
RADAR + Re3:	1.8 meters

Worst maximum offtrack error

RADAR only:	4 meters
RADAR + YOLOv3:	3 meters
RADAR + YOLOv3 + Re3:	5 meters
RADAR + Re3:	10 meters

Table 6.6: Maximum offtrack error comparison between the methods described in section 6.7.1 and on the dataset used. The values are averaged over several runs of the algorithm. Note that the larger error involving Re^3 is due to track initialization errors.

As can be seen in table 6.6 both the YOLOv3 detector and the Re^3 tracker help reducing the offtrack error compared to ground truth given by the AIS data. The ground truth is a straight line between the first and last AIS detections. The plots therefore show larger error in the first and last part of the track, outside the line. As can be seen from the results, Re^3 must however be monitored to avoid introducing erroneous tracks.

The lack of perfect image stabilization and timing is not very visible in this test-set. It can anyway be noticed especially in the beginning and end of the recording.

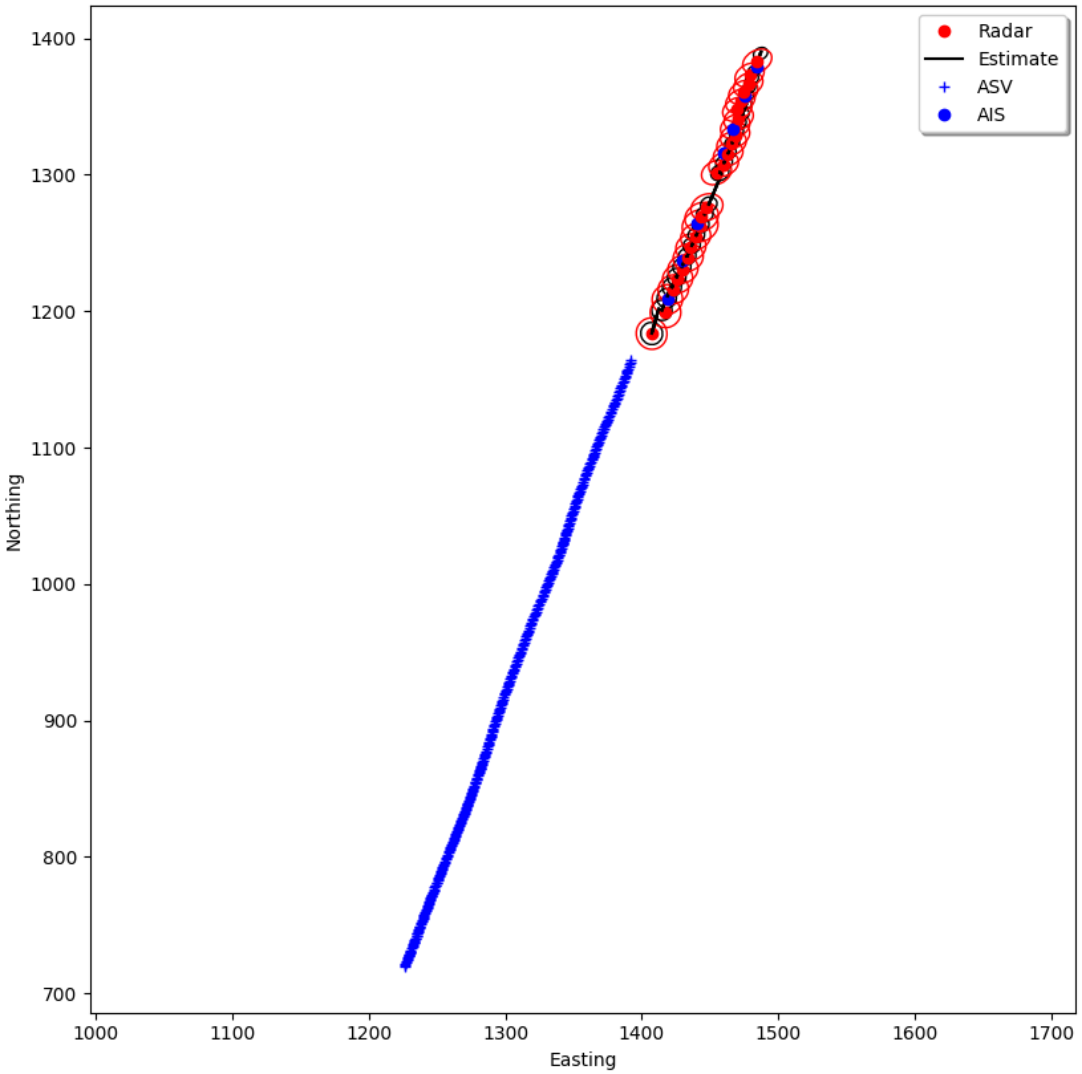


Figure 6.11: The ASV course over ground together with the tracked target, test 1. Track of own vessel in blue, both vessels going north-east.

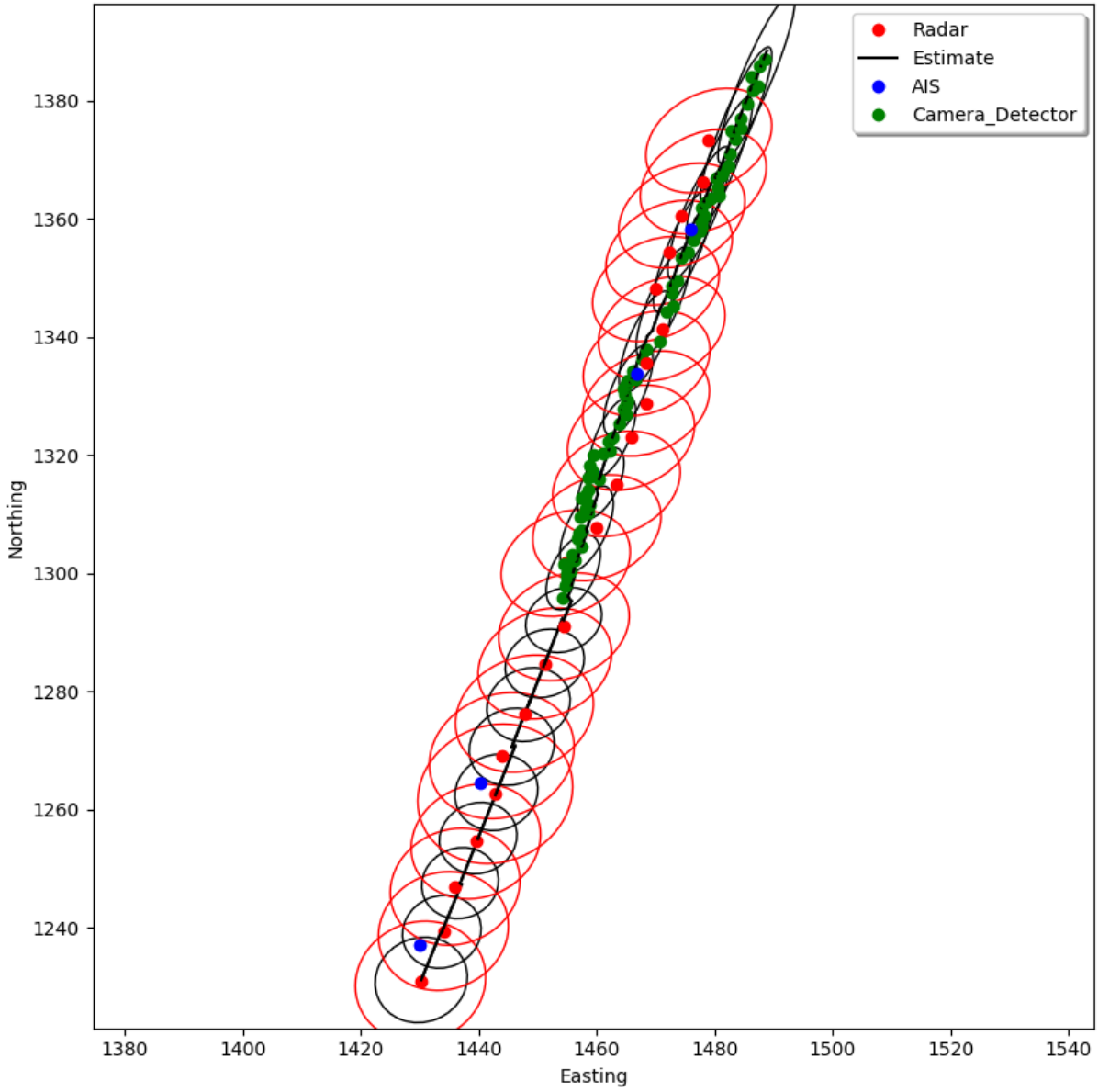


Figure 6.12: Example of improved tracking performance and reduced uncertainty by fusion of RADAR and YOLOv3 detector. Test 1, the YOLO detector is initialized some time after tracking is started.

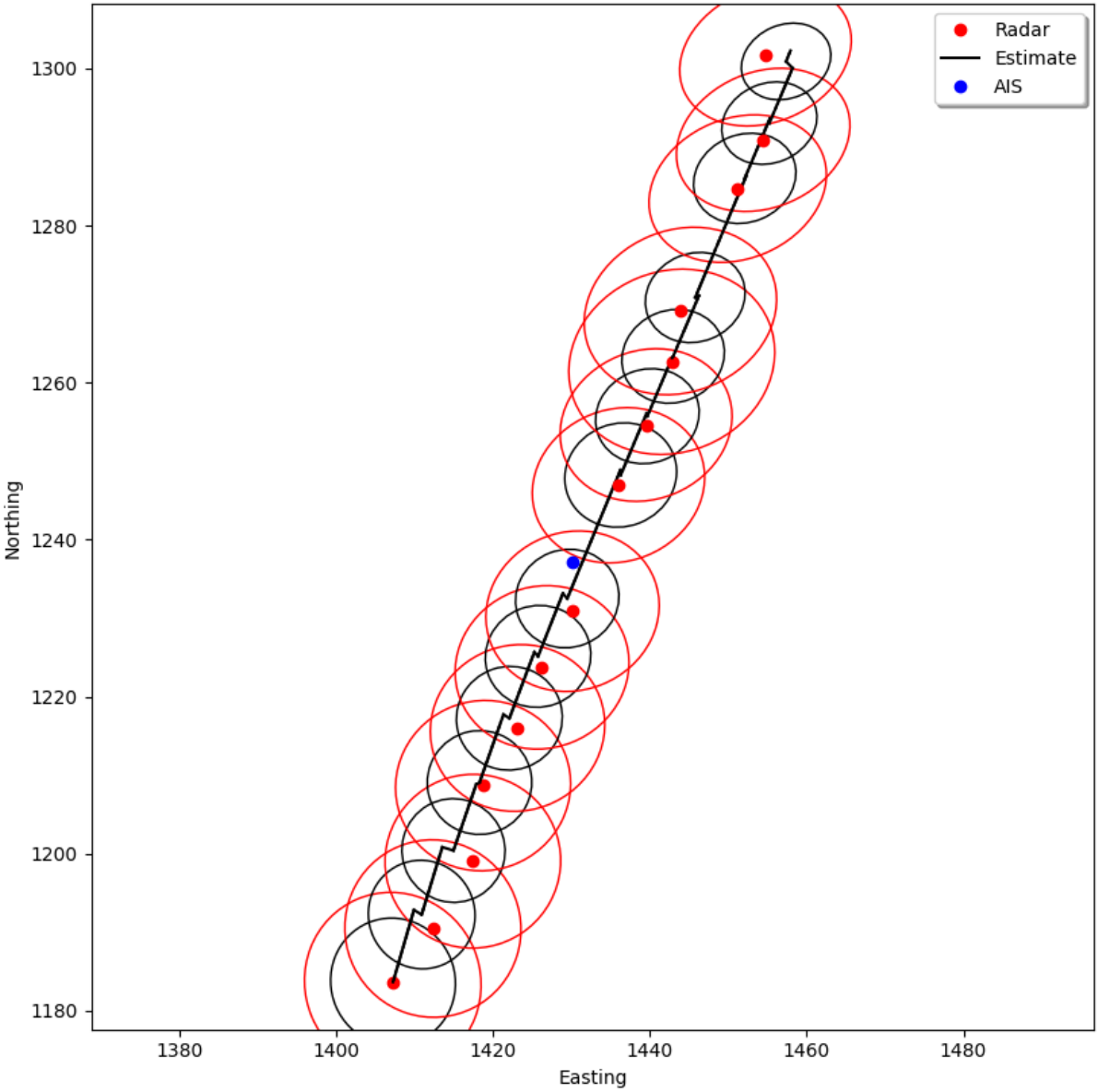


Figure 6.13: Target tracked using RADAR only, test 1

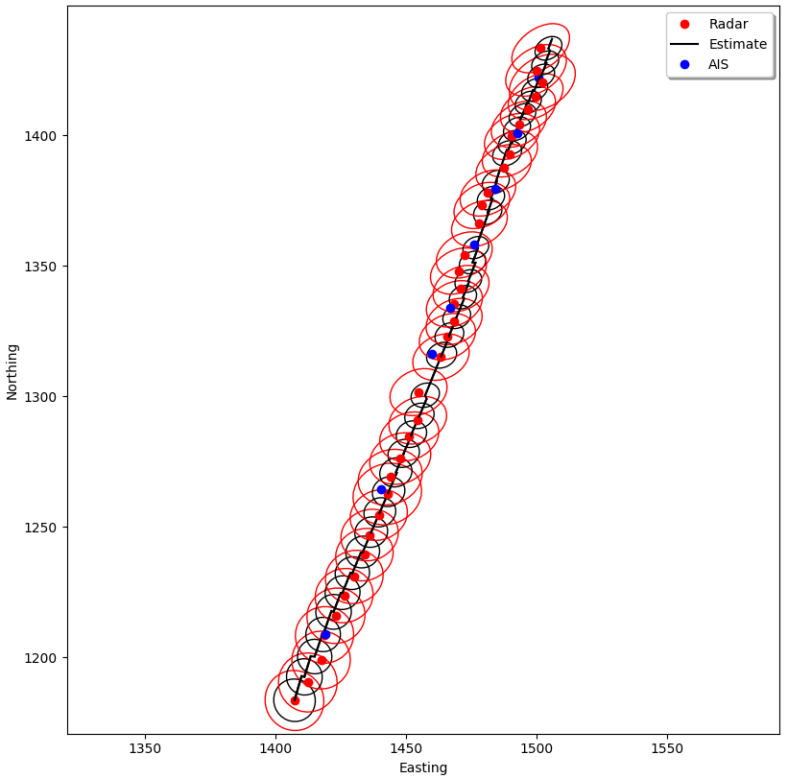


Figure 6.14: Target tracked using RADAR only, test 1

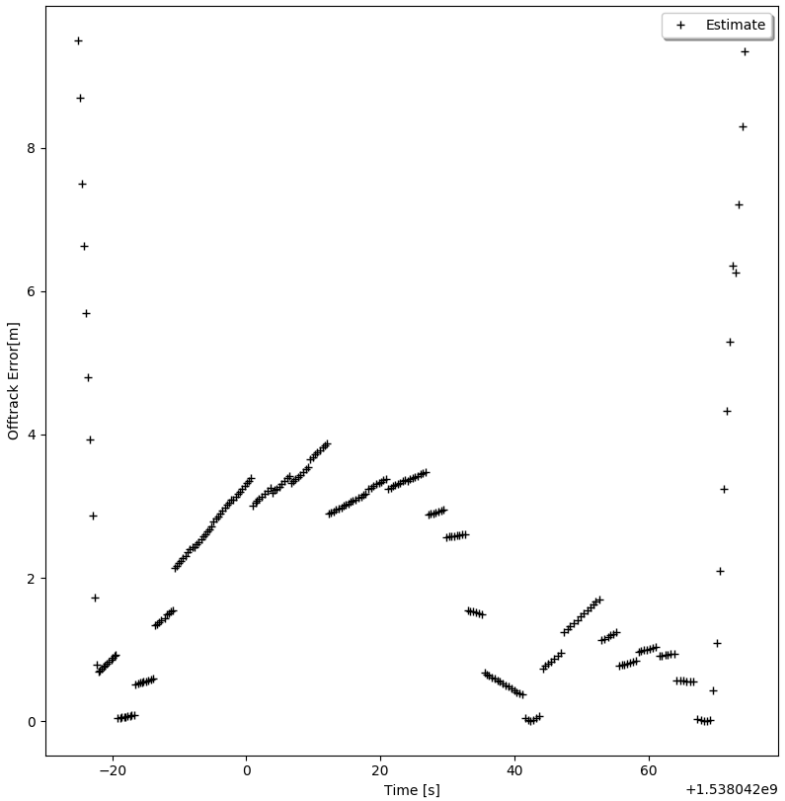


Figure 6.15: Track offset error using RADAR only, test 1

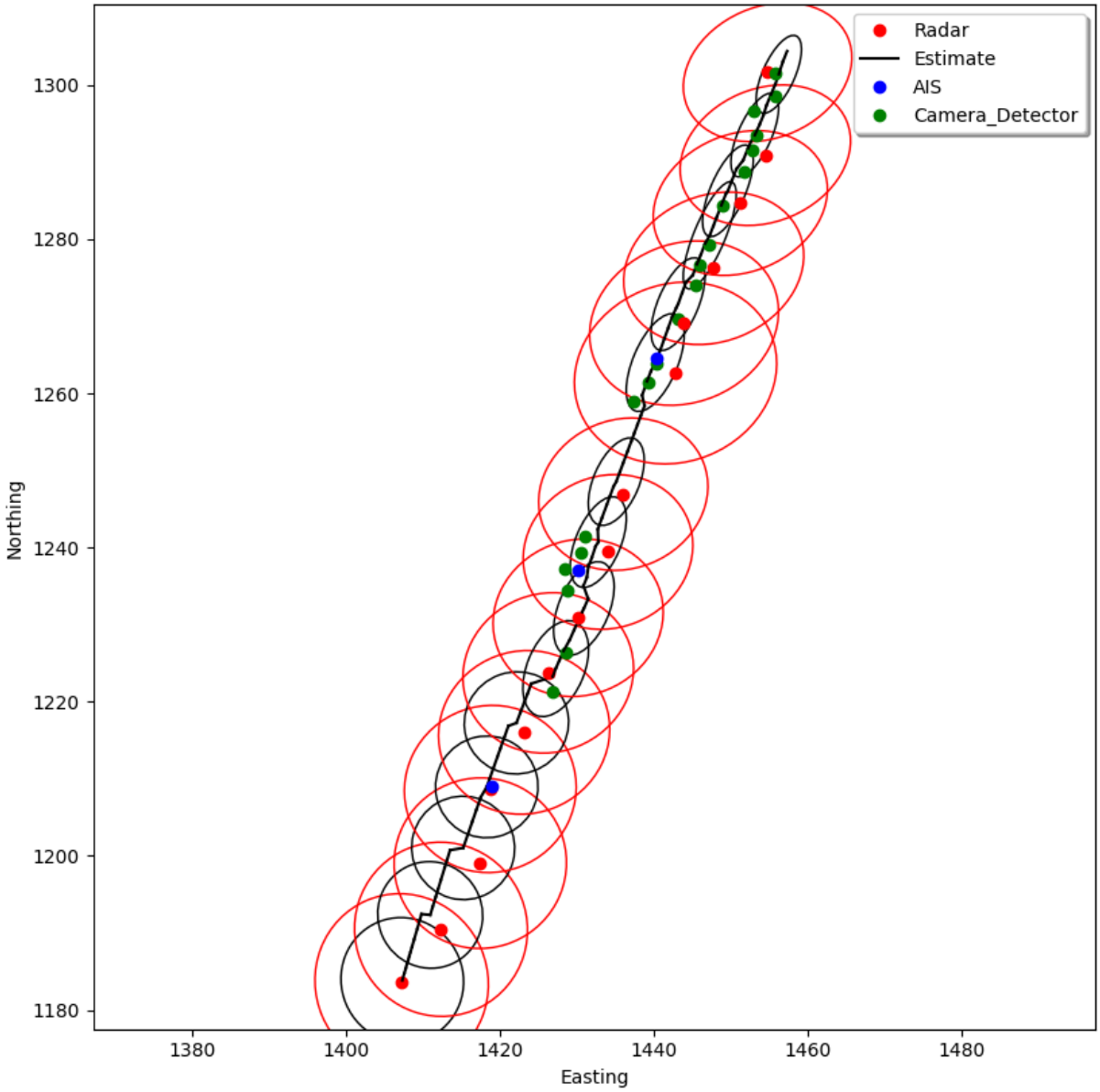


Figure 6.16: Target tracked using RADAR and YOLOv3, test 1

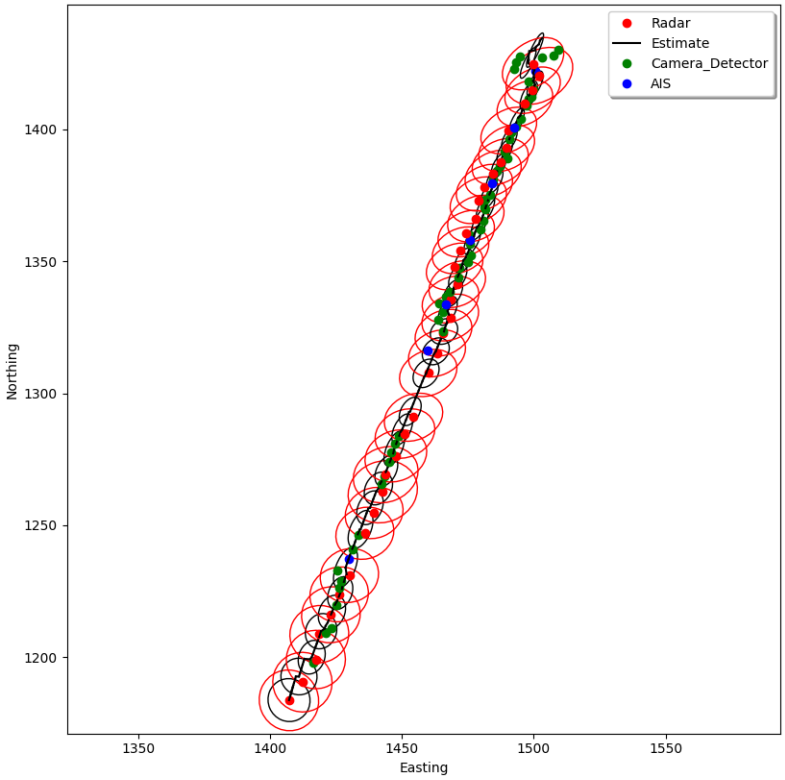


Figure 6.17: Target tracked using RADAR and YOLOv3, test 1

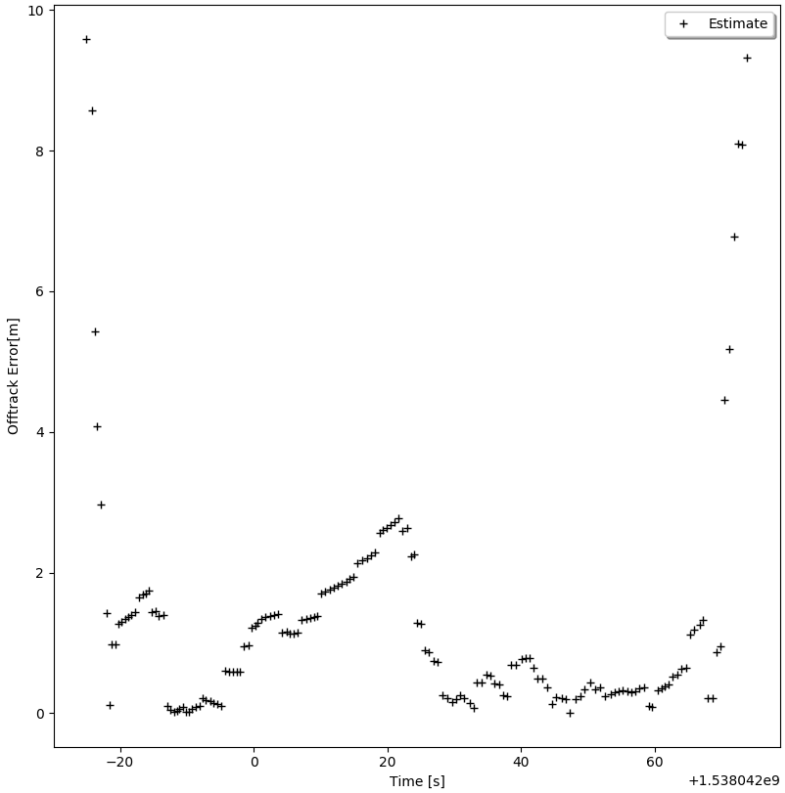


Figure 6.18: Track offset error using RADAR and YOLOv3, test 1

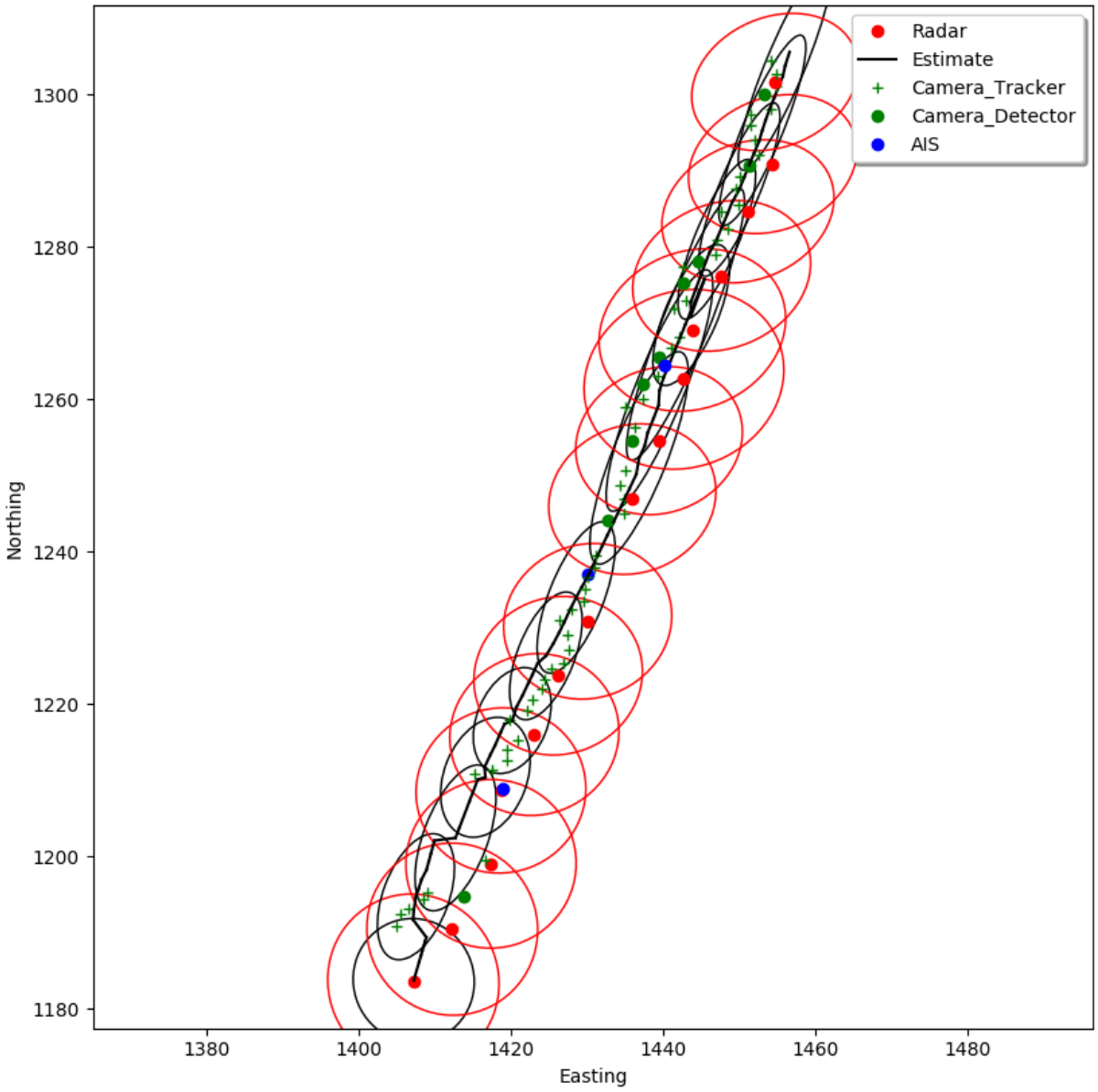


Figure 6.19: Target tracked using RADAR, YOLOv3 and Re^3, test 1

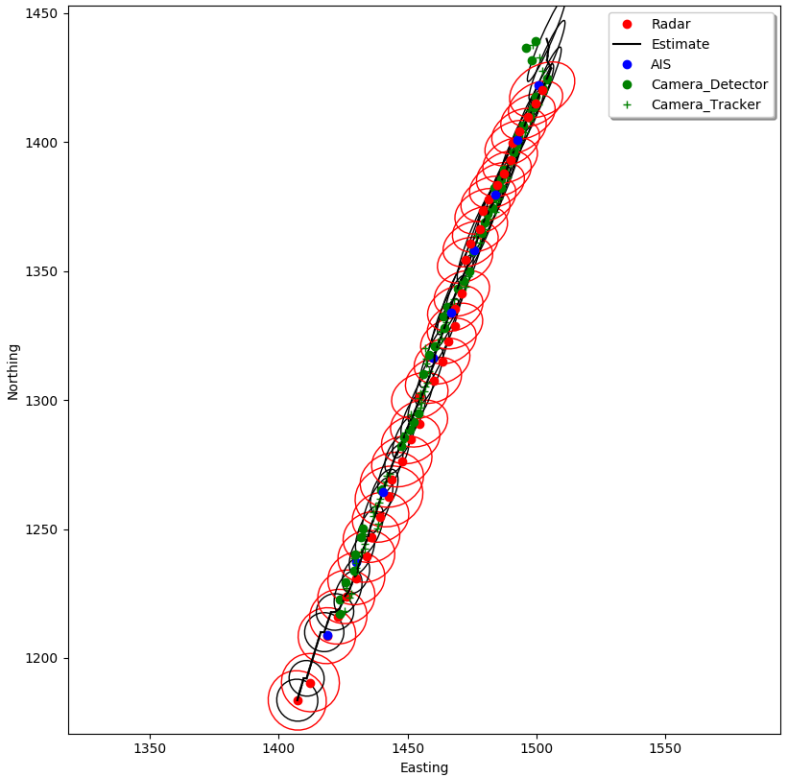


Figure 6.20: Target tracked using RADAR, YOLOv3 and Re^3, test 1

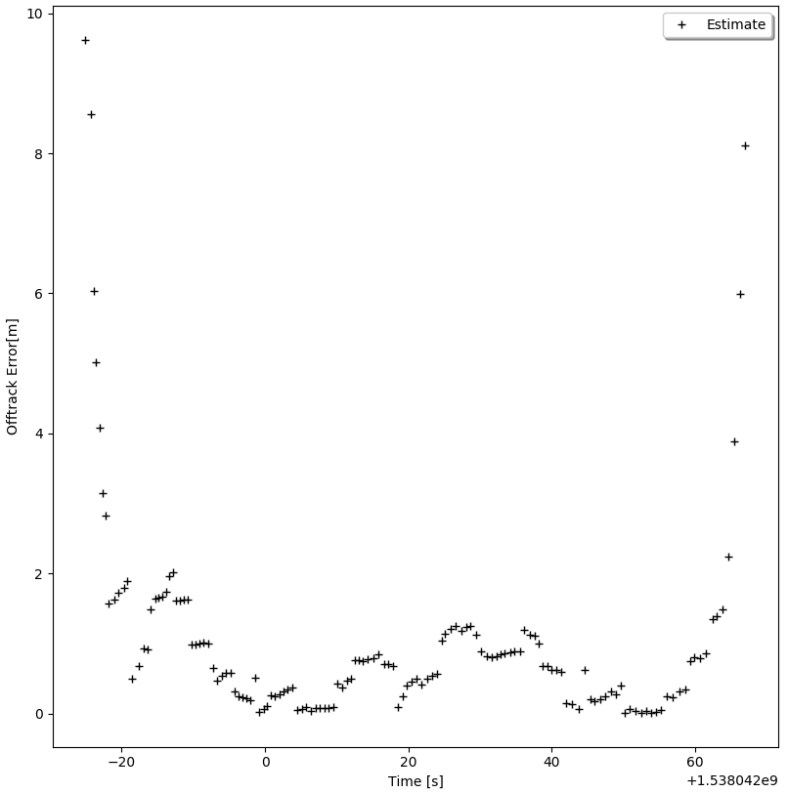


Figure 6.21: Track offset error using RADAR, YOLOv3 and Re^3, test 1

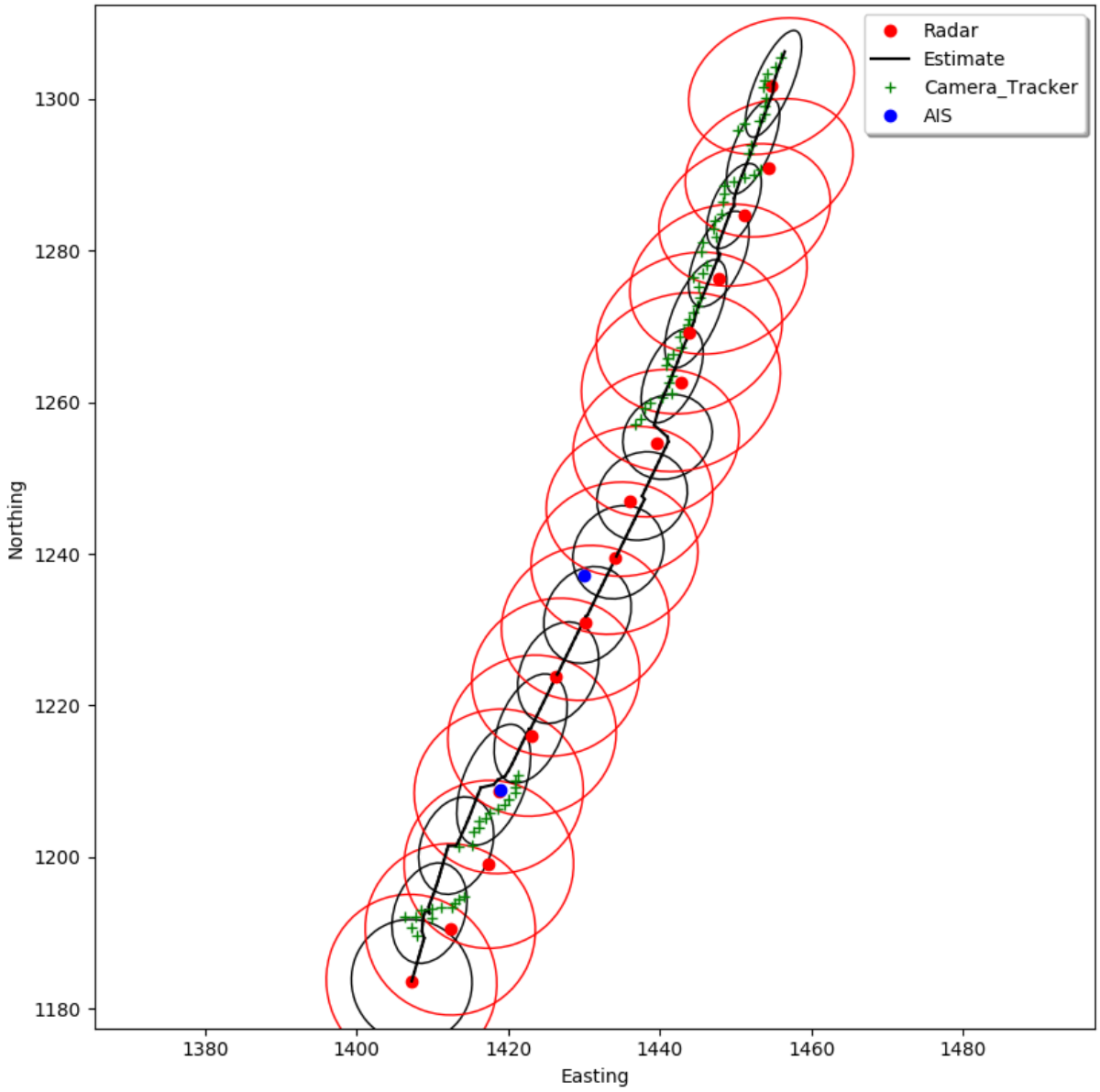


Figure 6.22: Target tracked using RADAR and Re^3 without detector, test 1

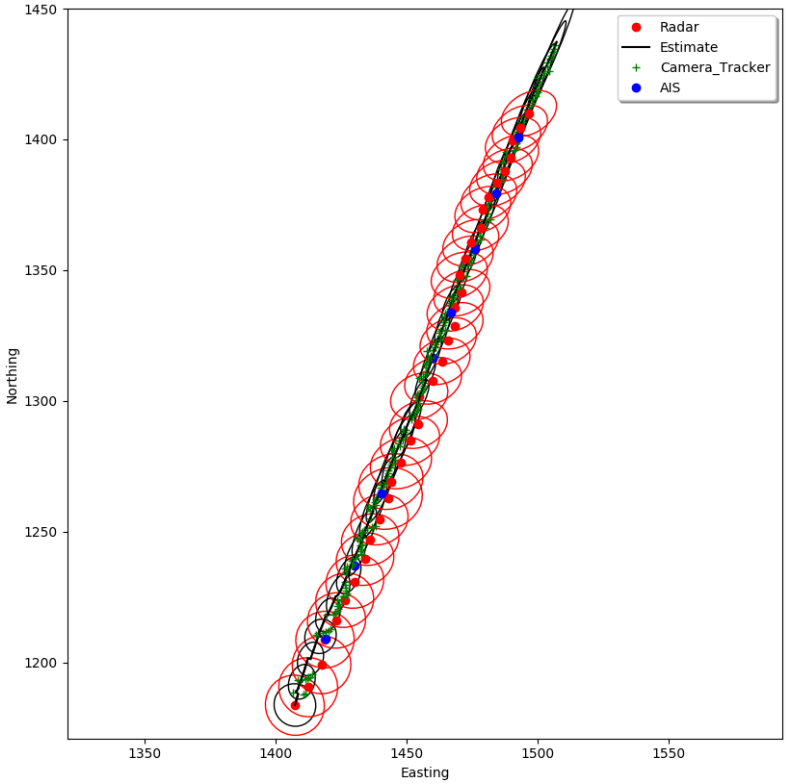


Figure 6.23: Target tracked using RADAR and Re^3 without detector, test 1

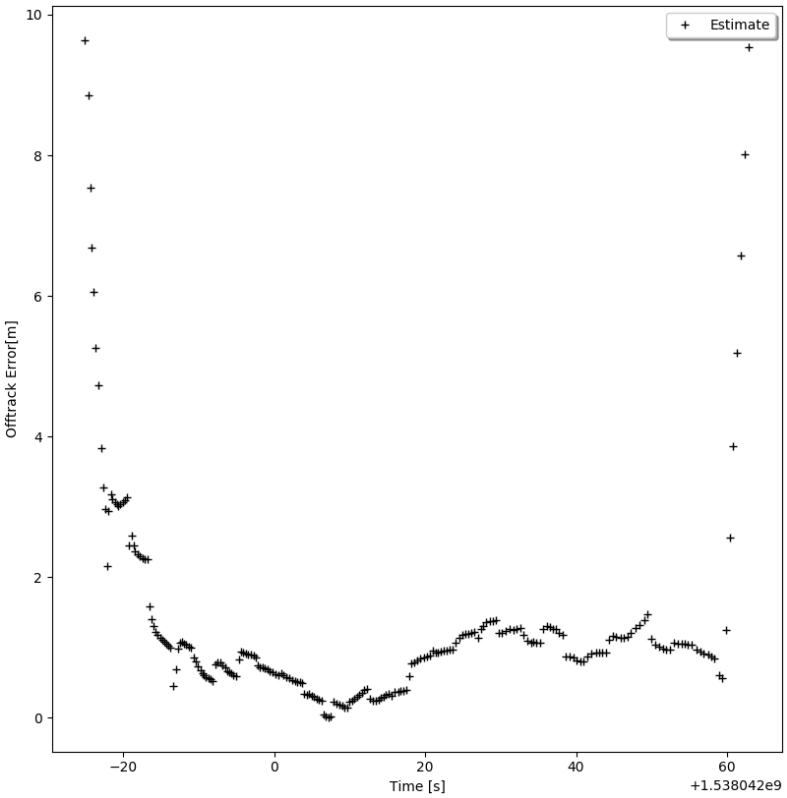


Figure 6.24: Track offset error using RADAR and Re^3 without detector, test 1

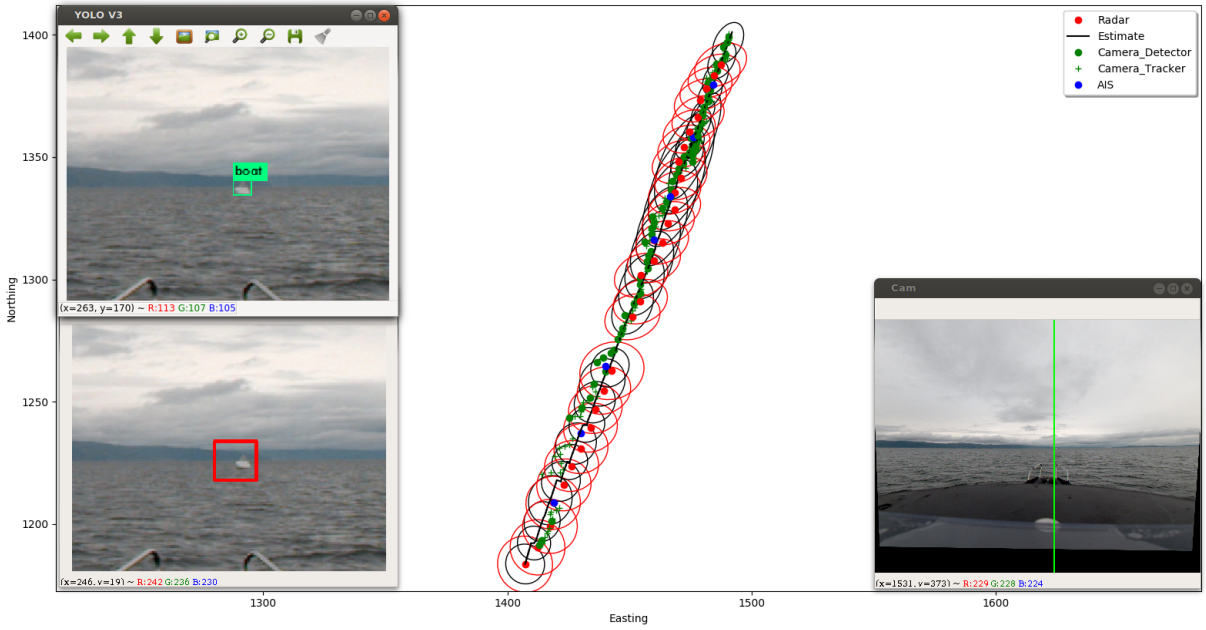


Figure 6.25: Target tracked using all modules with quite low confidence in Re^3, $\sigma_c = 0.1$, test 1. An even lower confidence in Re^3 creates a growing and unrealistically high uncertainty.

6.7.2 Test 2, overtaking tracked vessel

This example is a continuation of the previous where the ASV have caught up with the target drone and is now overtaking it on starboard side. Large variations in both speed and heading is observed in the ASV. The detections translate from camera 0, into camera 4 (port side) and further to camera 3.

The images in Appendix figures A.1, A.2 and A.3 show the attitude of the ASV during this test-set. The tracked vessel is lost behind parts of own vessel during certain maneuvers.

Average maximum offtrack error

RADAR only:	15 meters
RADAR + YOLOv3:	14 meters
RADAR + YOLOv3 + Re3:	13 meters
RADAR + Re3:	15 meters

Best maximum offtrack error

RADAR only:	15 meters
RADAR + YOLOv3:	9 meters
RADAR + YOLOv3 + Re3:	12 meters
RADAR + Re3:	13 meters

Worst maximum offtrack error

RADAR only:	15 meters
RADAR + YOLOv3:	27 meters
RADAR + YOLOv3 + Re3:	30 meters
RADAR + Re3:	21 meters

Table 6.7: Maximum offtrack error comparison between the methods described in section 6.7.1 and on the dataset used. The values are averaged over several runs of the algorithm. On this dataset the largest errors are due to angle errors on the YOLO detections, due to imperfect synchronization.

As can be seen in table 6.7 the visual methods provide limited improvement over the RADAR data for tracking the target under these conditions. The dataset is challenging, with large variations in heading, roll, pitch and speed of the ASV. It is clear that the lack of ideal synchronization between the INS and Ladybug camera impacts the data quality of the detections. From the RADAR data one can also see that the Autosea RADAR tracker introduces artifacts in the track. The target is traveling in a straight line and at constant speed as the AIS data indicates.

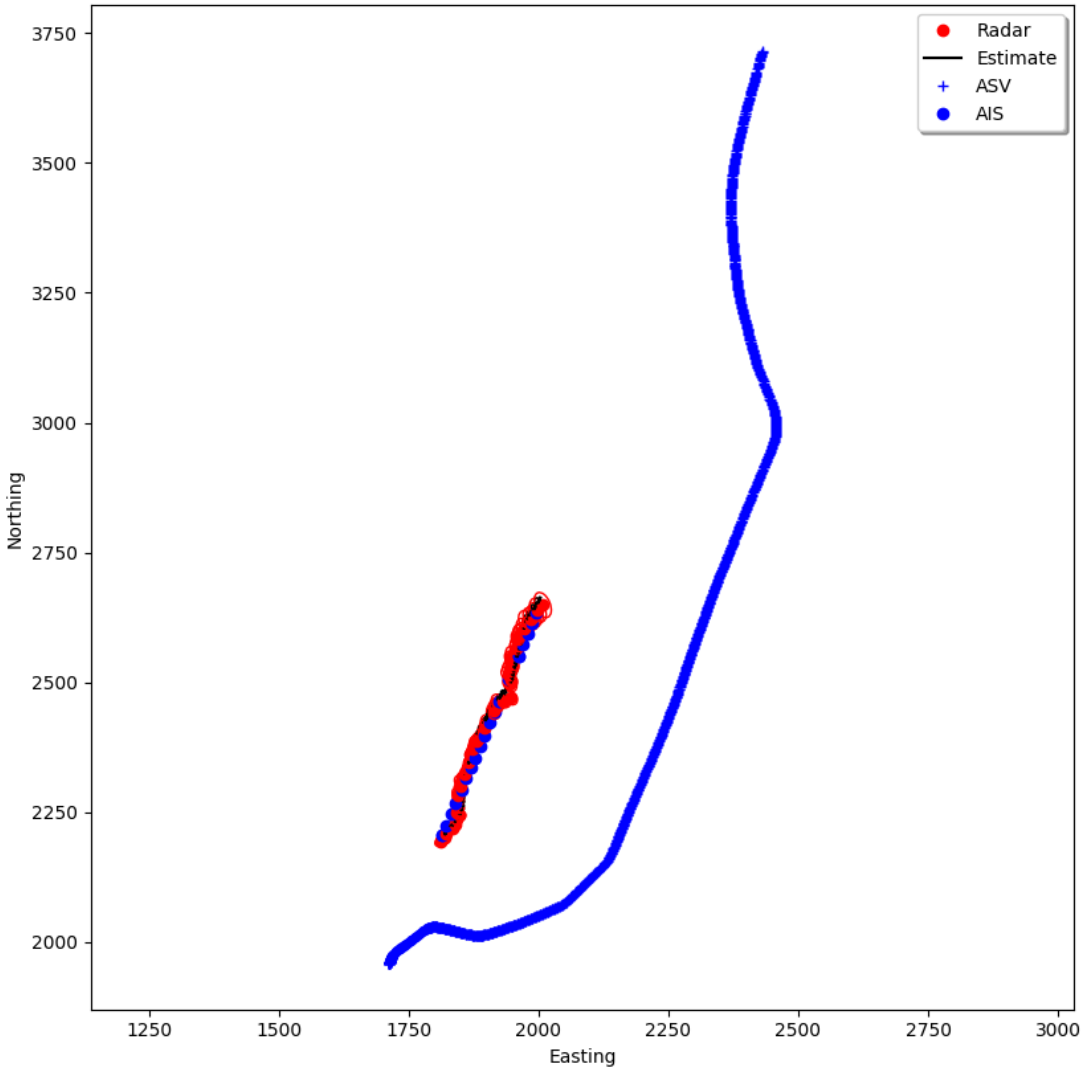


Figure 6.26: The ASV course over ground together with the tracked target, test 2. Track of own vessel in blue, going north-east. Notice the sharp maneuvers which negatively affect the detections. The ASV also have big changes in speed during the first turns.

Figure 6.27: Target tracked using Re^3, test 2

Figure 6.28: Tracking initialization of Re^3 by RADAR. Missed detection shown, test 2

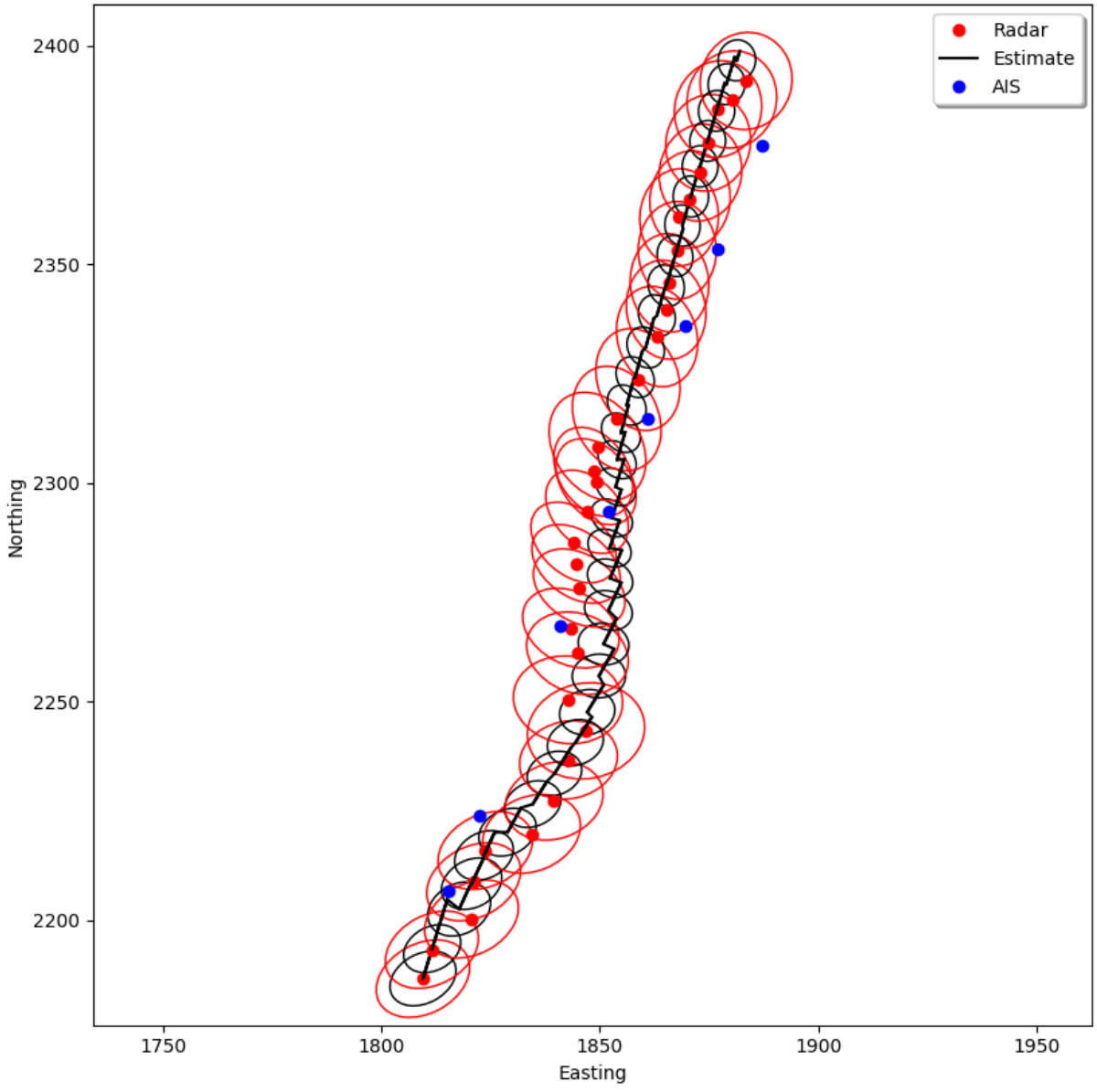


Figure 6.29: Target tracked using RADAR only, test 2

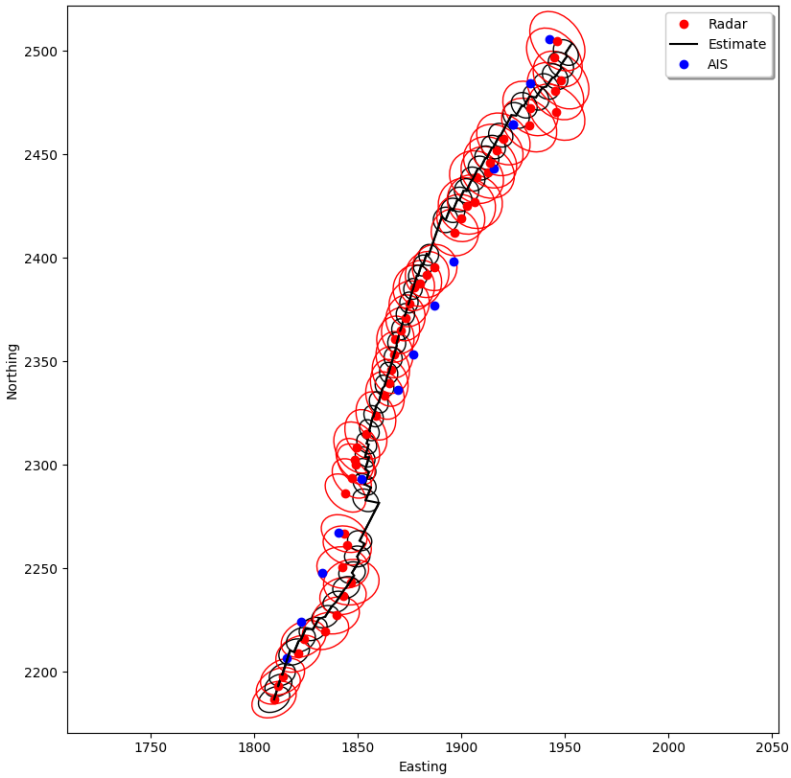


Figure 6.30: Target tracked using RADAR only, test 2

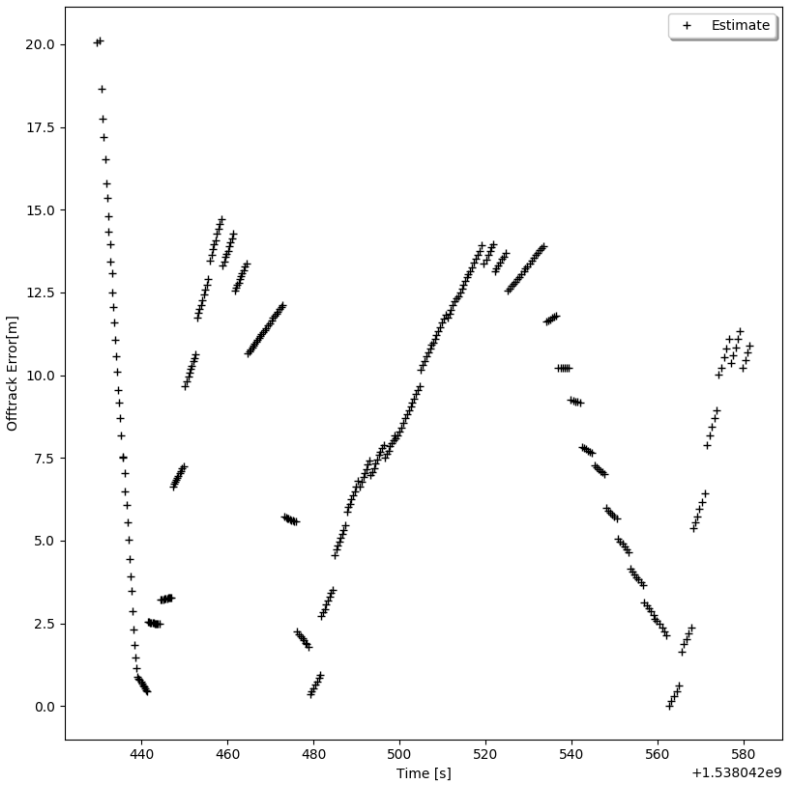


Figure 6.31: Track offset error using RADAR only, test 2

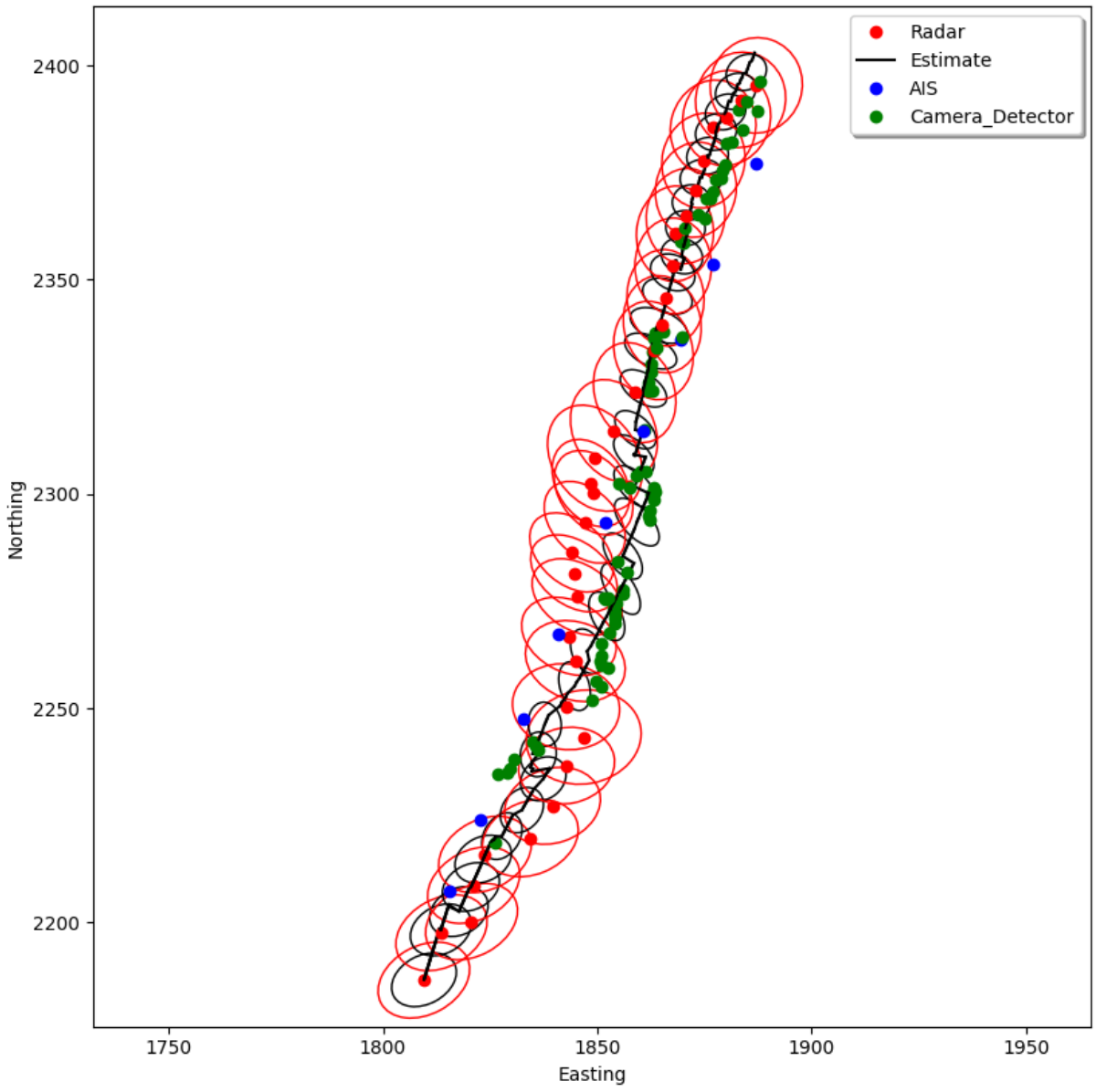


Figure 6.32: Target tracked using RADAR and YOLOv3, test 2

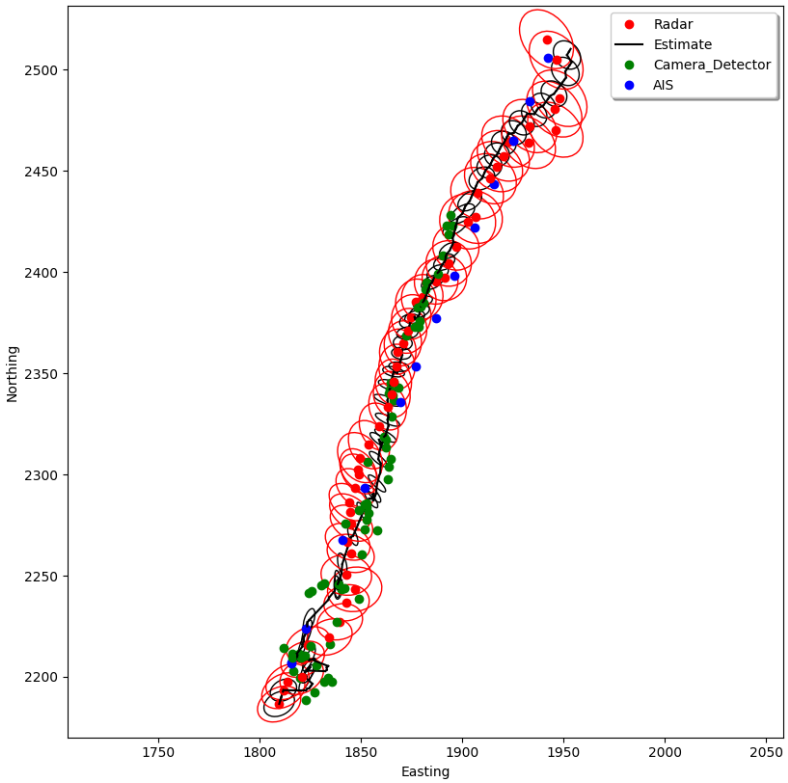


Figure 6.33: Target tracked using RADAR and YOLOv3, test 2

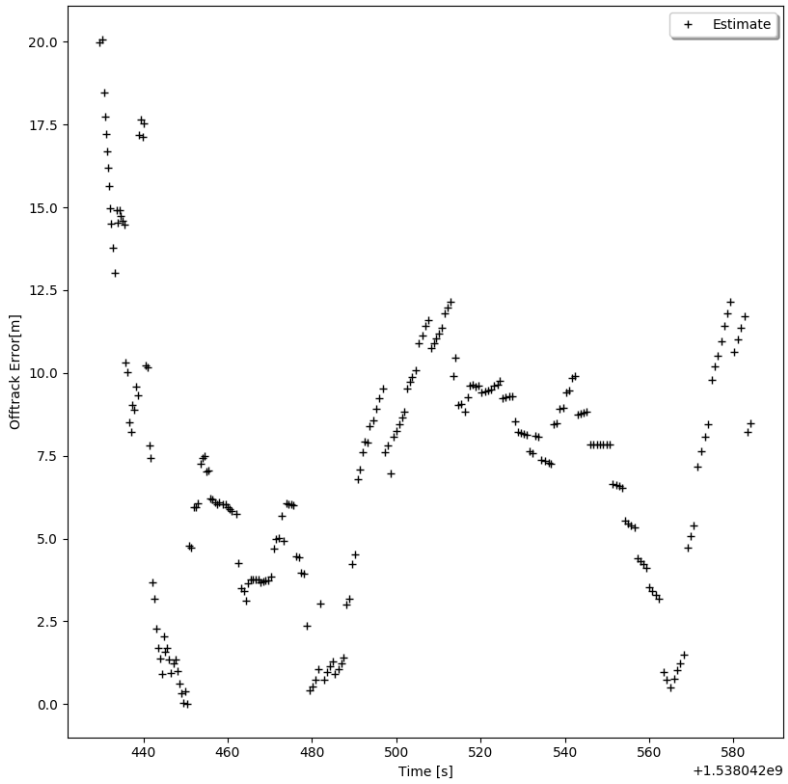


Figure 6.34: Track offset error using RADAR and YOLOv3, test 2

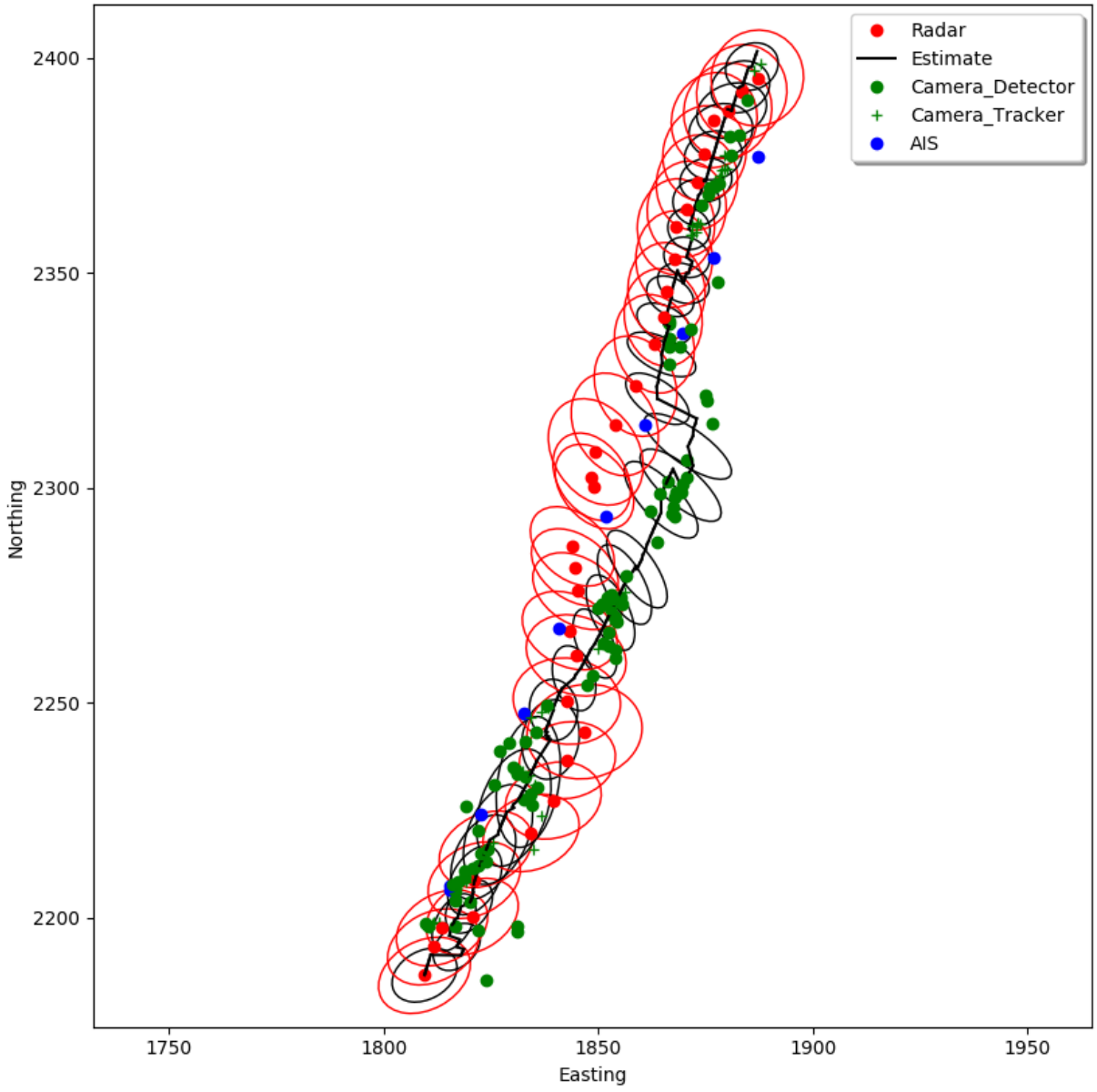


Figure 6.35: Target tracked using RADAR, YOLOv3 and Re^3, test 2

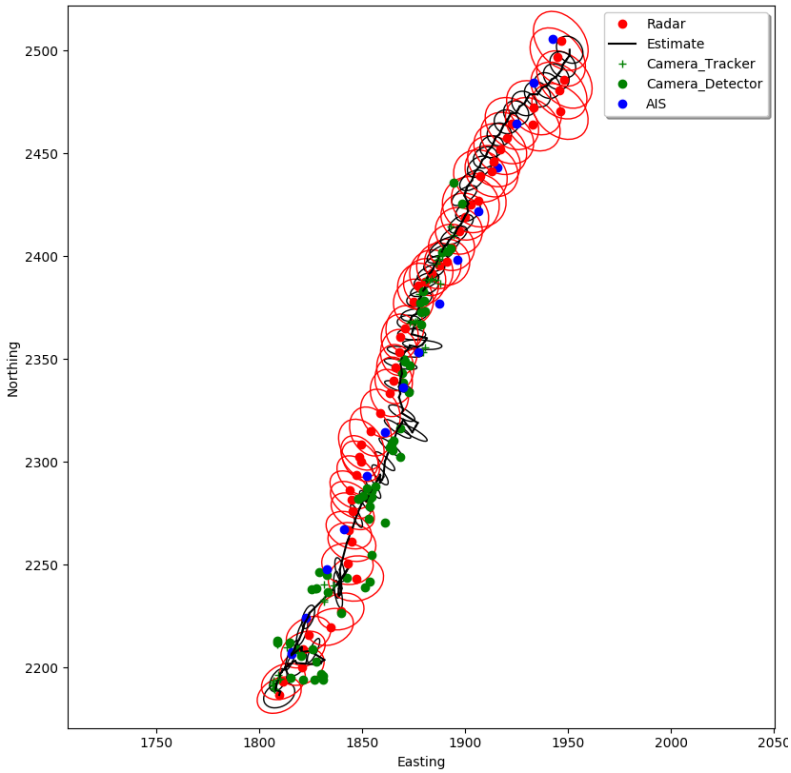


Figure 6.36: Target tracked using RADAR, YOLOv3 and Re^3, test 2

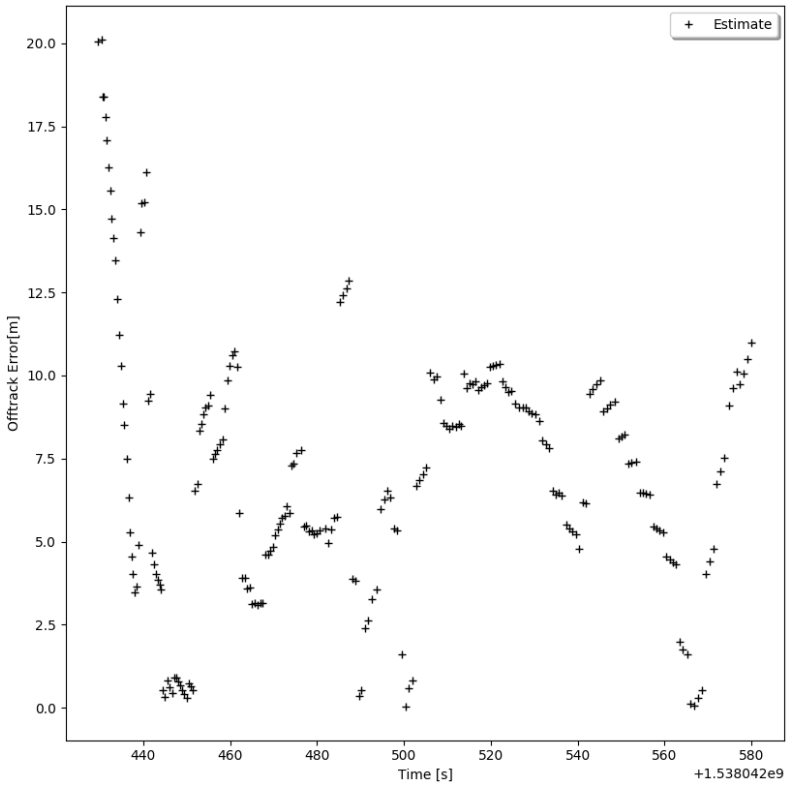


Figure 6.37: Track offset error using RADAR, YOLOv3 and Re^3, test 2

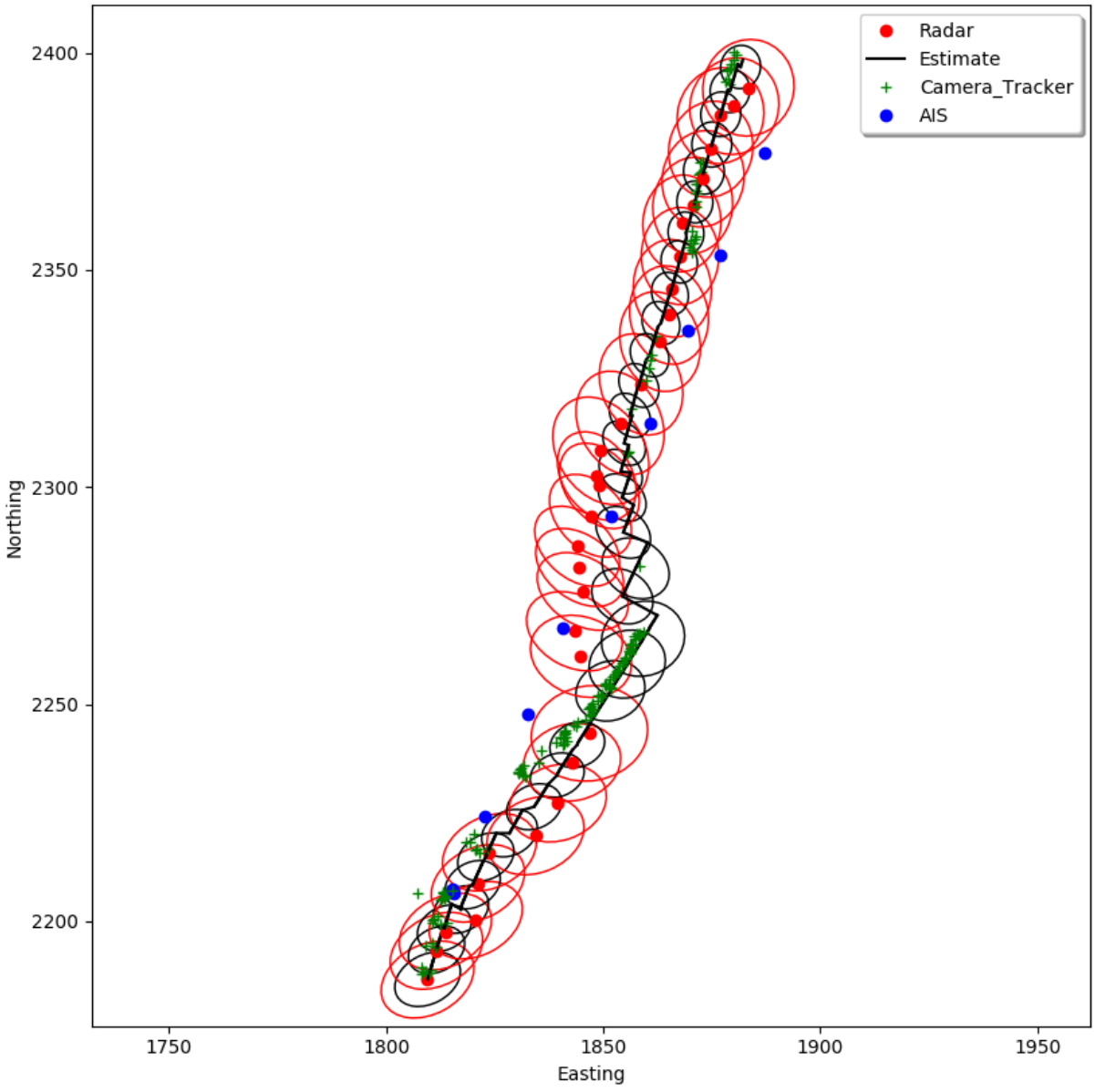


Figure 6.38: Target tracked using RADAR and Re^3 without detector, test 2

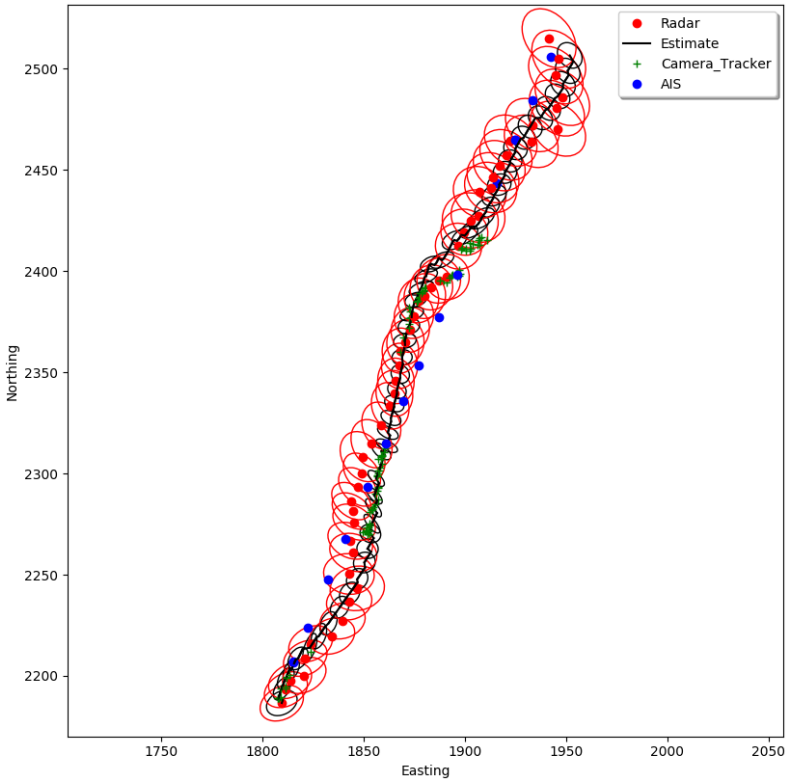


Figure 6.39: Target tracked using RADAR and Re^3 without detector, test 2

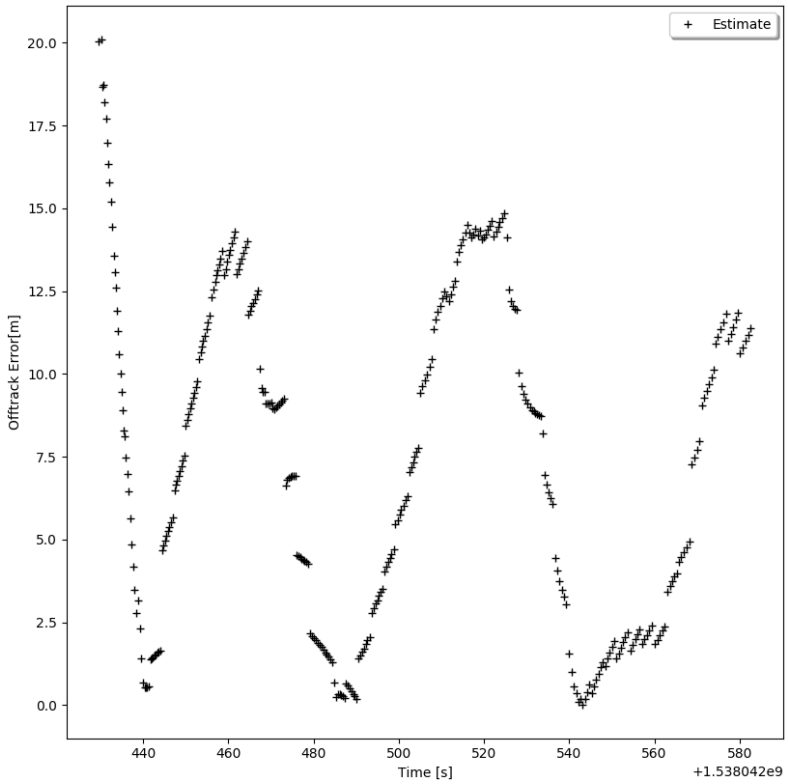


Figure 6.40: Track offset error using RADAR and Re^3 without detector, test 2

6.8 Discussion of test results

The two test-sets shown above were chosen since the time synchronization between INS and Ladybug camera was best on these two. It is still clear, and rather frustrating, that synchronization is poor.

On these data-sets out at open sea, and with few and far apart targets, the performance of the YOLOv3 detector is good. The boat is detected in almost every frame of the cropped image and false detections are very few. The necessary cropping of the images are not surprising as the target boat is only covering a few pixels in the images. Beyond 300m the boat is so small that it is less reliably detected, and beyond 500m there are no true detections. The Ladybug camera provides crisp and clear images with a reasonable resolution. Even better camera systems, with higher resolution or by using more cameras to cover the horizon, might provide sufficient resolution to avoid cropping the pictures. One can also see a slight shift of the vessel position within the crop when the algorithm changes which camera is used for the crop. This is due to imperfect FOV calculation or calibration parameters. More work is needed to find the exact values here. However this shift creates little noise compared to the vessel motion of the ASV.

In Test 1, section 6.7.1, both the YOLO detector and the Re^3 tracker provides valuable aid to the tracking algorithm. The angle calculation to the target vessel bounding box matches the RADAR tracker output within $< 1°$. The Re^3 bounding box tracks the target vessel, and both initialization by YOLO and the RADAR tracks are successful. The implemented track-before-detect method, in which Re^3 must be verified by either YOLO or RADAR before it is used in the fusion algorithm, have shown to be necessary. Without this verification a wrongful tracking by Re^3 and its high output-rate can create divergence of the EKF.

In Test 2, section 6.7.2, the visual methods have a much more challenging problem. As can be seen in picture A.2, the tracked vessel is completely hidden during some maneuvers. The Re^3 tracker handle this as any other occlusion, which works quite OK. When the target is visible again it is however often not within the bounding box and Re^3 is therefore providing erroneous track information. Since the image stabilization is not able to fully counteract the motion, the horizon is not in the center of the image. Should Re^3 therefore be able to track the target, it would anyway be discarded as an error by the stringent limitations set in place for avoiding false tracks in Re^3.

YOLOv3 is in it self unaffected by motion of the vessel and any occlusion. The image is processed and any detections and calculation of angles and position must be handled by the tracking algorithm. The visual detectors are therefore much more robustly fused with other sensors compared to a visual tracker. The errors introduced due to poor stabilization and timing should be possible to correct in a new real-time setup.

The tests demonstrate that the EKF-based tracker have a tendency to be optimistic with regards to the speed of the tracked vessel, which again makes the position uncertainty increase along the axis between the ASV and the tracked vessel. This behaviour is mostly seen when the ASV is following the tracked vessel and where there are very small changes in angle relative between the vessels, as in Test 1, 6.7.1. The behaviour is further mostly observed when the YOLO detector and the Re^3 tracker both give measurement input to the EKF fusion algorithm.

The reason for the uncertainty and inaccuracy of the track introduced by the camera under certain scenarios, as mentioned above, are unknown. The YOLO detector is more accurate, and is given more confidence when updating the track, compared to the Re^3 tracker. The measurements from the RADAR are less frequent and should possibly be weighted more. In the data one can also see that the estimated position from Re^3 oscillates slightly. This oscillation might introduce a slightly higher total velocity which impact the estimated forward speed of the EKF.

In summary the tracking method works well for single or separable targets. The big issue at the moment,

and on the available data, is timing and synchronization. The synchronization issue is a known and feared problem in industries collecting much data, such as surveying using LiDAR or Multibeam echosounders. This is especially true if the time delay of a sensor is not constant as this is not easily corrected in post-processing. In a real-time setup the timing must always be correct and some method for verifying this is needed in a reliable real world COLAV system. It is unknown if ROS is suitable for this stringent time synchronization of large amounts of data in real-time.

Chapter 7

Concluding Remarks and Further Work

The full EKF-based fusion algorithm, utilizing the IPDA-based RADAR tracker [5] and both the visual deep learning YOLO detector [3] and Re^3 tracker [4] as shown in figure 5.1, have been implemented and described in this thesis. The implementation is a version of track-to-track fusion by using the Autosea RADAR tracker to provide RADAR data. The fusion of camera data is not that easy to classify into track-to-track or measurement-level fusion. This is due to the fact that both a tracker and a detector is utilized and no covariance, or other position uncertainty, information is available from the camera, detector or tracker. The overall pipeline might therefore be seen as a mixture of track-to-track and measurement-level fusion.

The implemented way of detecting and tracking boats in image data is influenced by the available data from the experiment in September 2018. The vessels being tracked using RADAR were quite far away and to be able to detect the boats at all using YOLOv3 the images had to be cropped, effectively providing enlarged boats to the CNN network. This method is not optimal for detection at close range, in for instance harbour environments, where the uncropped images would be better suited. The method is further not optimal for tracking multiple objects as the tracker only covers, at most, a part of each image.

Optimally the visual detector would be able to detect even tiny boats in the full image without the need for cropping. The same would be true for an optimal visual tracker. However very small objects are affected too much by the changes in the image, introduced by own vessel motion and poor stabilization, to be reliably tracked. In the Re^3 tracker the object tracked is also converted to a fixed size in the CNN network, limiting the tracking performance of very small parts of the image. Other approaches to tracking or much better stabilization is needed for tracking of very small/distant objects in images.

Throughout the thesis it has been challenging to handle the dynamics of the ASV's own attitude and synchronization between sensors. The real-life autonomous ASV recorded data with it's sharp turns, large speed variations and large associated effect on the ASV attitude, was challenging to combine with the time- and resource intensive deep learning based visual object detector and object tracker. Good timing quickly proved vital, but unfortunately the logged data in separate formats and on two computers made perfect timing difficult.

Real-time calibration and stabilization of images to match the horizon with the straight line in the center of the image are computationally expensive. A reversed way of handling the images, where only the interesting and cropped parts are transformed, might be advantageous from a computational perspective. The CNN structure used in both the YOLO detector and the Re^3 tracker are processing intensive and the tests have shown that each module should have separate GPU processing resources to fully utilize the possibilities.

The dual deep learning based pipeline used in this thesis shows good potential as a visual method for detection and tracking of boats and ships. However since both the visual detector and visual tracker lacks information about distance, and therefore cannot output any indication of uncertainty in position directly, the fusion between RADAR and camera is non-trivial. Using the Jaccard index or IoU for matching and evaluating the tracked object in Re^3 and the detected object in YOLOv3 was rather obvious, as both modules output bounding boxes on similar format. The novel approach introduced in this thesis, for using the RADAR to create bounding boxes which again can be tested using IoU, have shown potential for tracking vessels giving higher tracking rate and lower uncertainty in angle.

All in all the thesis has outlined a new method for combining and evaluating tracks from classic active sensors, where probabilistic methods are used for tracking, together with passive camera sensors, where such probabilistic information is lacking. Much work remains however, in areas such as how to reliably know that the visual tracker is performing correctly and in tuning the EKF parameters. More advanced tracking methods based on for instance JIPDA would be interesting to pursue, but how the visual tracks/measurements and lack of good uncertainty information are to be handled within the tracker is an open question.

7.1 Discussion

Fusion between RADAR and the YOLOv3 detector is the most robust solution presented in this thesis. Both of these give true detections and relatively reliable data. The combination of the active RADAR providing both angle and distance together with the more frequent angle detections by YOLOv3 complement each other nicely. The angle resolution is also much better using the camera compared to the RADAR.

The Re^3 tracker does provide a method for substantially increasing the update frequency of detected targets from RADAR. The robustness of the method is however questionable with the current tracker. It relies currently too much on good stabilization of the images by use of the INS provided attitude. Although any tracker would need relatively stable images between frames so that the area being searched contain the target. This should be possible on unstabilized images as well if the frame-rate and tracking speed is high enough. If the tracker is capable of tracking small enough parts of the image the cropping would also not be necessary.

From a computational perspective the combination of one visual tracker per camera combined with an extremely high end detector could still make sense. A heavily trained and general detector, not focusing on real-time speed, could in the future be at least as reliable as a human crew. The lower speed demand for detection and classification opens up possibilities for more detailed scanning of the whole image.

From a COLAV perspective it might be sufficient to detect ships, boats, kayaks and other surface vessels, but from a broader safety perspective any object in the sea is a potential hazardous threat. For future development it is necessary to also detect and classify most everything. For example: swimmers and divers, floating shipping containers, trees, pallets and even seaweed which could hide other objects or get tangled up in propellers. We therefore have a dual problem where we need to detect everything in our path that is not water, and at the same time classify which objects/vessels which are moving and might prove a threat in the near future. Progress is fast in these areas of research and development. Which methods and which sensor suite will be used in the autonomous future is yet to be known. One realization stands out nevertheless; Any chosen visual method must be highly general to be able to detect unforeseen obstacles.

Timing and synchronization must be given special attention as it affects the entire system. It does not help to have the best sensors in the world if they are giving measurements with a wrong time-stamp.

Special care must therefore be taken when designing the sensory system. The use of 1PPS together with a time-string from the GNSS system to synchronize all other sensors are preferable. Sensors not able to utilize the 1PPS must be tested extensively on all settings to decide the delay of the output.

7.2 Suggestions for further work

New and improved methods for both visual object detection and visual object tracking are continuously being developed. The CenterNet [38] network have for instance an interesting approach to the detection problem in combination with other sensors.

New sensors are also being developed and may prove superior to the currently used sensors. For instance may RADAR technology currently being developed for the automobile industry be perfectly suited for situation awareness in harbour environments, complementing LiDAR sensors.

Below are listed some possible developments:

- Active stabilization using gimbal-like equipment can help limit processing of pictures and improve picture quality, as well as reduce the accuracy demand for the onboard INS or IMU. Also this can introduce cameras with lower FOV, giving less distortion and more resolution in the interesting directions.
- Synchronize all sensors perfectly together.
- Find the accurate parameters for calibration of the whole Ladybug camera system.
- Train the chosen detection method, using transfer learning, on boats and other interesting objects to possibly limit erroneous detections and increase detection certainty of the wanted object types.
- Investigate methods for estimating accuracy and tracking robustness directly in the visual tracker chosen. Check if filtering of the visual tracker output improves estimates.
- Implement the target polygon output from the RADAR tracker as basis for bounding box size in Re^3.
- Replace the CV model with the CT model and evaluate tracking performance.
- Combine and fuse the RADAR and camera with AIS and LiDAR data.
- Implement robust multi-object tracking around the complete $360°$ horizon. Potentially both in the visual tracker and in the world coordinate tracker.
- Further tune the EKF parameters to optimize tracking, or replace the EKF-based tracking with multi-object tracking based on JIPDA or similar.
- Evaluate whether a measurement-level fusion, in which the camera detections are used directly to aid the RADAR data, is more accurate and robust compared to track-to-track level fusion using the standalone RADAR tracker and camera detections.
- To handle time delays, asynchronous sensor measurements and history of detections a Factor Graph based implementation would be very interesting to pursue.

Bibliography

[1] R. A. Olsen, "Deep learning based detection and tracking of ships in camera data for sensor fusion with radar," *Norwegian University of Science and Technology (NTNU)*, 12 2018.

[2] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.

[3] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *arXiv:1804.02767*, 2018.

[4] D. Gordon, A. Farhadi, and D. Fox, "Re3: Real-time recurrent regression networks for visual tracking of generic objects," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 788–795, 2018.

[5] D. K. M. Kufoalor, E. F. Wilthil, I. B. Hagen, E. Brekke, and T. A. Johansen, "Autonomous COLREGs-compliant decision making using maritime radar tracking and model predictive control," in *Euroean Control Conference (Submitted)*, 2019.

[6] E. F. Wilthil, A. L. Flaten, and E. F. Brekke, "A target tracking system for asv collision avoidance based on the pdaf," in *Sensing and Control for Autonomous Vehicles* (P. Fossen and Nijmeijer, eds.), vol. 474, pp. 269 – 288, Alesund, Norway: Springer, 2017.

[7] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Čehovin, G. Fernandez, A. Lukežič, T. Vojir, G. Bhat, and A. Eldesokey, "The Visual Object Tracking Challenge Results VOT2018." `http://data.votchallenge.net/vot2018/presentations/vot2018_presentation.pdf`, 2018.

[8] T. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control*. Wiley, 05 2011.

[9] Veritas, Det Norske, *Ships Diving Support Vessels and Diving Systems*, July 2012.

[10] V. Kamsvaag, "Fusion between camera and lidar for autonomous surface vehicles," Master's thesis, Norwegian University of Science and Technology (NTNU), 2018.

[11] O. K. Helgesen, "Sensor fusion for detection and tracking of maritime vessels," Master's thesis, Norwegian University of Science and Technology (NTNU), 2019.

[12] Y. Bar-Shalom and X.-R. Li, *Multitarget-multisensor tracking : principles and techniques*. Yaakov Bar-Shalom, 1995.

[13] K. Chang, R. Saha, and Y. Bar-Shalom, "On optimal track-to-track fusion," *IEEE Transactions on Aerospace and Electronic Systems*, 1997.

[14] A. Rahman and Y. Wang, "Optimizing Intersection-Over-Union in Deep Neural Networks for Image Segmentation," in *International Symposium on Visual Computing (ISVC)*, 2016.

[15] Z. Zhu, Q. Wang, L. Bo, W. Wu, J. Yan, and W. Hu, "Distractor-aware siamese networks for visual object tracking," in *European Conference on Computer Vision*, 2018.

[16] D. Hermann, R. Galeazzi, J. C. Andersen, and M. Blanke, "Smart sensor based obstacle detection for high-speed unmanned surface vehicle," *International Federation of Automatic Control*, vol. 28, no. 16, pp. 190–197, 2015.

[17] L. Elkins, D. Sellers, and W. R. Monach, "The autonomous maritime navigation (AMN) project: Field tests, autonomous and cooperative behaviors, data fusion, sensors, and vehicles," *Journal of Field Robotics*, vol. 27, pp. 790–818, nov 2010.

[18] International Maritime Organization, *Resolution MSC.74(69) (adopted on 12 May 1998) Adoption of new and amended performance standards, AIS*, 1998.

[19] International Maritime Organization, *SN Circular 227 Guidelines for the installation of a Shipborne Automatic Identification System (AIS)*, 2003.

[20] SIMRAD, "Broadband 4G™ Radar." `http://www.navico-commercial.com/en-US/Products/Radars/Broadband-Solid-State/Broadband-4G-Radar-Pro-en-us.aspx`, 2017.

[21] D. Scaramuzza, "Omnidirectional Camera," *Computer Vision: A Reference Guide, Editors: Katsushi Ikeuchi, ISBN: 978-0-387-30771-8 (Print) 978-0-387-31439-6 (Online), Springer, April, 2014*, 2014.

[22] Geyer, Christopher and Pajdla, Tomas and Daniilidis, Kostas, "Short course on omnidirectional vision," 2003.

[23] V. Grassi and J. Okamoto, "Development of an omnidirectional vision system," *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, vol. 28, pp. 58 – 68, 03 2006.

[24] Y. Shen, H. Shin, W. Sung, S. Khim, H. Kim, and P. Rhee, "Evolutionary adaptive eye tracking for low-cost human computer interaction applications," *Journal of Electronic Imaging*, vol. 22, no. 1, 2013.

[25] PointGrey, FLIR, "TAN2012009-Geometric Vision using Ladybug Cameras." `https://www.ptgrey.com/support/downloads/10400`, 2012.

[26] PointGrey, FLIR, "TAN2008019-Overview of the Ladybug Image Stitching Process." `https://www.ptgrey.com/support/downloads/10378`, 2008.

[27] SONY-Semicon, "The Industry's Smallest Pixel Size Class for Industrial Applications and ITS (Intelligent Traffic Systems) Applications A Variety of Functions." `https://www.sony-semicon.co.jp/products{_}en/IS/sensor0/img/product/cmos/IMX264{_}265{_}Flyer.pdf`, 2018.

[28] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[29] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[30] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2015.

[31] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "STRIVING FOR SIMPLICITY: THE ALL CONVOLUTIONAL NET," *International Conference on Learning Representations (ICLR)*, 2015.

[32] I. Vasilev, *Python deep learning : exploring deep learning techniques and neural network architectures with PyTorch, Keras, and TensorFlow*. Packt Publishing Limited, 2019.

[33] M. Bjelonic, "YOLO ROS: Real-time object detection for ROS." https://github.com/leggedrobotics/darknet_ros, 2018.

[34] J. Redmon, "Darknet: Open source neural networks in c." http://pjreddie.com/darknet/, 2013–2016.

[35] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. Lawrence Zitnick, "Microsoft COCO: Common Objects in Context," *European Conference on Computer Vision (ECCV)*, 2014.

[36] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[37] Y. Li, Y. Chen, N. Wang, and Z. Zhang, "Scale-Aware Trident Networks for Object Detection," *arXiv preprint arXiv:1901.01892v1*, 2019.

[38] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "CenterNet: Keypoint Triplets for Object Detection," *arXiv preprint arXiv:1904.08189v3*, 2019.

[39] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as Points," *arXiv preprint arXiv:1904.07850v2*, 2019.

[40] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[41] T.-W. Hui, X. Tang, and C. C. Loy, "A Lightweight Optical Flow CNN - Revisiting Data Fidelity and Regularization," *arXiv preprint arXiv:1903.07414*, 2019.

[42] A. He, C. Luo, X. Tian, and W. Zeng, "A Twofold Siamese Network for Real-Time Object Tracking," *arXiv preprint arXiv:1802.08817v1*, 2018.

[43] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu, "High performance visual tracking with siamese region proposal network," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[44] G. Wang, C. Luo, Z. Xiong, W. Zeng, and M. Research Asia, "SPM-Tracker: Series-Parallel Matching for Real-Time Visual Object Tracking," *arXiv preprint arXiv:1904.04452v1*, 2019.

[45] Y. Bar-Shalom, F. Daum, and J. Huang, "The Probabilistic Data Association Filter: Estimation in the presence of measurement origin uncertainty," *IEEE Control Systems*, vol. 29, pp. 82–100, dec 2009.

[46] E. Brekke, "Fundamentals of sensor fusion." Unpublished textbook for a future course at NTNU, 02 2019.

[47] H. W. Kuhn, *The Hungarian method for the assignment problem*, vol. 2. Wiley, mar 2010.

[48] S. Blackman and R. Popoli, *Design and Analysis of Modern Tracking Systems (Artech House Radar Library)*. Artech House, 1999.

[49] J. B. Orlin and Y. Lee, "QuickMatch: A Very Fast Algorithm for the Assignment Problem," tech. rep., Sloan School of Management Massachusetts Institute Technology Cambridge, 1993.

[50] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, p. 35, mar 1960.

[51] Mohinder S. Grewal and Angus P. Andrews, "Applications of Kalman Filtering in Aerospace 1960 to the Present," *IEEE Control Systems Magazine*, pp. 69–78, jun 2010.

[52] R. Brown and P. Hwang, *Introduction to Random Signals and Applied Kalman Filtering with MATLAB Exercises, 4th Edition*. Wiley, 2012.

[53] S. Fossen and T. I. Fossen, "eXogenous Kalman Filter (XKF) for Visualization and Motion Prediction of Ships using Live Automatic Identification System (AIS) Data," *Modeling, Identification and Control: A Norwegian Research Bulletin*, vol. 39, no. 4, pp. 233–244, 2018.

[54] D. K. M. Kufoalor, E. F. Brekke, and T. A. Johansen, "Proactive Collision Avoidance for ASVs using A Dynamic Reciprocal Velocity Obstacles Method *," in *International Conference On Intelligent Robots and Systems (IROS18)*, 2018.

[55] D. Musicki, R. Evans, and S. Stankovic, "Integrated probabilistic data association," *IEEE Transactions on Automatic Control*, vol. 39, pp. 1237–1241, jun 1994.

[56] Y. Bar-Shalom and E. Tse, "Tracking in a Cluttered Environment with Probabilistic Data Association," *Automatica, Vol. 1 1*, pp. 451–460, 1975.

[57] S. Kato, S. Tokunaga, Y. Maruyama, S. Maeda, M. Hirabayashi, Y. Kitsukawa, A. Monrroy, T. Ando, Y. Fujii, and T. Azumi, "Autoware on board: Enabling autonomous vehicles with embedded systems," in *ICCPS18*, pp. 287–296, IEEE Press, 2018.

[58] Y. Boers and J. Driessen, "A multi target track before detect application," in *Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'03)*, 2003.

Appendix A

Appendix

A.1 Examples of motion of the ASV during maneuvers

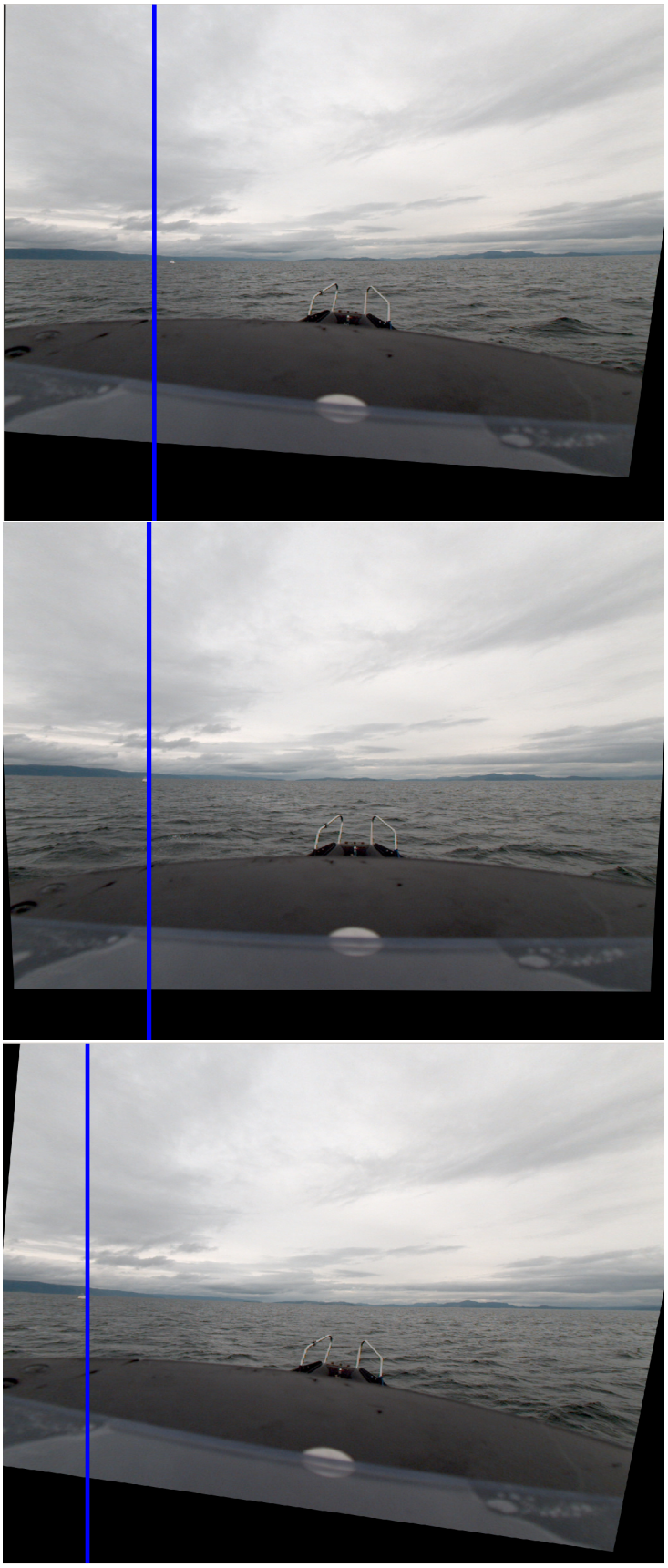


Figure A.1: Examples of how motion from ASV maneuvers affect the camera images looking forward

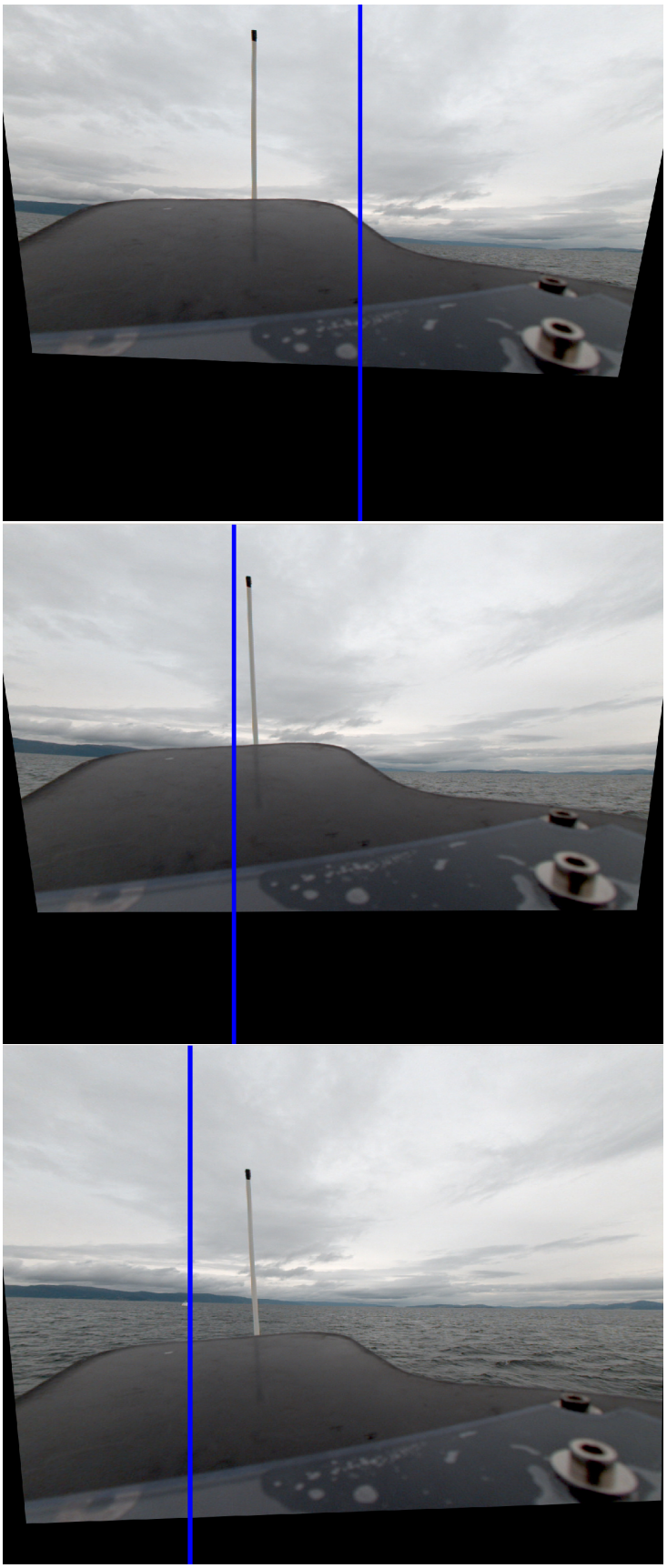


Figure A.2: Examples of how motion from ASV maneuvers affect the camera images looking to port

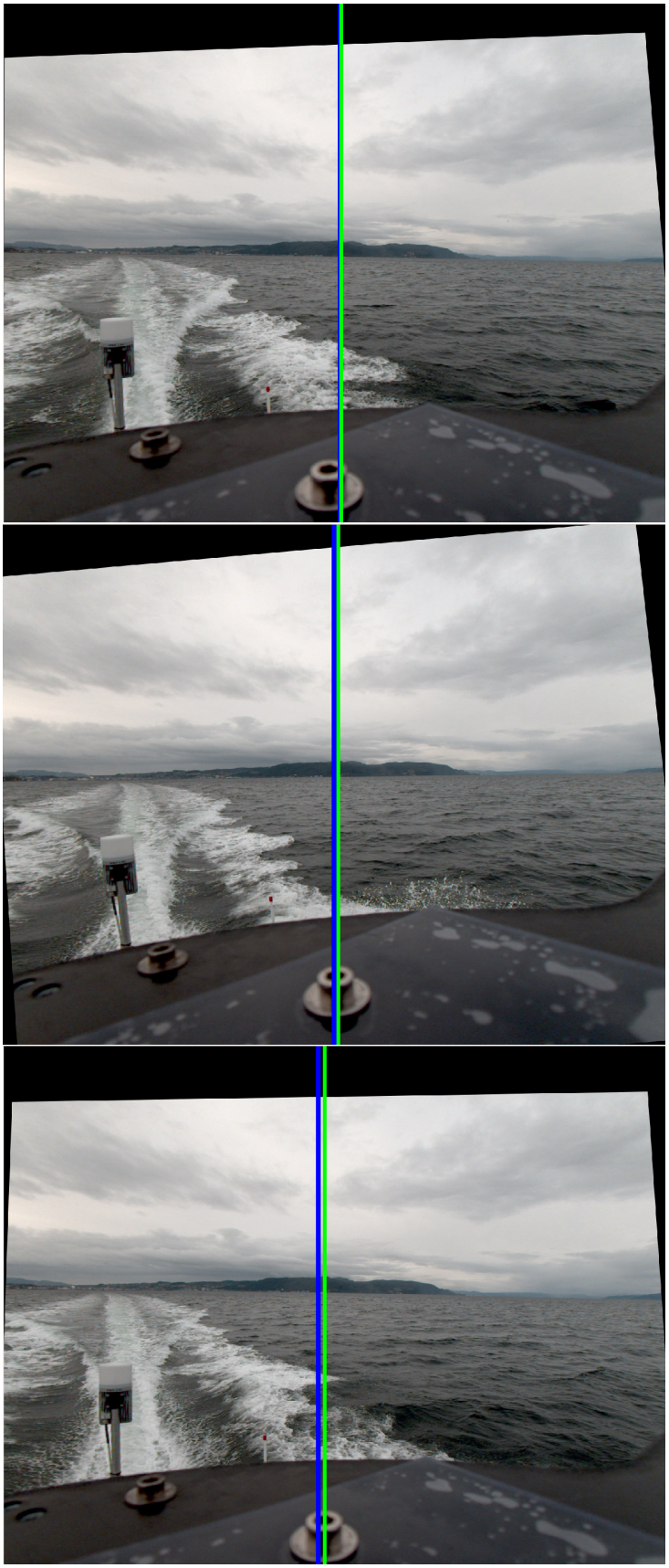


Figure A.3: Examples of how motion from ASV maneuvers affect the camera images looking aft

A.2 Results from tests of the YOLO detector

These results are from the specialization project report [1].

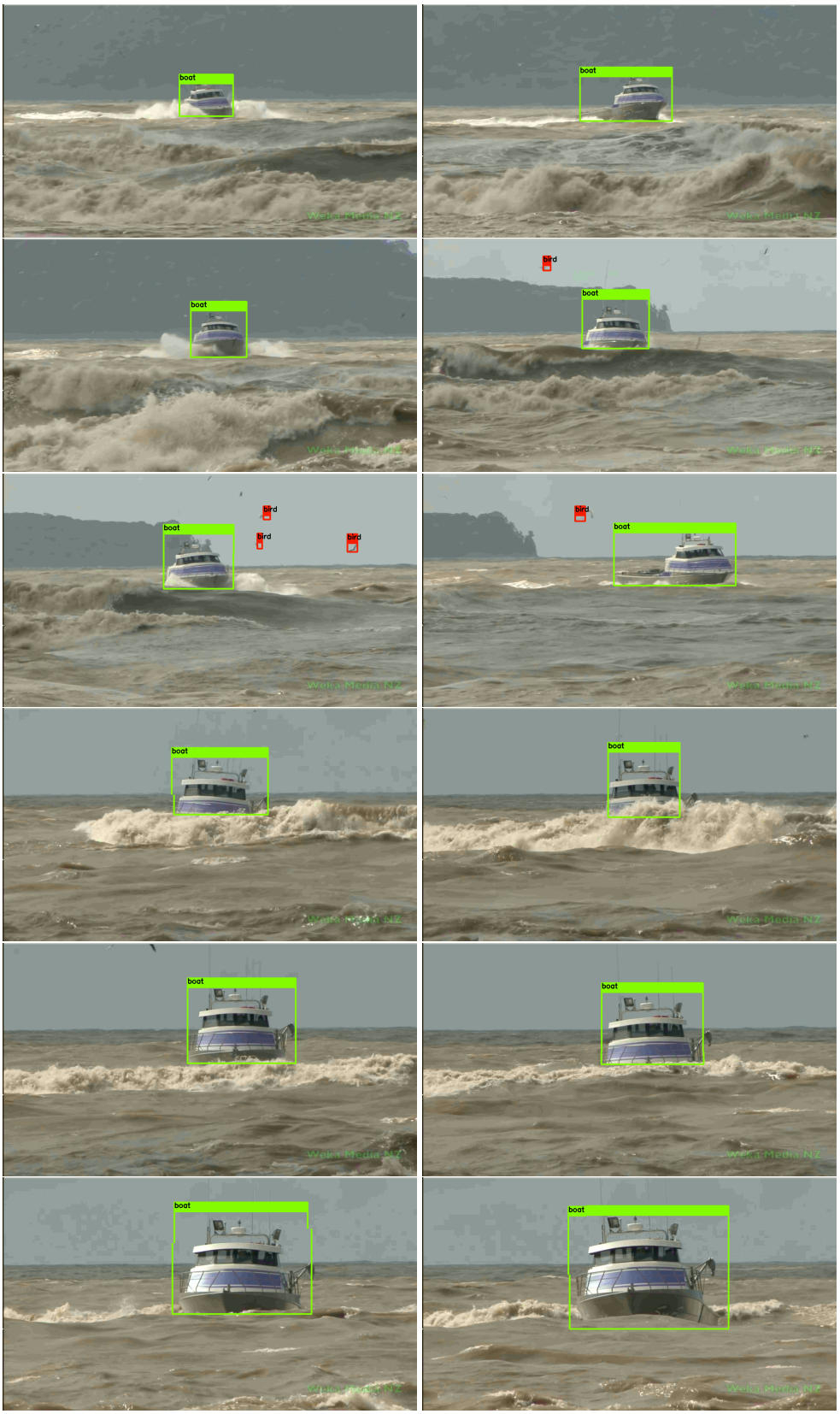


Figure A.4: Examples of the performance of the YOLOv3 detector. Part 1 of 2. Video from YouTube: Weka Digital Media NZ

Figure A.5: Examples of the performance of the YOLOv3 detector. Part 2 of 2. Video from YouTube: Weka Digital Media NZ

Figure A.6: Examples of the performance of the YOLOv3-tiny detector. Video from YouTube: Weka Digital Media NZ

A.3 Results from tests of the Re^3 tracker

These results are from the specialization project report [1].

Figure A.7: Examples of the tracking performance of the Re^3 tracker. The tracker was initialized manually with a bounding box containing the boat. Images from experiment with ”Ocean Space Drone 1”. The images are cropped for clarity

Figure A.8: Examples of the tracking performance of the Re^3 tracker. The tracker was initialized manually with a bounding box containing the boat. Video from YouTube: Weka Digital Media NZ

A.4 Results from combined Re^3 tracker and YOLO detector

These results are from the specialization project report [1].

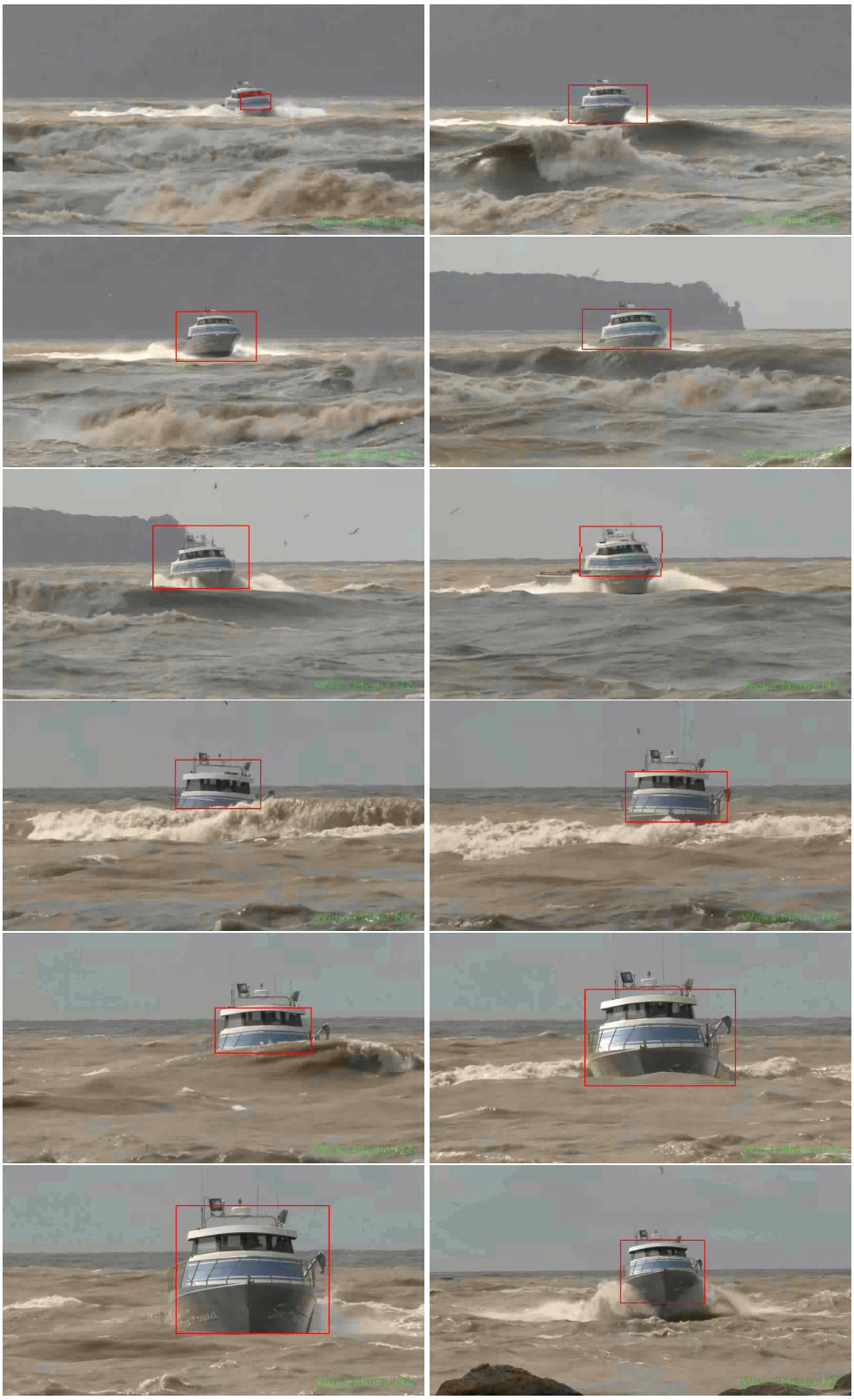


Figure A.9: Examples of the performance of the combined YOLOv3-tiny detector and the Re^3 tracker. Part 1 of 2. Notice that it is the ”bad” tiny version of YOLO that is used. Video from YouTube: Weka Digital Media NZ

Figure A.10: Examples of the performance of the combined YOLOv3-tiny detector and the Re^3 tracker. Part 2 of 2. Notice that it is the ”bad” tiny version of YOLO that is used. Video from YouTube: Weka Digital Media NZ

Runar A. Olsen

Sensor fusion of radar data with deep learning based detection and tracking of ships in camera images

NTNU
Norwegian University of
Science and Technology