FINAL YEAR PROJECT REPORT

NORWEGIAN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Deep learning based detection and tracking of ships in camera data for sensor fusion with radar

Runar André Olsen

Supervisor:
Edmund F. Brekke

Co-supervisors:
Arild Hepsø
Kenan Trnka
Erik Wilthil

Trondheim
December 18, 2018

Preface

This report is the result of the final year project course TTK4551 for the two year MSc program in Cybernetics and Robotics at Department of Engineering Cybernetics at Norwegian University of Science and Technology (NTNU). The report outlines a combined detection and tracking method in images for collision avoidance (COLAV), and should be considered as a prestudy for sensor fusion in the upcoming Master's Thesis of spring 2019.

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 help during the experiments and especially Erik Wilthil for valuable advice on the report outline.

Runar André Olsen

Trondheim, December 18, 2018

Abstract

As autonomous vessels start to populate our roads, waterways and skies the demand for robust and safe collision avoidance (COLAV) systems will be high. This report describes a detection and tracking system for image data captured using omnidirectional (360°) camera systems on board Autonomous Surface Vessel (ASV)s which can further be used in a sensor fusion with active sensors such as RAdio Detection And Ranging (RADAR) and Light Detection And Ranging (LiDAR) together with ownship position and attitude estimates from an Inertial Navigation System (INS). Both the image detector and the image tracker are deep learning Artificial Intelligence (AI) based and trained to detect and track boats and ships. The overall aim is a robust, real-time collision avoidance (COLAV) system to enhance safety at sea.

The pipeline described in the report shows good performance and also inherit possibilities for further optimization through training of the under-laying deep learning networks.

Contents

Acronyms

Re^3 Real-time Recurrent Regression. ii, iii, 2, 3, 12, 17, 21–23, 25, 27–30, 35–38, 44–49, 51

1PPS 1 Pulse Per Second. 28, 33

2D 2 Dimensional. 31

3D 3 Dimensional. 15

AI Artificial Intelligence. i, 2, 17

AIS Automatic Identification System. ii, 2, 5, 7

API Application Programming Interface. 12, 13, 15, 29, 34

ASV Autonomous Surface Vessel. i, 1, 2, 23, 28, 30, 31, 33, 51

BODY Body-fixed. 5–7, 10, 30, 31

CNN Convolutional Neural Network. ii, 11, 17–21

COCO Common Objects in COntext. 20, 38

COLAV collision avoidance. i, 2

COR Center Of Rotation. 10

CPU Central Processing Unit. 19, 49

CUDA Compute Unified Device Architecture. 19, 49

ECEF Earth Centered Earth Fixed. 5, 6

ENU East North Up. 6

FMCW Frequency Modulated Continuous Wave. 8
FN False Negative. 24
FOV Field Of View. 10
FP False Positive. 24, 25

GNSS Global Navigation Satellite System. 7, 9, 10, 28, 31, 33
GPU Graphics Processing Unit. 19, 21, 44, 49

IMU Inertial Measurement Unit. 9
INS Inertial Navigation System. i, ii, 2, 8–10, 28, 30, 31, 33, 51
IoU Intersection over Union. 25, 26, 29, 47
IPDA Integrated Probabilistic Data Association. 27, 30, 51
IR infrared. 2

KF Kalman Filter. 10

LiDAR Light Detection And Ranging. i, 2, 3, 7, 23
LOS Line of Sight. 24
LSTM Long Short-Term Memory. ii, 18, 19, 21

ML Machine Learning. 17

NED North East Down. 5–7, 14, 30, 31
NM Nautical Miles. 8
NMEA National Marine Electronics Association. 28, 33, 34
NTNU Norwegian University of Science and Technology. i, 2, 33

PDAF Probabilistic Data Association Filter. 27

RADAR RAdio Detection And Ranging. i, ii, 2, 3, 7–12, 23, 27–31, 33, 35, 51
RGB Red Green Blue. 17
RNN Recurrent Neural Network. 18, 19
ROS Robot Operating System. iii, 27–30, 34, 35
RPAS Remotely Piloted Aircraft Systems. 1

RPM Rounds Per Minute. 8
RS-232 Electronic Industries Association (EIA) Recommended Standard 232. 34, 35
RTK Real-Time Kinematics. 31

SLAM Simultaneous Localization And Mapping. 23
SNR Signal to Noise Ratio. 24

TN True Negative. 24
TP True Positive. 24
TTL Transistor–Transistor Logic. 34, 35

USB Universal Serial Bus. 33, 34

VHF Very High Frequency. 7

WGS84 World Geodetic System of 1984. 5

YOLO You Only Look Once. ii, iii, 2, 3, 12, 17, 19–21, 23, 25, 27–30, 38–49, 51

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) often called simply drones, have hit the mass marked. 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. The common denominator for the systems in the private market is that they are relatively cheap and safe due to the small size and low weight. The more expensive and potentially dangerous systems in the commercial and military market have much stricter demands concerning safety and reliability. 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.

1.1 Background and Motivation

This report considers Autonomous Surface Vessel (ASV) which are marine surface vehicles such as ships and boats. 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 [1]. 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 completely 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 [2].

In contrast other vessels can 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 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 cameras detecting visible light (daylight) and infrared (IR) cameras give detailed information about the object, but need interpretation to be useful to computers.

The problem to be investigated in this report 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 Norwegian University of Science and Technology (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 collision avoidance (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 the images would have given good distance information. This is unfortunately not feasible on the available vessel. Another obvious limitation is that 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 can 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 thesis of Kamsvåg [3] which this report in part builds on. LiDAR have great details and high update frequency, but have short range and is more affected by atmospheric conditions such as rain, snow or fog compared to RADAR.

In this report we suggest a novel approach for detection and tracking of targets in image data and video. It is based on two Artificial Intelligence (AI) based methods where a hybrid image/video tracker Real-time Recurrent Regression (Re^3) is initialized and updated by an image detector You Only Look Once (YOLO)v3. The AI methods are robust against changes in lighting conditions, appearance and size. The underlying deep learning networks can furthermore be optimized on the 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 explained in chapter 3.

Hermann et al. [4] 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. [5] 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 had performed several successful demonstrations they conclude that there are still big limitations of the system.
The previous work by Kamsvåg [3] with fusion of passive and active sensors, i.e. camera images and LiDAR, did not use any image tracker, but proposed to fused the measurements from an image detector (Faster R-CNN) with the LiDAR measurements directly. 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.

Further on the tracks from the image tracker are to be associated and fused with RADAR measurements from other contributers in the Autosea project. [6] [7] This will be a large part of the authors upcoming Master's Thesis. The RADAR measurements are shown in the suggested pipeline in figure 5.1 and will be presented through the report together with the image processing from the cameras.

1.2 Report Outline

The rest of the report 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.
- 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.
- Chapter 5 explains the suggested tracking pipeline for sensor fusion with omnidirectional (360°) camera and RADAR.
- 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.
- Chapter 7 includes an overview of the status of the proposed pipeline. It gathers the main takeaways in the report and also suggests future work that has not been completed within this project.

Chapter 2

Sensors

In modern vessels there are an abundant number of sensors which give valuable information to the helmsman and potential guidance and navigation systems. If communication of data such as AIS is installed much of this data can also be transmitted to other vessels.

2.1 Reference Frames

The dynamics of a vessel can be expressed in several different frames. The most common used for surface vessels are [1]:

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 accurate 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 is also referenced to the WGS84 ellipsoid.

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.

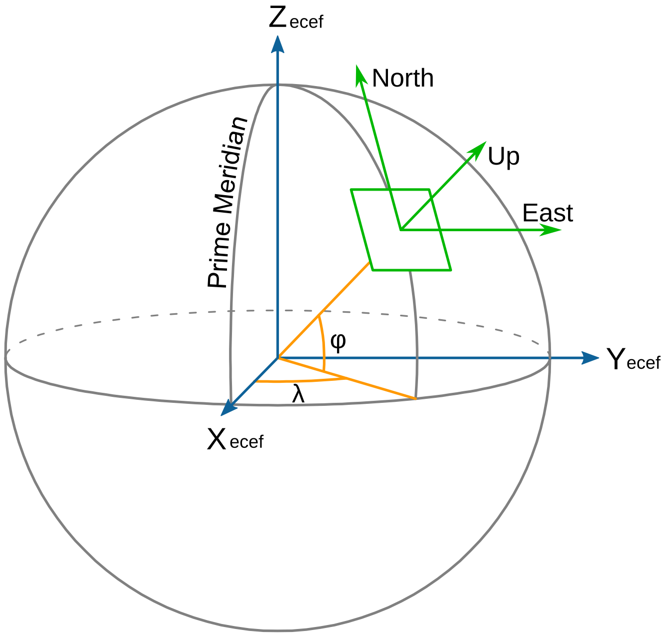


Figure 2.1: East North Up (ENU) is similar to NED but is rotated so that up is positive. Here shown compared to Earth Centered Earth Fixed (ECEF). Wikimedia Commons.

Rotation and rigid body transformation 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.

$$SO(3) = \{\mathbf{R} \in \mathbb{R}^{3\times 3} \mid \mathbf{R}\mathbf{R}^T = 1, det\mathbf{R} = 1\}$$

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

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

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

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.

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.

$$\mathbf{R}_b^n(\Phi) = \mathbf{R}_{x/b}^n(\phi)\mathbf{R}_{y/b}^n(\theta)\mathbf{R}_{z/b}^n(\psi) = \begin{bmatrix} c\psi c\theta & -s\psi c\phi + c\psi s\theta s\phi & s\psi s\phi + c\psi c\phi s\theta \\ s\psi c\theta & c\psi c\phi + s\phi s\theta s\psi & -c\psi s\theta + s\theta s\psi c\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix}$$

With the correct rotation and translation, 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. All ships above 500 gross tonnage and all passenger ships of any size should have installed AIS [8]. 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 therefor 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 sends out 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.

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). 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

$$r = \frac{c \cdot t}{2} \tag{2.1}$$

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 [9]. 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 two 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. The data output from this RADAR also 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 etc.

2.4.1 Inertial Navigation System (INS)

The 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

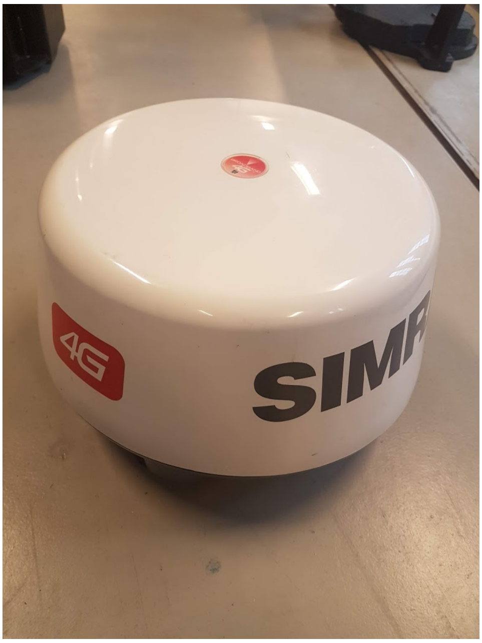


Figure 2.2: Simrad Broadband 4G maritime RADAR

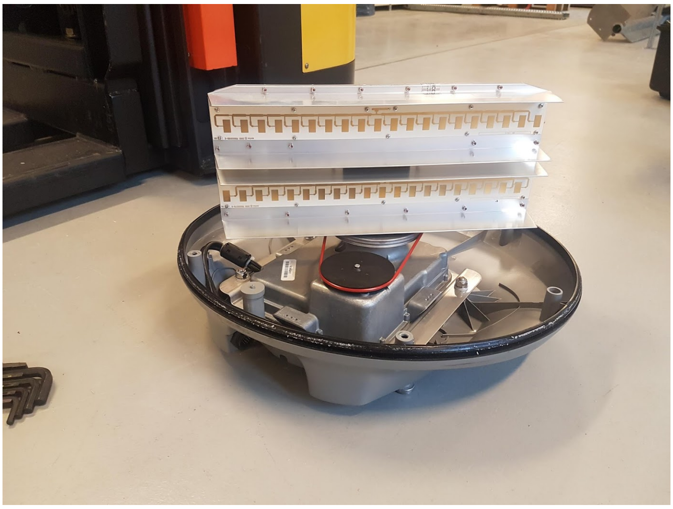


Figure 2.3: Simrad RADAR with protective dome removed

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 [1] and is a sensor fusion algorithm based on nonlinear Kalman Filter (KF) that is running within the INS.

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

- 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

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.

2.4.2 Digital Daylight Camera

Most cameras today use digital sensor arrays with several Megapixels. The Ladybug camera 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 ”fisheye”, and catadioptric cameras (camera and mirror systems) can provide 360 degree coverage using only one lens [10][11]. 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 lenses 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 [12][11]. 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 Convolutional Neural Network (CNN) for detections and would require significant training effort.

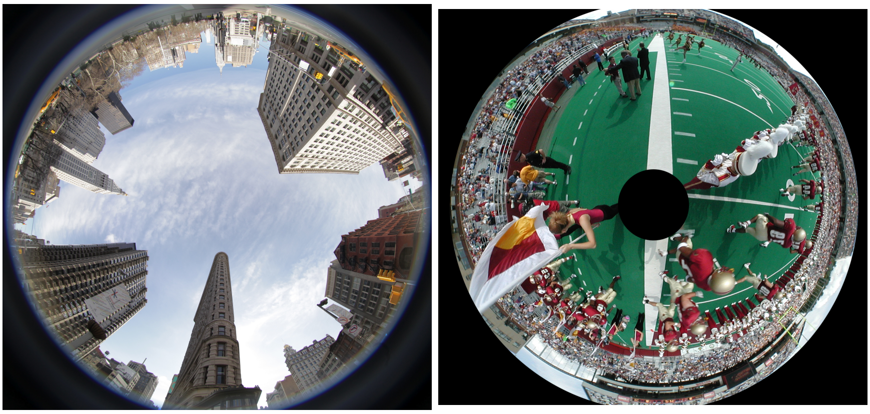


Figure 2.4: Fisheye camera and hyperbolic mirror camera compared. Wikimedia Commons and Columbia University CAVE project.

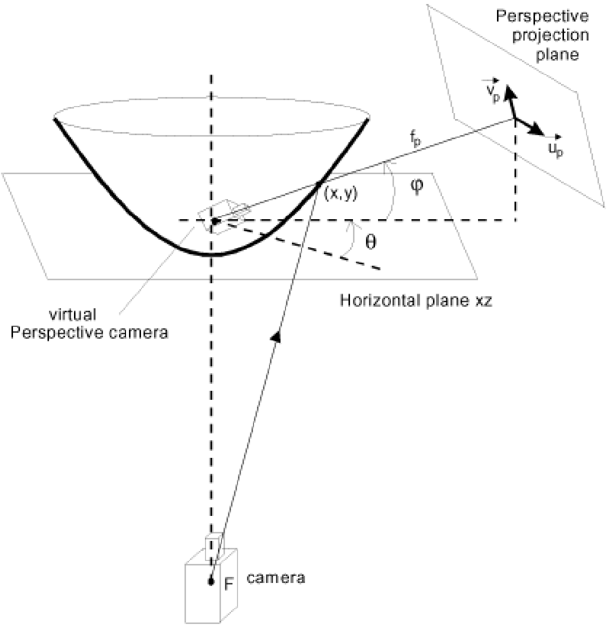


Figure 2.5: Hyperbolic mirror camera principle [12]

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.

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 [11][13]. 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 is valuable. The exact camera system used in this report is the ”Ladybug 5+” from ”FLIR® Machine Vision”.

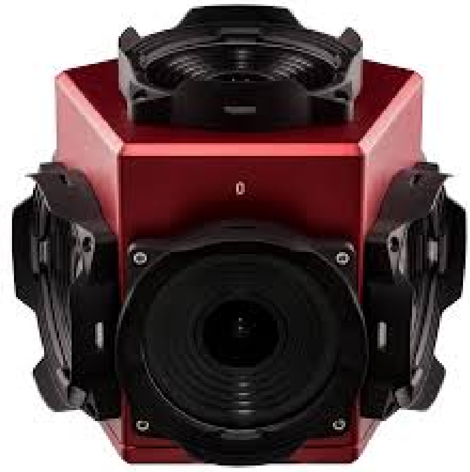


Figure 2.6: The FLIR Ladybug 5+ camera system

2.4.4 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 primarily two 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, performs best on images that have somewhat quadratic shape as the images are converted into a fixed size before fed through the network [14] [15]. Secondly 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 must be used during processing. The cameras need no further calibration or adjustment between each other. The images are however output from the Ladybug system uncalibrated and have a large barrel distortion as can be seen from images 2.8, 2.10 and 2.11. The calibration parameters can be extracted from the Ladybug API as seen in table 2.2. To convert from pixel location to the needed 3D ray a transformation must be performed using the camera matrices 2.5 through 2.12 [16] [17].

The angle of view along the vertical and horizontal axis of each camera is calculated as follows:

$$\alpha = 2\arctan\frac{d}{2f} \tag{2.2}$$

$$\alpha_y = 2\arctan\frac{11.1}{2\cdot 4.4} = 103.19^\circ \tag{2.3}$$

$$\alpha_x = 2\arctan\frac{7.7}{2\cdot 4.4} = 82.37^\circ \tag{2.4}$$

Where d is the sensor size in mm (height or width) and f is the focal length in mm [18].

The cameras have wide angle of view as can be seen in eqation 2.3 and 2.4, and introduce much barrel distortion which must be counteracted. The rectified images can be modeled using the standard perspective camera model (pinhole camera) [16].

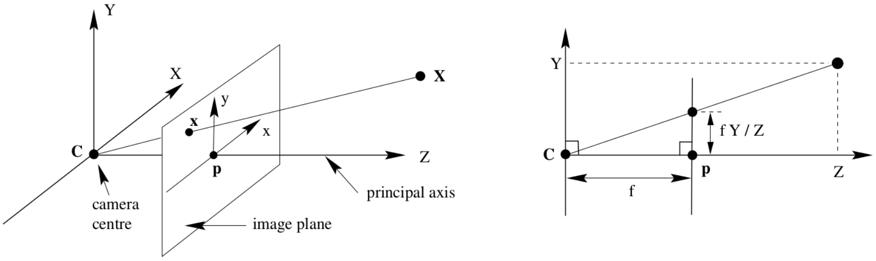


Figure 2.7: The pinhole camera model

$$\mathbf{P} = \mathbf{K}\left[\mathbf{R}|\mathbf{t}\right] \tag{2.5}$$

$$\mathbf{K} = \begin{bmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

$$\left[\mathbf{R}|\mathbf{t}\right] = \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} & t_1 \\ r_{2,1} & r_{2,2} & r_{2,3} & t_2 \\ r_{3,1} & r_{3,2} & r_{3,3} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

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 length, the x_0 and y_0 are the center point offset of the sensor and the s parameter is a skew value which is usually 0.

$\begin{bmatrix}\mathbf{R}|\mathbf{t}\end{bmatrix}$ 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

$$\mathbf{u} = \mathbf{K}\hat{\mathbf{x}} \tag{2.8}$$

$$\mathbf{u} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \hat{\mathbf{x}} = \begin{bmatrix} x_{corrected} \\ y_{corrected} \\ 1 \end{bmatrix} = \begin{bmatrix} (1 + k_1 r^2 + k_2 r^4 + k_3 r^6)x \\ (1 + k_1 r^2 + k_2 r^4 + k_3 r^6)y \\ 1 \end{bmatrix} \tag{2.9}$$

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, r = \sqrt{(x_\mathrm{d} - x_0)^2 + (y_\mathrm{d} - y_0)^2} \tag{2.10}$$

$$\mathbf{x} = \begin{bmatrix}\mathbf{R}|\mathbf{t}\end{bmatrix} \mathbf{U} \tag{2.11}$$

$$\mathbf{U} = \begin{bmatrix} X^n \\ Y^y \\ Z^n \\ 1 \end{bmatrix} \tag{2.12}$$

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/distrortion center to the distorted coordinate x_d, y_d. $\mathbf{U}$ is homogeneous world coordinates given in NED. The distortion coefficients are provided as a calibration file for the Ladybug camera, but can also be calculated using software such as OpenCV or Matlab given images of known checkerboard patterns or similar. See figure 2.9. Note that other calibration parameters than barrel distortion is not shown in equation 2.9, but can also be calibrated.

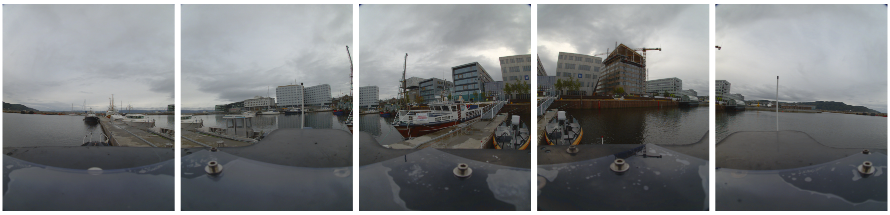


Figure 2.8: The 5 images that form the panoramic view from the Ladybug camera

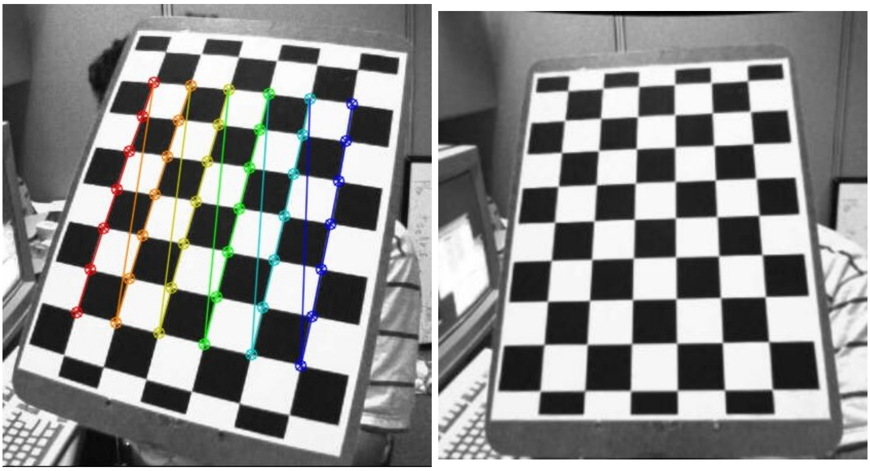


Figure 2.9: Calibration of barrel distortion using OpenCV. [19]

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:

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

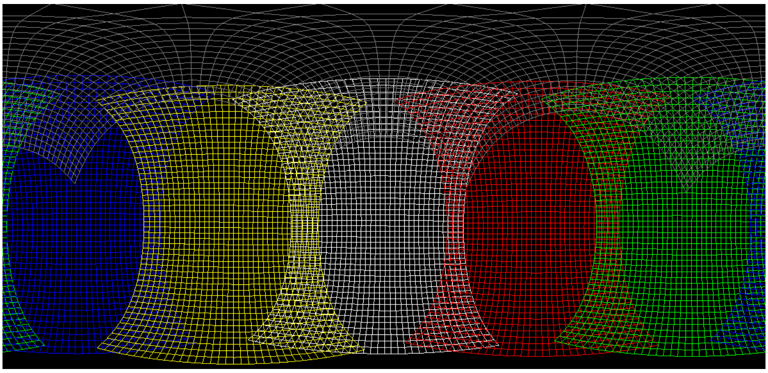


Figure 2.10: 2D polygon mesh of stitched Ladybug images [17]

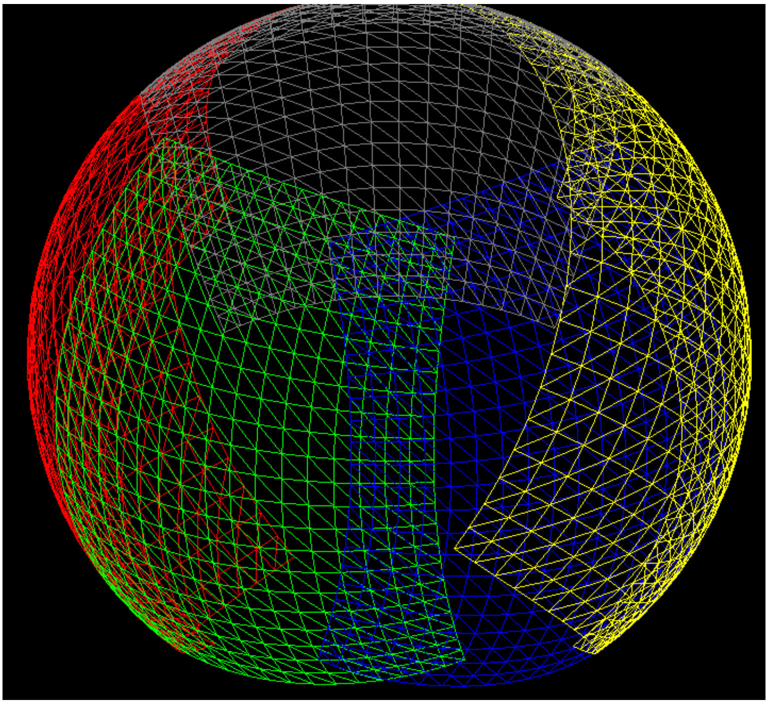


Figure 2.11: 3D polygon mesh of stitched Ladybug images [17]

Chapter 3

Deep Learning Methods

Deep learning is a modern 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 [20] and [21]. 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 the number of objects the detector should look for.

3.1 Convolutional Neural Network (CNN)

CNNs are a special type of neural networks used primerily 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,

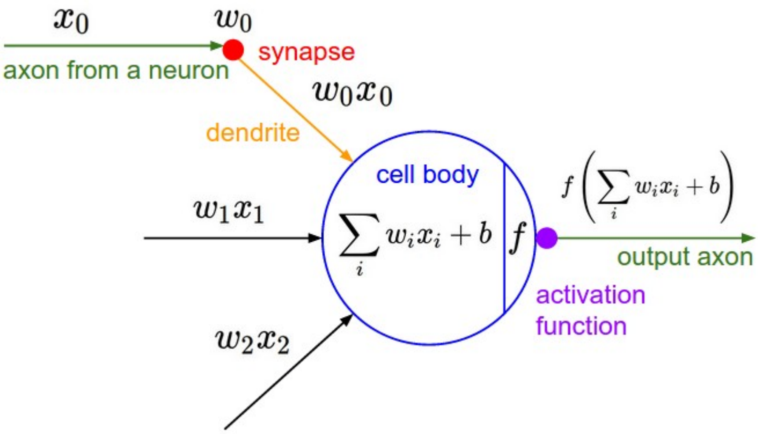


Figure 3.1: An illustration of the typical Neuron model used in fully connected neural networks

recent findings may lead the way for pooling layers to be replaced by convolutional layers with larger strides.

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.

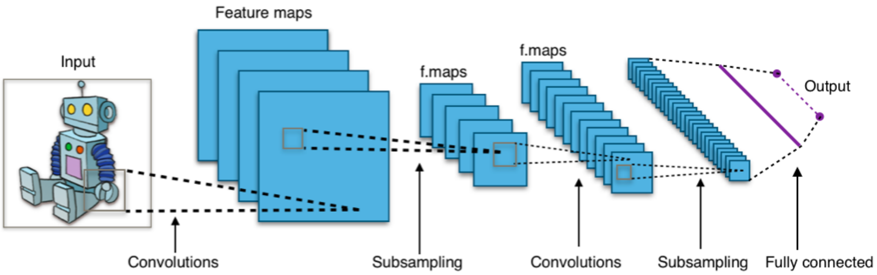


Figure 3.2: A typical CNN structure. Wikimedia Commons

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.

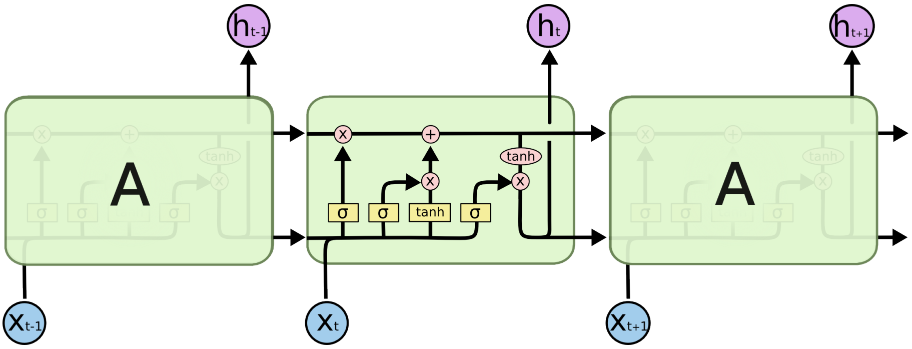


Figure 3.3: The LSTM chain over 3 timesteps, notice the 4 gates affecting the information flow

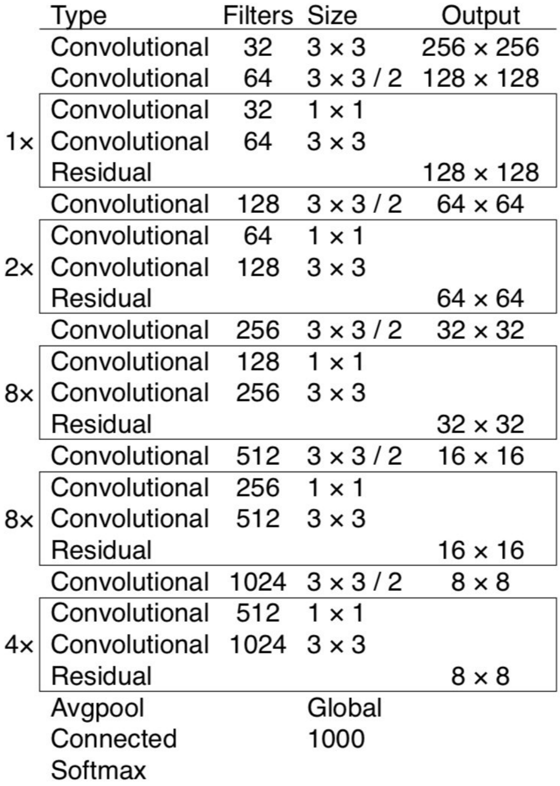
In a regular CNN does not retain any information from previous timesteps, and performs the same computations every time. To be able to utilize the previous data this must be stored and fed to the next computation, or looped in the data structure. This is referred to as a RNN. In contrast to standard RNN the LSTM 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 due to the output of the Sigmoid function σ which ranges from 0 to 1.
The next two layers the Sigmoid “input gate layer” and the $tanh$ 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 retains important information from previous images and ignore information which is not useful due to occlusion for instance. The network can therefore 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). 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 [22]. 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 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 [23][15]. Compared to other high end detectors YOLOv3 is faster and just as robust. It is slightly weaker in adjusting the bounding box to perfectly match the ground truth.



	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

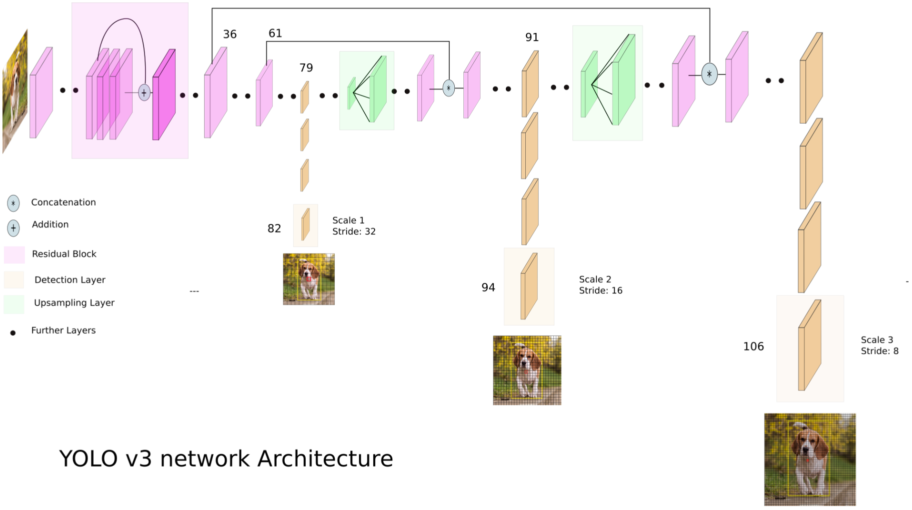


Figure 3.5: The feature extractor of YOLOv3 on the Darknet-53 CNN

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.[14] 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.[24] 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.[14] The tracker is capable of tracking multiple objects simultaneously in real-time.

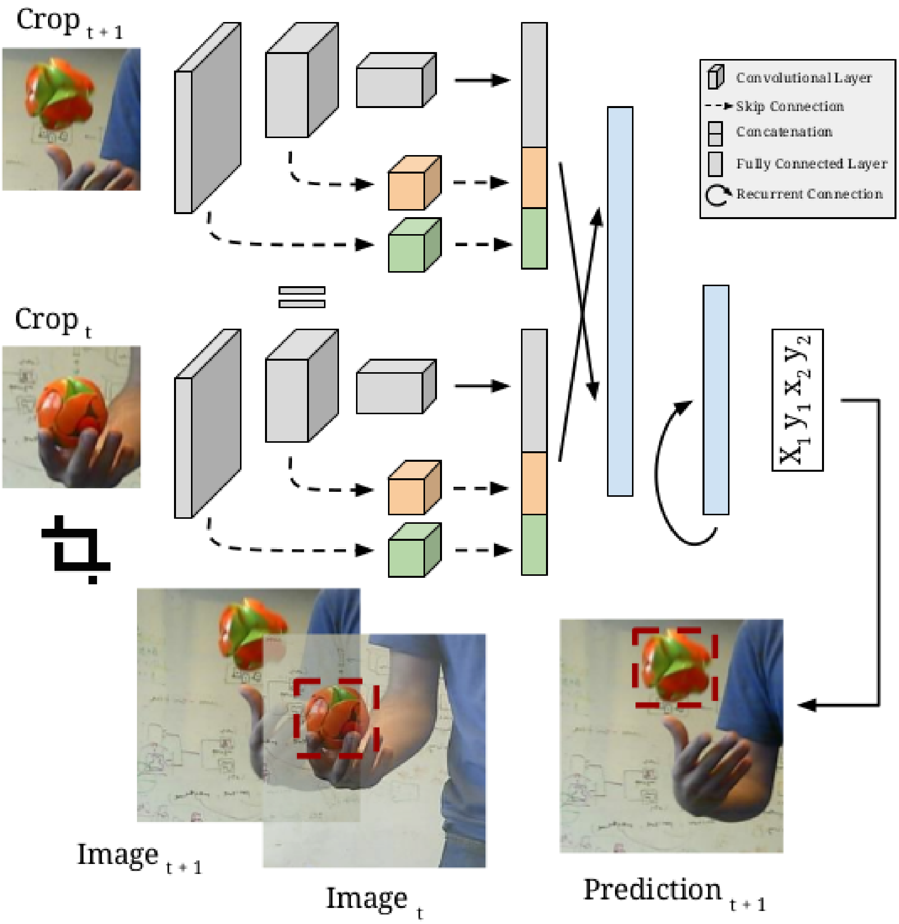


Figure 3.6: Overview of the structure of Re^3

Chapter 4

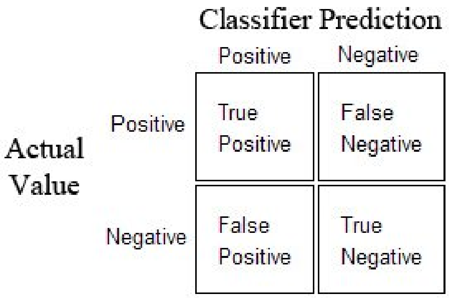
Target Detection and Tracking

In this report we suggest to use a video (series of images) tracker as the basis of fusing 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 report as this is state-of-the-art in terms of speed and robustness as of the time of writing of this report. The Re^3 tracker is used in this report as it is a generic tracker and is demonstrated to be robust and fast [14].

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. 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 use 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). The 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.



		Classifier Prediction	
		Positive	Negative
Actual Value	Positive	True Positive	False Negative
	Negative	False Positive	True Negative

Figure 4.1: Example of the simplest confusion matrix

4.1 Data Association and Sensor Fusion

4.1.1 Data Association

Data association is the process of associating measurements with inherent uncertainty to known prior tracks [25]. The main problems that arise in the data association are whether or not a detection is a true target. For each sensor a twofold problem arises: Is the detection a true object, a True Positive (TP), or is it an error, a False Positive (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 4.1. If the sensors are capable of 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.

4.1.2 The Assignment Problem

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 TP in the further processing. However when multiple objects are being tracked simultaneously it is also important to match which detection belongs to which object. From 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. Another important possibility that the hypotheses must consider is that the detection does not belong to any tracked object and is a FP or a new object.

The Kuhn–Munkres algorithm or ”Hungarian algorithm” is one way to solve the assignment problem [26]. 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 4.3 for an example of the matrix. Other methods include the ”Auction algorithm” [27], successive shortest path algorithms and Djikstra’s algorithm [28].

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 much harder. The Jaccard index, also known as Intersection over Union (IoU), as shown in figure 4.2, 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 [14][15]. The bounding boxes are easy to compare using IoU which directly outputs a measure of overlap between 0 and 1 [29].

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}, \quad \in \mathbb{R} \quad | \quad [0 \leq J(A,B) \leq 1] \tag{4.1}$$

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

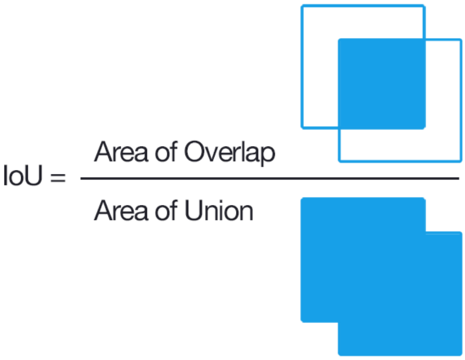


Figure 4.2: Intersection over Union visualized. Wikimedia Commons

See figure 4.3 for an illustration on how the matrix with 4 targets and 4 measurements could look like.

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

Chapter 5

Proposed Tracking Pipeline

The proposed 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 are available. As 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. This would reduce the need for processing the images multiple times, and should give rise to increased detection and tracking speeds. However this would limit some of the module based architecture and would require significant number of hours to implement. This option is not given further attention in the report.

5.1 Autosea Project

There has been built up a framework and a repository for sensor fusion utilizing primarily ROS and Python in the Autosea project. In this repository a fully functional RADAR tracker based on Probabilistic Data Association Filter (PDAF) and Integrated Probabilistic Data Association (IPDA) is available.[30] [31] [32] This has been used during the early experiments where data was gathered also for this project. 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. [33][7] and Wilthil et al. [6]

One important feature from the Autosea RADAR tracker is that detections of land is filtered out using a map database from the Norwegian mapping authorities. The remaining detections are robustly gathered as objects with a center point and a polygon enclosing each object. The data from the RADAR as well as the ASV pose from the INS is output and stored using ROS as different $rostopics$.

All sensory data is time tagged 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 used to synchronize the time information given in the National Marine Electronics Association (NMEA) standard messages such as GPGGA and GPZDA or other time strings. These messages give information about GNSS satellite status from the INS as well as time or other information such as attitude.

5.2 Image Detections using YOLOv3

The YOLO 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 boats at a distance covering as little as 12×40 pixels, or 2% of the width of the raw image. 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 center point plus width and height of the box. The center point, width and height is converted to corners of the bounding box outside the detector. Since the detector and the Re^3 tracker works om the same images no further processing is done using the detector and the bounding boxes that contain boats are handed over to the image tracker. Since the detector processes each image separately and has no recurrent network or memory, the output can change between image frames. This includes the order of detections.

5.3 Image Tracking using Re^3

For each tracked object the image tracker outputs a bounding box array with the 4 corners and a unique name as parameters. The tracker will normally be initialized using the bounding boxes from the YOLO detector. However in future implementations it can also be initialized around the horizon from the RADAR detections. It is assumed that the RADAR-only detections is some distance away since the image detector does not find it. The bounding box would then be initialized to a fixed, rather small, size around the horizon in the direction of the RADAR detection. This could further be evolved into using the polygon from the Autosea tracker as size of the bounding box.

The Re^3 image tracker is primarily written to accept one tracked object. It is however also adapted to accept more tracked objects. These two tracker functions are separated in the code as two objects and you cannot track only one object with the multi-tracker. To overcome this problem we have utilized both trackers and swapped between them when the number of targets changed. Another possibility is to just always track a virtual target so that the number is never less than 2. The tracker is not started if it is only the virtual target present.

5.4 Proposed Sensor Fusion Pipeline

In figure 5.1 an overview of the proposed method is shown. The Autosea RADAR tracker remains to be implemented and fused in the pipeline.

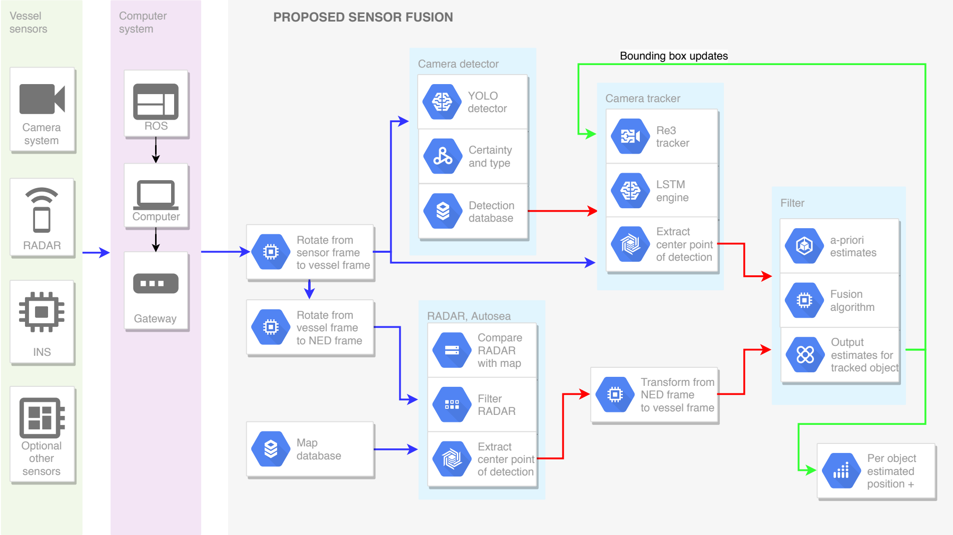


Figure 5.1: Proposed Sensor Fusion pipeline

The Ladybug camera system is interfaced into ROS using a library from Autoware and the 6 images are easily extracted from the ROS node. The raw images output from the Ladybug camera are uncalibrated and must be calibrated using the Ladybug API or using parameters found during prior calibration with OpenCV etc. [19].

The YOLO detector is using the exact same images as the Re^3 tracker. The detector does however only perform detections on every fift imageframe. New detections from the detector will be used to update the tracker if the bounding box from the detector and the tracker deviates too much. This can be calculated using IoU [29], which gives a measure for how good the two bounding boxes aligns. A value below approximately 0.7 would be reason to update the tracker bounding box. However this value should be tuned during experiments and it might also not be a good idea to update the value based on just a single detection from YOLO as this might be erroneous. A special case is where the IoU is zero as this indicates no overlap. This might indicate a new object to be tracked.

If the Re^3 tracker does not receive any validating detections from the YOLO detector, or from other sensors, this is a strong indication that the tracker is not tracking the correct object. This bounding box will then be flagged as invalid and be deleted within a specified time if it is still not validated by any measurements. This time must be experimentally tuned.

Since the problem can involve many different objects that needs to be tracked and detected simultaneously, the chosen method must be very robust and not be to computationally expensive. The data association problem when tracking multiple objects is very challenging. The Kuhn–Munkres algorithm or ”Hungarian algorithm” is one way to assign which track belongs to which measurement. [26] This algorithm is used in the method to assign detections from YOLO to tracks in Re^3 as the output order from YOLO is not fixed.

For future improvements it is prepared for a IPDA based RADAR tracker from the Autosea project [7] to run in parallel to the video tracker and detector. The interfacing of the RADAR detections from the Autosea library and the complementary sensor fusion pipeline is still to be completed. The same is true for utilizing the INS data for position and attitude of the ASV. This data has to be extracted from the Autosea ROS framework where it is stored or output as several $rostopics$. The RADAR detections are presented in a local NED coordinate frame where the Munkholmen island is origin. The position of the ASV is given in the same coordinate frame from the Autosea library.

The detections from the Autosea RADAR tracker are verified to not be land, and is therefore likely to be boats/ships or some other marine obstacles. If the YOLO detector do not find the object that the RADAR tracks, it is proposed that the Re^3 tracker is anyway initialized on this object with a bounding box around the horizon. After initialization the video tracker and Autosea RADAR tracker should track the same object. Since the object is far away and occupy only a few pixels in the image, the Re^3 tracker might have problems locking on to the object, especially if there is much motion from waves etc. Any bounding boxes from these detections that drift away from the horizon should therefore be removed or reinitialized.

When the full fusion algorithm as shown in figure 5.1 is implemented, it will provide substantially higher update rates for position estimates of the tracked vessels.

5.5 Transformation

Since the best available RADAR data is given in NED while the camera data is related to the vessel BODY frame, the RADAR data and the camera must be mapped together in either BODY or NED coordinates for fusion with the other sensors. Since the camera embeds no depth/distance information and the RADAR data contains no elevation information this can be done easily by mapping all data to polar coordinates. From Cartesian coordinates as the RADAR is given in the mapping is:

$$p(t)_{radar} = \begin{bmatrix} \phi(t) \\ r(t) \end{bmatrix} = \begin{bmatrix} atan2(\Delta y(t), \Delta x(t)) \\ \sqrt{\Delta x(t)^2 + \Delta y(t)^2} \end{bmatrix} \tag{5.1}$$

where Δx, Δy is the relative position between the ASV and the other vessel. ϕ is the angle and r is the distance between the ASV and the other vessel. The $atan2(y, x)$ function gives the angle in radians between $-\pi$ and π. The angles can then be rotated between NED and BODY for aligning the sensors. The RADAR data is presented in two ways with a centroid of the detected vessel/object as well as points of the corresponding polygon. The centroid is commonly used to represent the RADAR detection.

As the RADAR gives no information of altitude or height of the detected object we have taken a 2 Dimensional (2D) approach in the sensor fusion method, and not tried to estimate this from the image data either. This simplification allows the image data to be easily transformed to polar coordinates as well. Each camera has an angle of view which is fixed, see equation 2.3 and 2.4. From the camera calibration this angle is found together with projection errors. Once the camera is calibrated it is further simplified and considered to be a linear relation between image coordinates and angle.

$$p(t)_{camera} = \begin{bmatrix} \phi(t) \end{bmatrix} = \begin{bmatrix} \phi_{min} + \frac{(\phi_{max} - \phi_{min})u(t)}{u_{max}} \end{bmatrix} \tag{5.2}$$

Where u is the calibrated pixel coordinate in width. The bounding box coordinates containing a vessel being tracked by the image tracker are transformed to polar coordinates in the vessel BODY frame using 5.2. The distance to the objects are only given by the RADAR data. Between RADAR detections it is assumed that the vessel in question has constant speed and constant heading. The accuracy of the INS is assumed to be perfect. Any error in the GNSS position due to loss of Real-Time Kinematics (RTK) or other disturbances only affects the comparison of the RADAR data and the map. Since all sensors are fixed to the vessel frame and calculations are based on the ASV as origin, any error in initial position does not propagate further.

Chapter 6

Tests and Results

The proposed task asked for experiments with time synchronized capture of camera data together with RADAR and INS data. The experiment was performed in the last week of september 2018 together with several PhD students at NTNU using Maritime Robotics ASV ready boat ”Telemetron”. One ASV from Kongsberg, ”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.

During the experiments the targets 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. [7] for more details about the experiments.

6.1 Interfacing the Ladybug

6.1.1 Time-Synchronization

The first task that was completed was to get the time synchronization up and running. The Ladybug 5+ camera has an auxiliary port used for power and interfacing with the camera. Data is output over a shielded Universal Serial Bus (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 NMEA sentences as well as a dedicated signal wire for 1PPS. The usage of time strings together with 1PPS is common

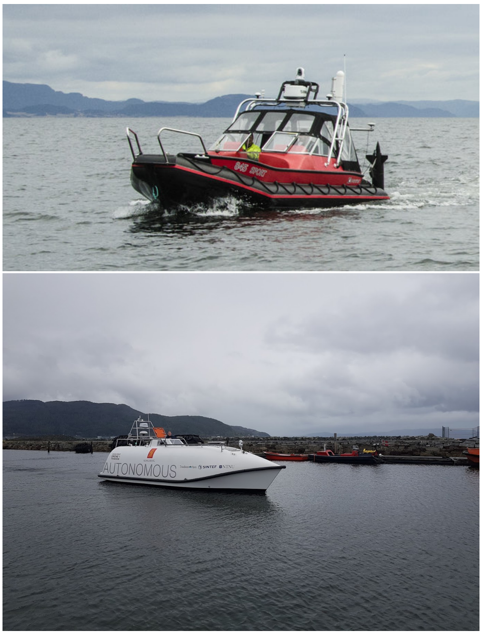


Figure 6.1: ”Telemetron” and ”Drone 1” used in the experiments

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.

6.1.2 Ladybug ROS node

There exists a ROS package for the Ladybug camera system from another open-source project called Autoware [34]. This relies on the Ladybug Linux API 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 of the problem, and this remains an open issue. The driver has been successfully installed on a Laptop later on, and the ROS package verified to work well. The images are output as one

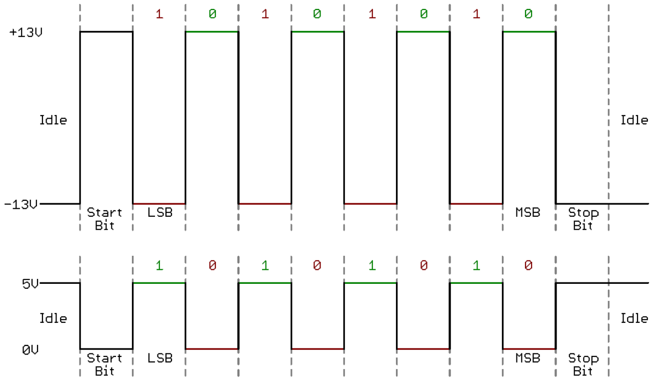


Figure 6.2: Comparison between RS-232 (top) and TTL voltage levels for logic

node with 6 separate ROS topics, one for each of the 6 cameras.

During the experiments the data from the Ladybug had to be recorded on a Windows PC using the supplied LadybugCapPro software.

6.2 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. It has proved to be impressive in locking on to boats with large changes in visibility and appearance. The bounding box is adapting to the visible part of the boat. See the images in figures 6.3 and 6.4 for examples of the tracking performance. What can be noticed is that the tracker locks on to the boat, but it is conservative in the size of the bounding box, only tracking a small part of the boat as it passes by the camera. Last it locks on to an islet as well, as the boat passes this. Due to the recurrent network the islet is not further tracked when it is out of the picture and the bounding box reduces towards the boat.

In figure 6.4 ”Telemetron” does rather aggressive maneuvers without the tracker loosing track of the drone. The drone is quite far away as it is not very visible in the image. Still the tracker performs well. The exact distance is yet to be found from the RADAR data, but the bounding box robustly locks on to the ”Drone 1” target when it is as little as 10×30 pixels. See figure 6.4. This might also be further improved by cropping the raw image to a smaller size around the object before being fed to the Re^3 tracker, as the image is downsized within the tracking algorithm.

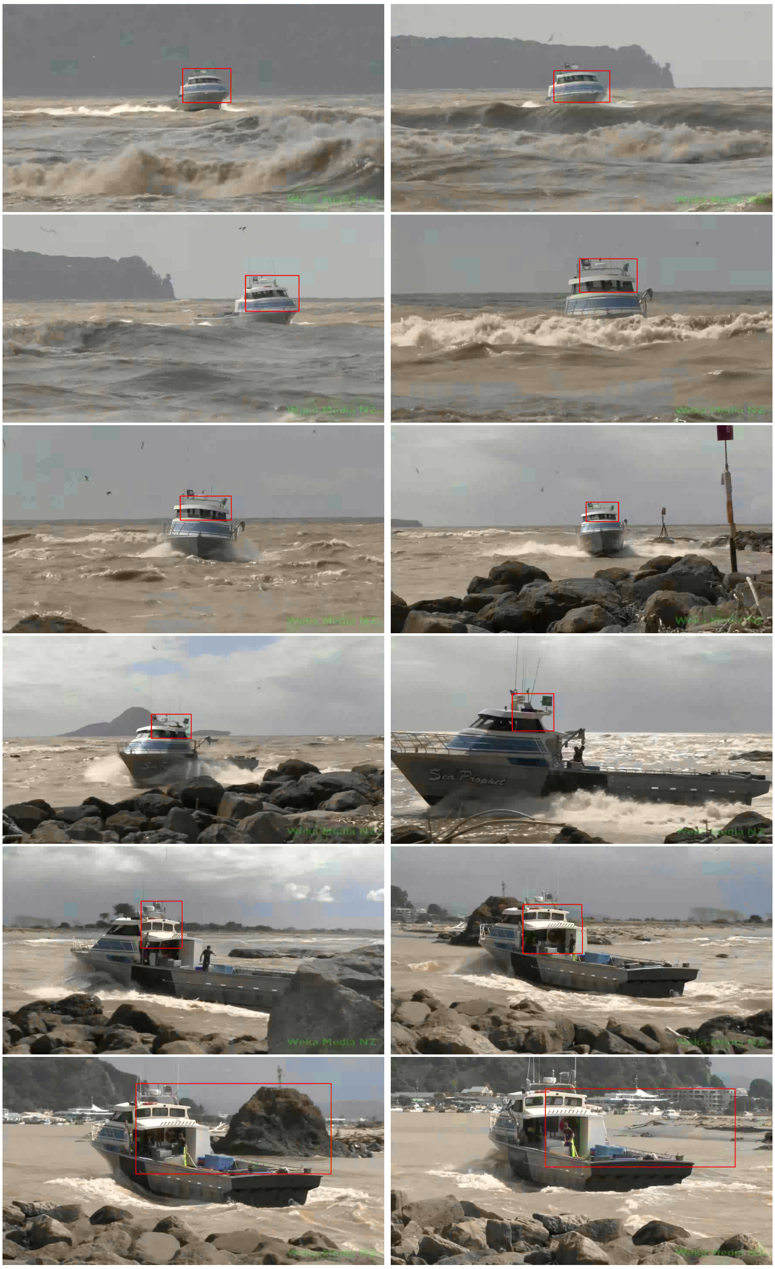


Figure 6.3: Examples of the tracking performance of the Re^3 tracker. The tracker was initialized manually with a bounding box containing the boat. Video from YouTube: Weka Digital Media NZ

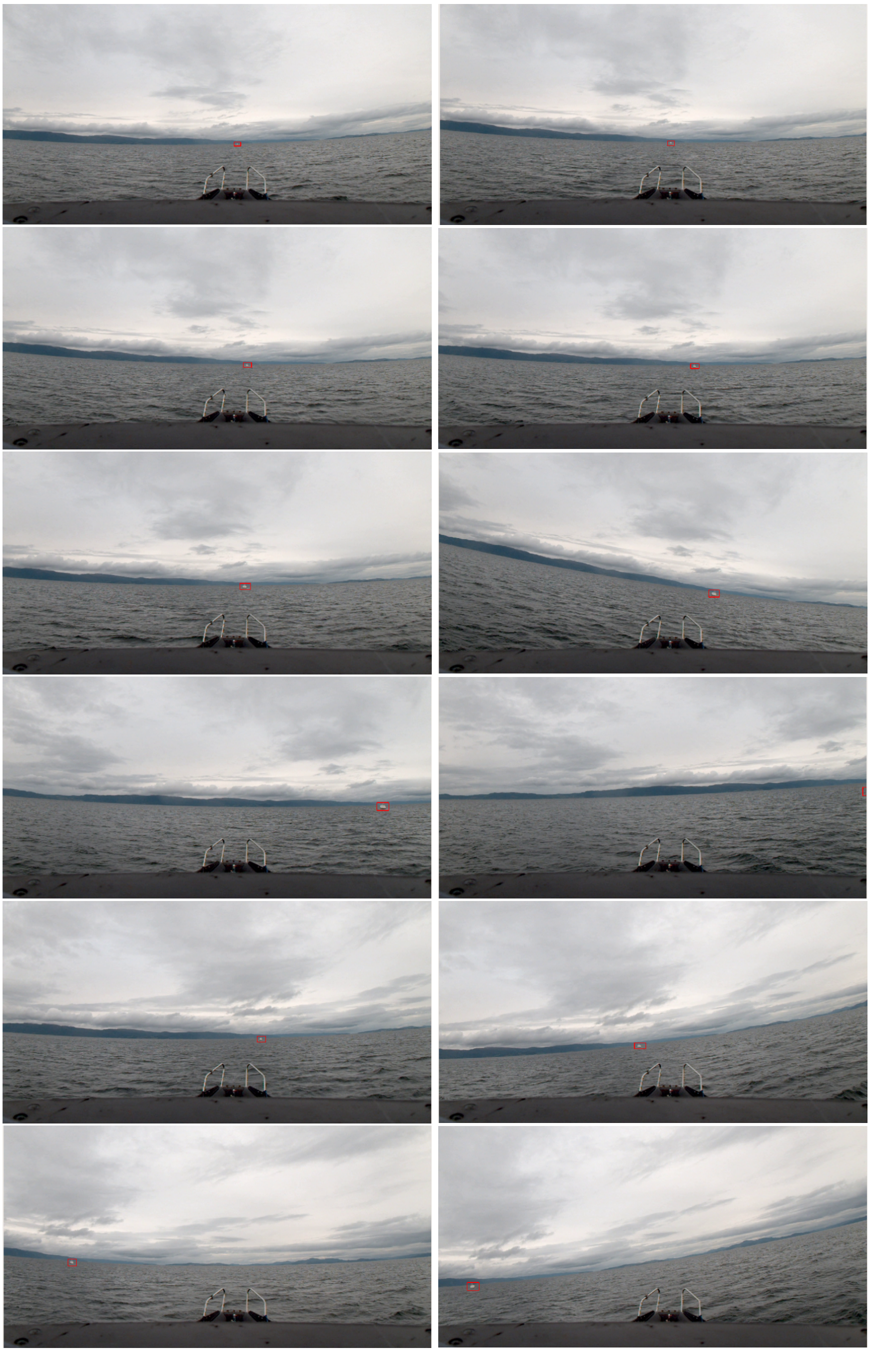


Figure 6.4: Examples of the tracking performance of the Re^3 tracker. The tracker was initialized manually with a bounding box containing the boat. Images from experiment with ”Drone 1”. The images are cropped for clarity

6.3 Verification of the YOLO3 Detector

YOLO comes in many variants with different size, speed and detection capabilities. The 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. The network is trained using the COCO dataset which contains 80 different object classes. The network is therefore not only detecting boats. As can be seen in figure 6.5 the regular YOLO detector is much better than the tiny version.

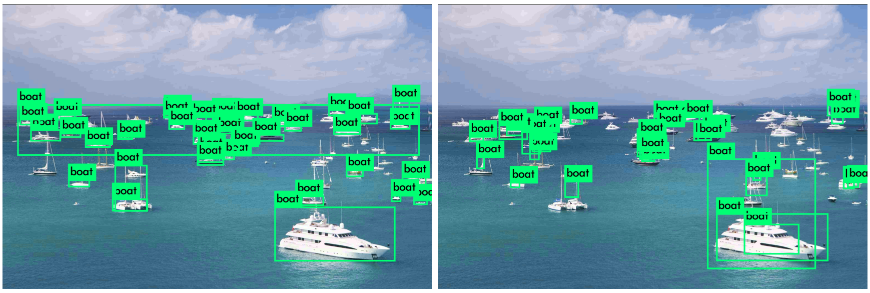
Using the standard net and weights the detector successfully detected boats and ships in a large range of scales with different appearance. It was also able to find multiple boats in the same image. To detect boats as far away as possible the raw image was also cut into tiles 1/6 of the original size. Two images was extracted from the raw image around the horizon and fed through the YOLO network. On the cropped images the YOLOv3 algorithm manages to detect boats that are far away and only occupy 12×40 pixels. That is 2% of the with of the raw image. See figure 6.10.

Without cropping the image the detector is unsuccessful in detecting the boats as can be seen in figure 6.9. 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 [15].

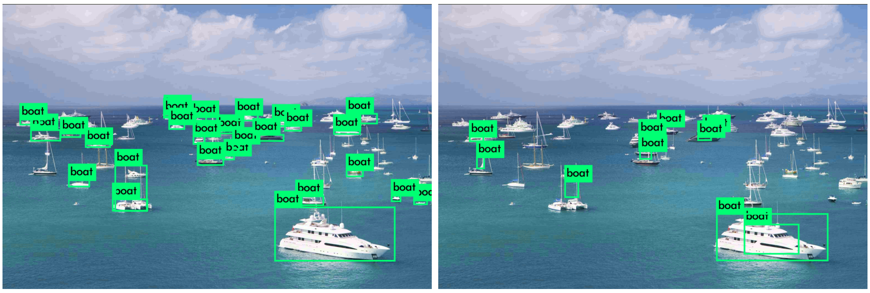
Using the smaller YOLOv3-tiny network the detector only detected the biggest boats and was unsuccessful in finding the more exotic images of boats. The tiny version of the detector is significantly faster, but the lower robustness limits the practical use.

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.

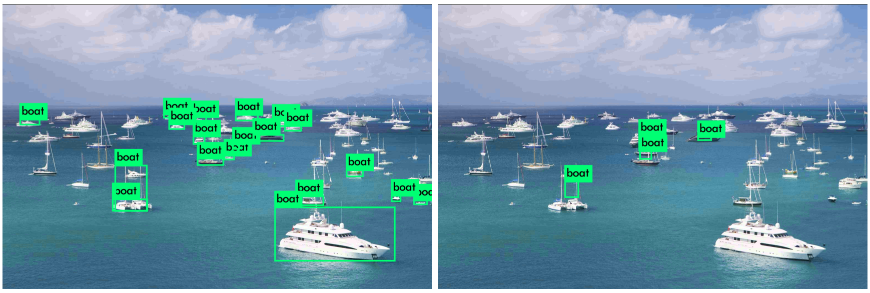
In figures 6.6, 6.7 and 6.8 the performance of the full size YOLOv3 and YOLOv3-tiny are evaluated on the same video as the Re^3 tracker. It is clear that the robustness of the tiny detector is severely lower than the full size detector. The full size detector finds the boats reliably and also correctly detects birds and people. The tiny version incorrectly detects rocks as elephants, and cows and is much less robust in detecting the boats.



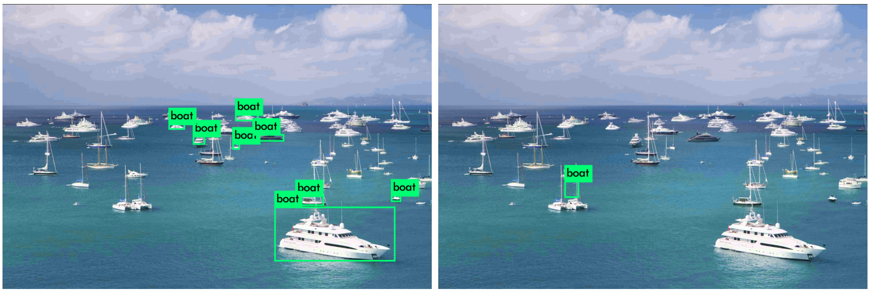
(a) YOLOv3 Accepted certainty: 0.15 **(b)** YOLOv3-tiny Accepted certainty: 0.1



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

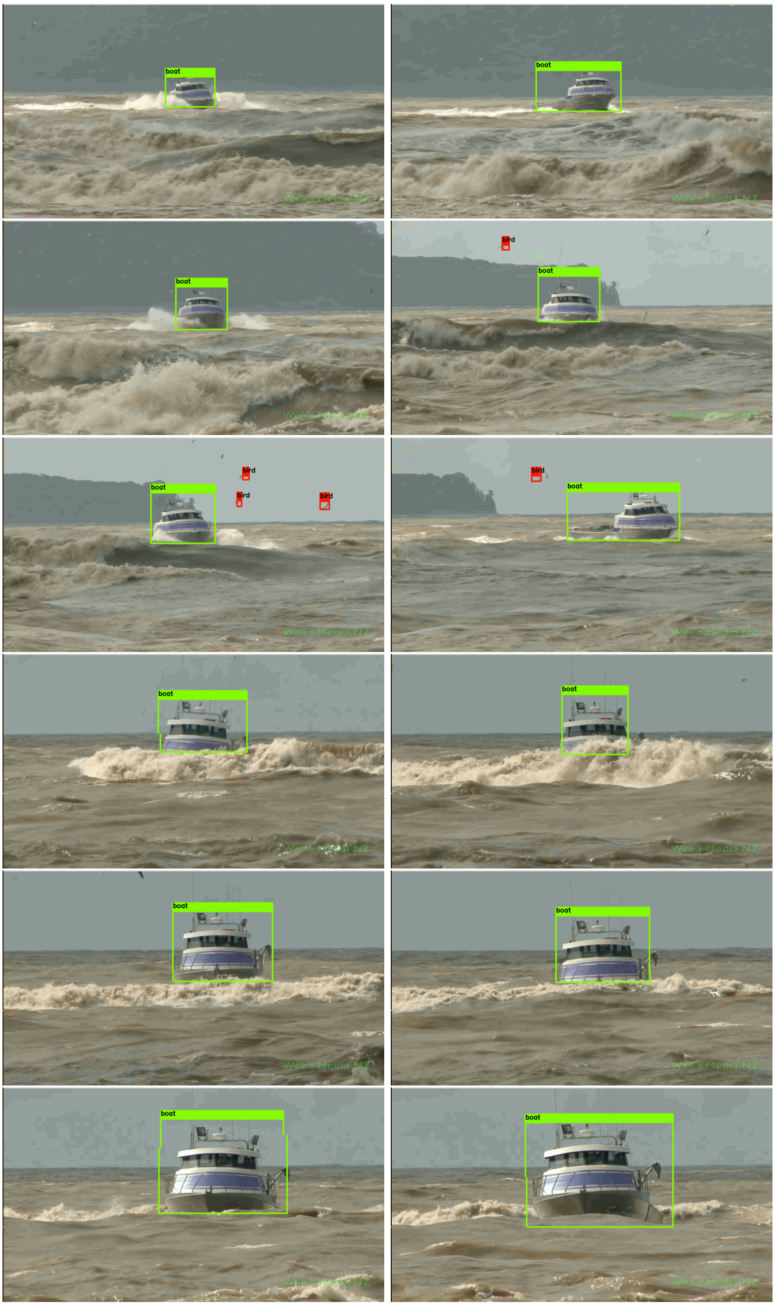


Figure 6.6: Examples of the performance of the YOLOv3 detector. Part 1 of 2. Video from YouTube: Weka Digital Media NZ

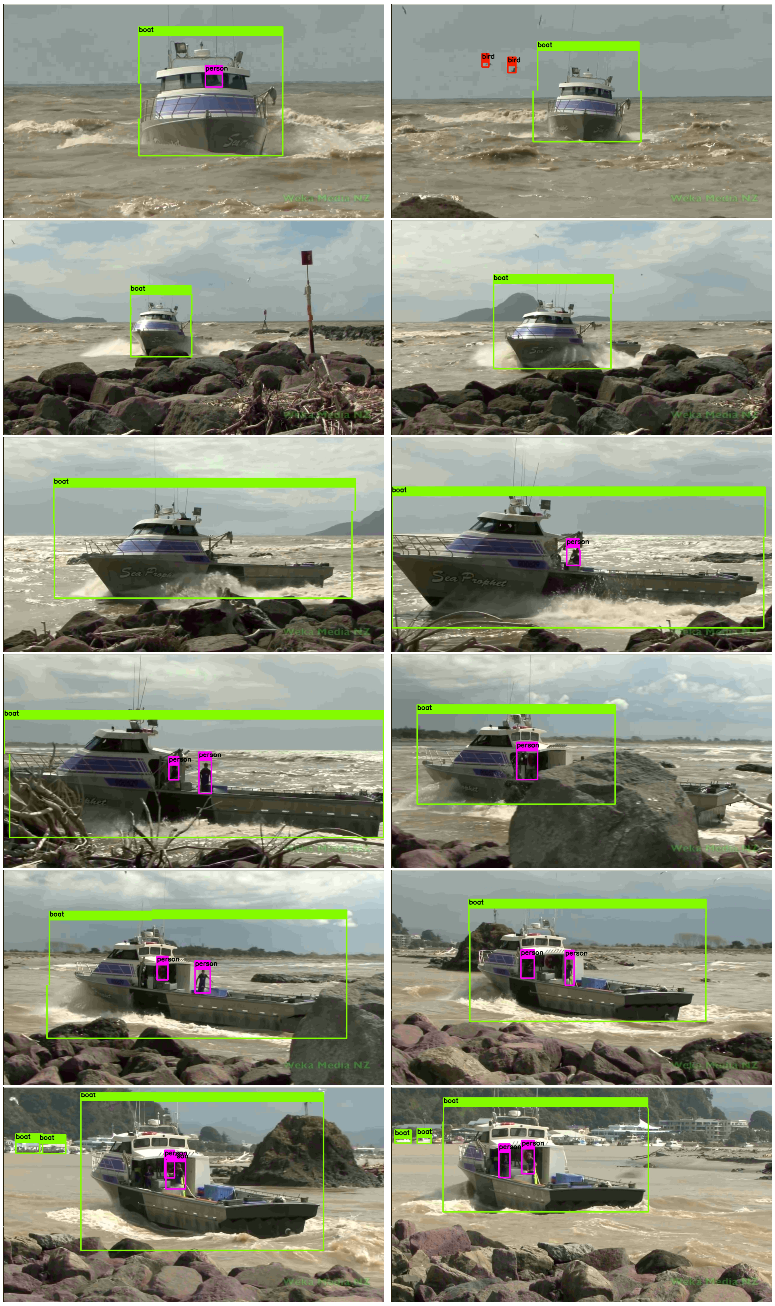


Figure 6.7: Examples of the performance of the YOLOv3 detector. Part 2 of 2. Video from YouTube: Weka Digital Media NZ

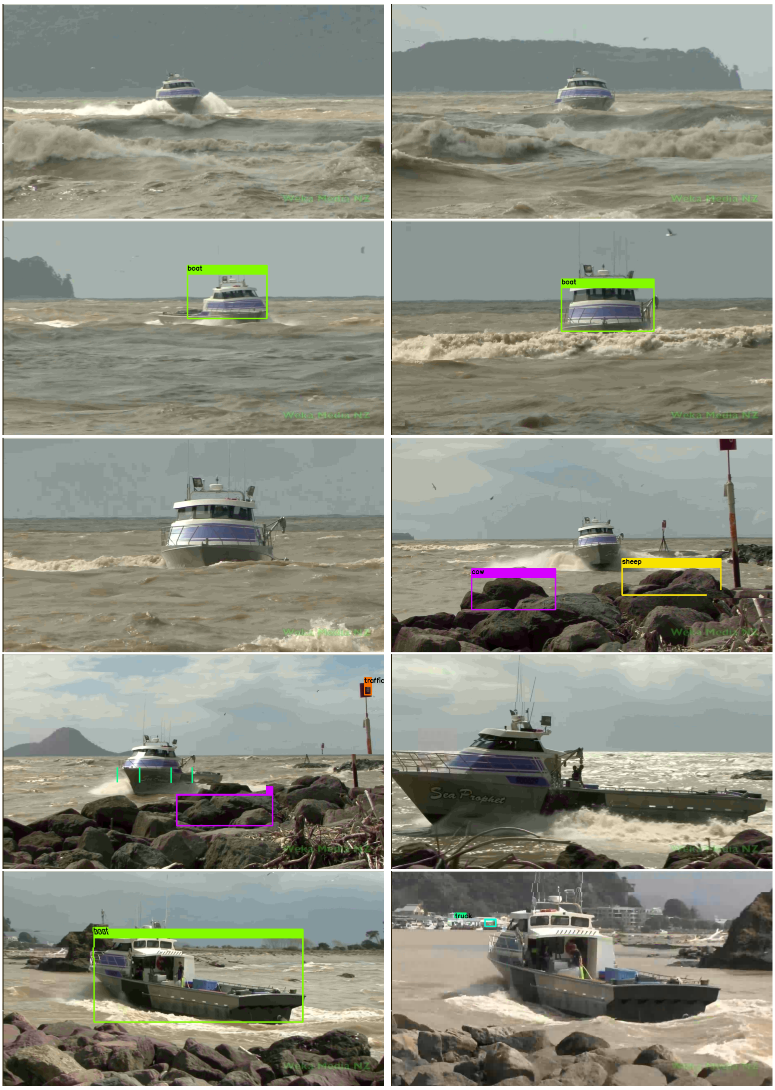


Figure 6.8: Examples of the performance of the YOLOv3-tiny detector. Video from YouTube: Weka Digital Media NZ

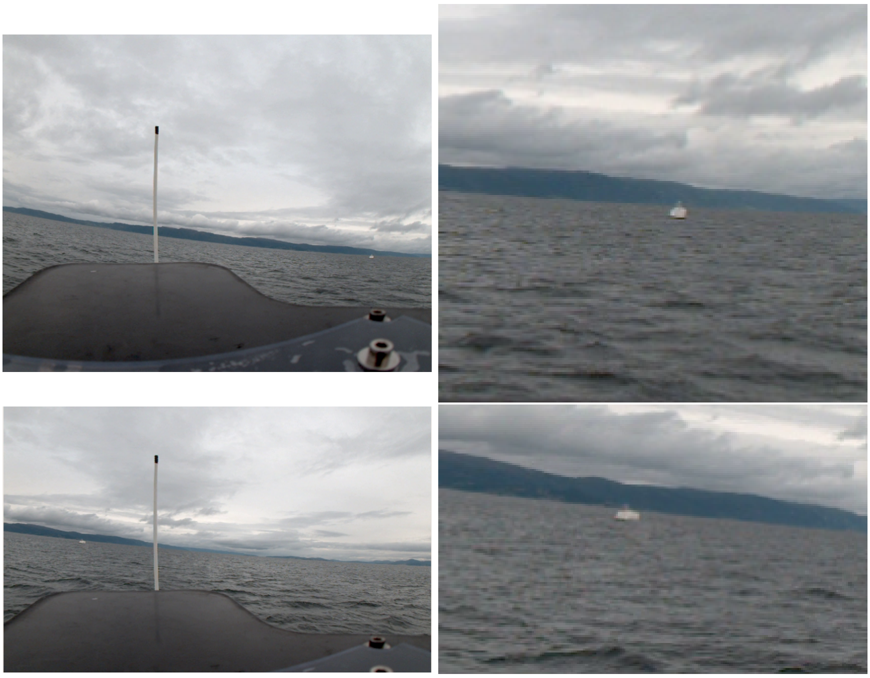


Figure 6.9: Examples of the performance of the YOLOv3 detector on the raw Ladybug images. The target vessel ”Drone 1” is too far away and too small to be detected

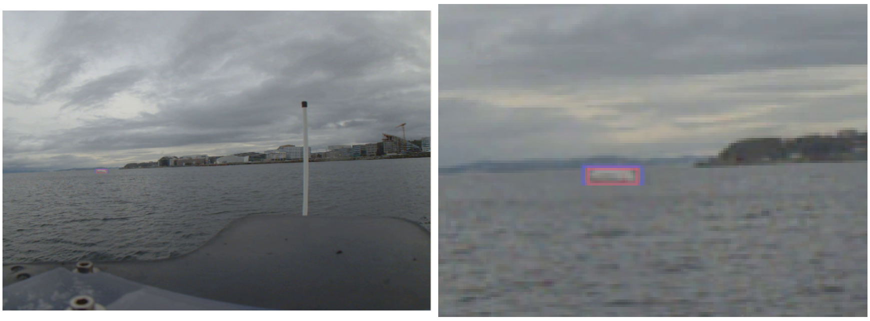


Figure 6.10: By tiling the original raw Ladybug image in such a way that we have two images along the horizon, and 6 tiles in total, the YOLOv3 detector is able to detect the target vessel even at a large distance.

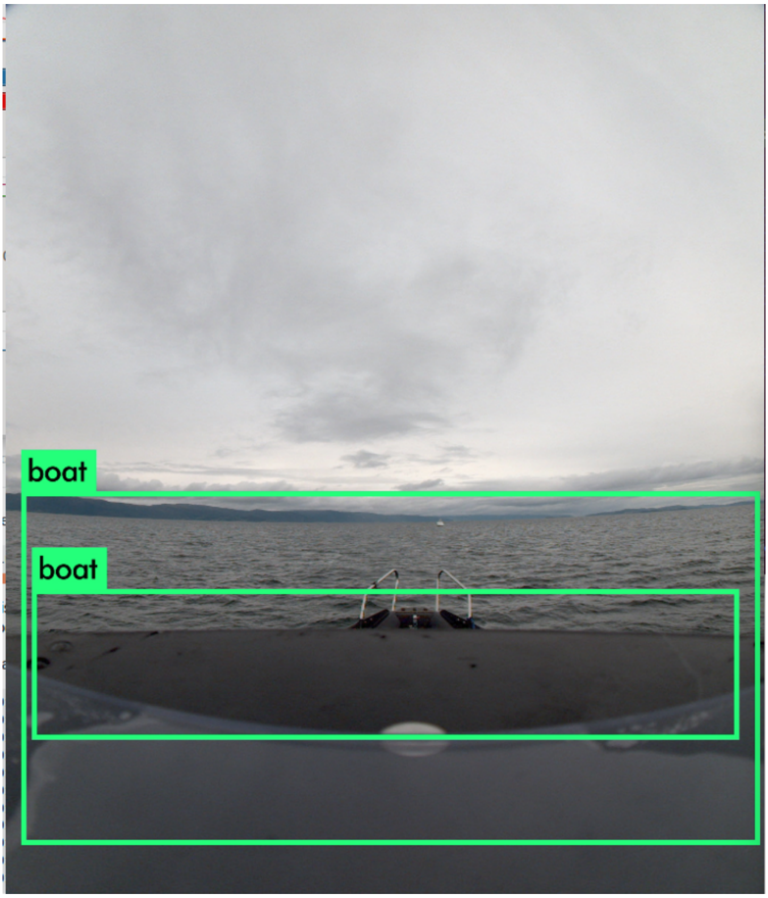


Figure 6.11: The YOLOv3-tiny detector also consistently detected the plate holding the Ladybug camera as a boat. This is an unwanted detection that creates noise in the data.

6.4 Combination of YOLO Detector and Re^3 Tracker

The combination of YOLO and Re^3 shows good results. Due to limited GPU memory on the used computer 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 figure 6.8 it provides updated bounding boxes that is robust enough for the tracker. To speed things up the YOLO network and Re^3 network was initialized only once and later used to only detect and track objects in new images.

If the full size YOLOv3 detector could have been used the bounding boxes for the boat would have been even more accurate.

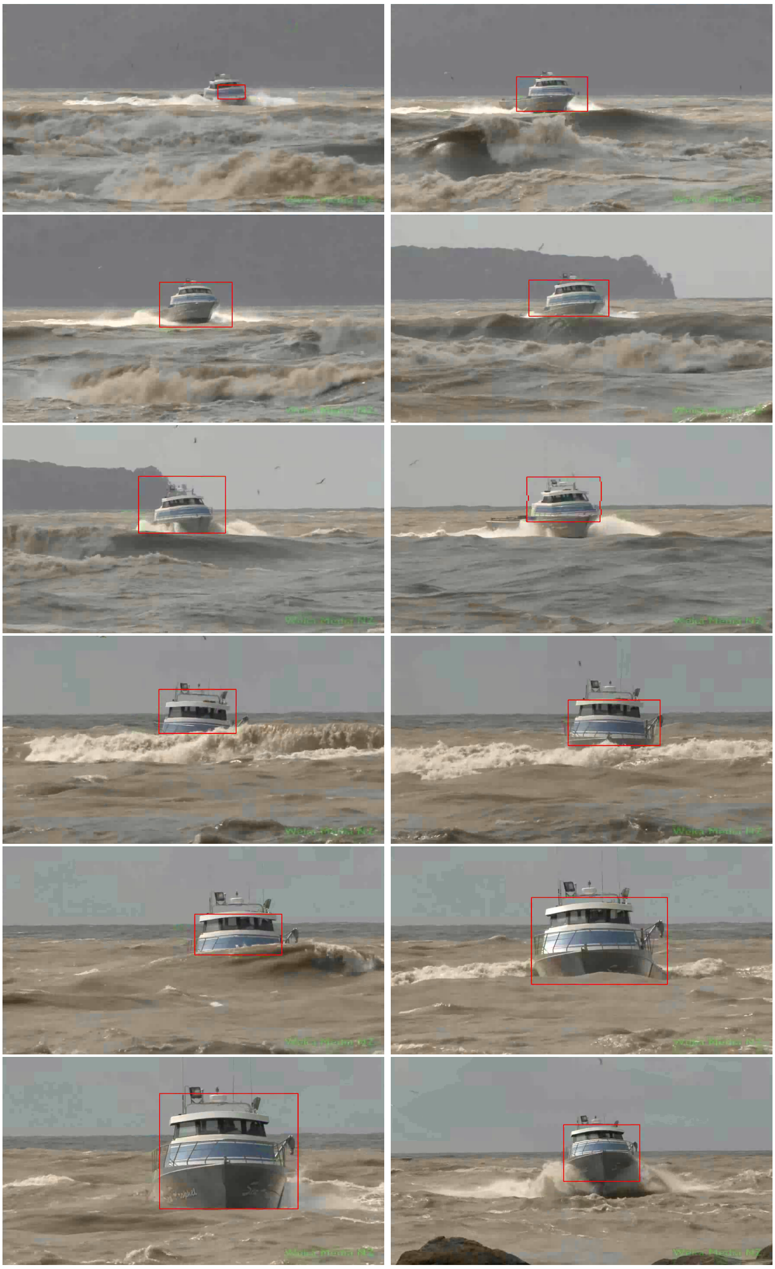


Figure 6.12: Examples of the performance of the combined YOLOv3-tiny detector and the Re^3 tracker. Part 1 of 2. Notice that it is the ”bad” tiny version of YOLO that is used. Video from YouTube: Weka Digital Media NZ

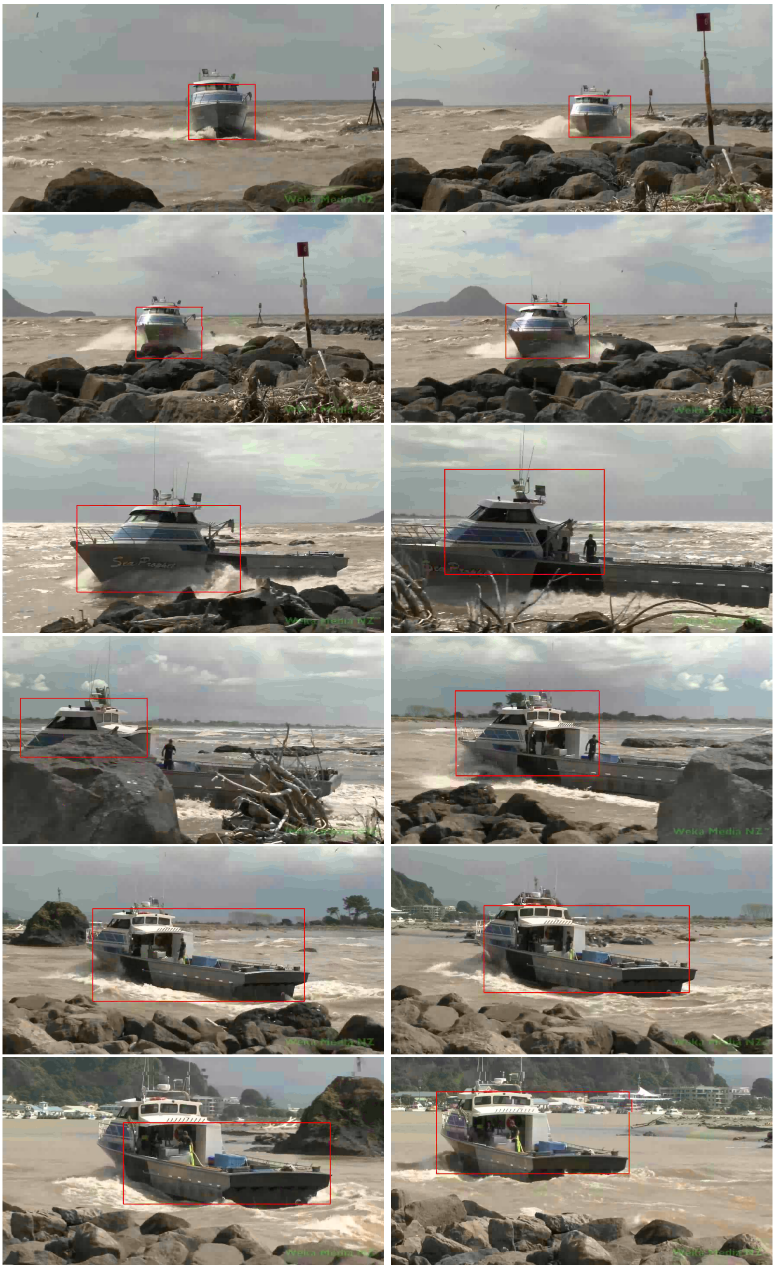


Figure 6.13: Examples of the performance of the combined YOLOv3-tiny detector and the Re^3 tracker. Part 2 of 2. Notice that it is the ”bad” tiny version of YOLO that is used. Video from YouTube: Weka Digital Media NZ

6.5 Single Object Tracking

Single object tracking is quite straight forward with the setup where the YOLO detector initializes and updates the Re^3 tracker. Problems arise quickly if the detector senses more than one object, resulting in the tracker being reinitialized on new objects every time another object is first in the array of detections from the YOLO detector. The problem could have been reduced by not updating the image tracker until several detections match on a new object only.

6.6 Multi Object Tracking

The multi-object tracking problem is a non-trivial part of the sensor fusion. In the fusion of the YOLO detector and the Re^3 tracker the number of initialized tracks grows continuously if the new detection are not matched with the existing tracks. The Kuhn-Munkres algorithm [26] is used to match the detections with existing tracks. If more detections than tracks are found, a new track is initialized for the new detection bounding box. If the IoU between the detections and the existing tracks are below a given value the tracker is updated with the new detection. Otherwise the tracker is left to continue tracking. If the IoU is zero it indicates that the detection is new. It could also indicate that the tracker has lost the desired object and the track will be terminated after a given time.

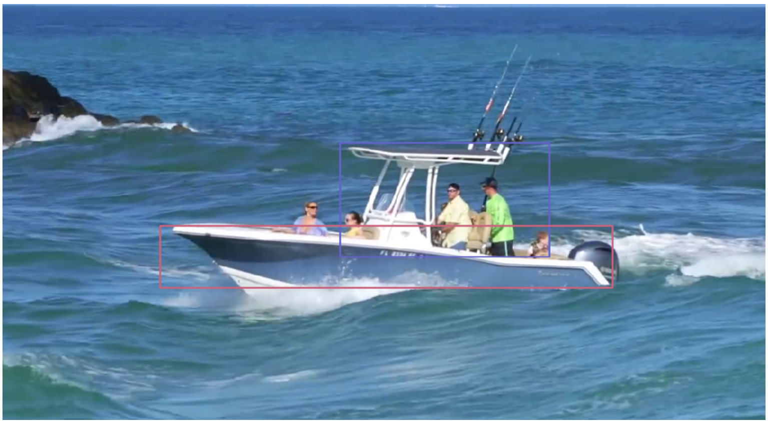
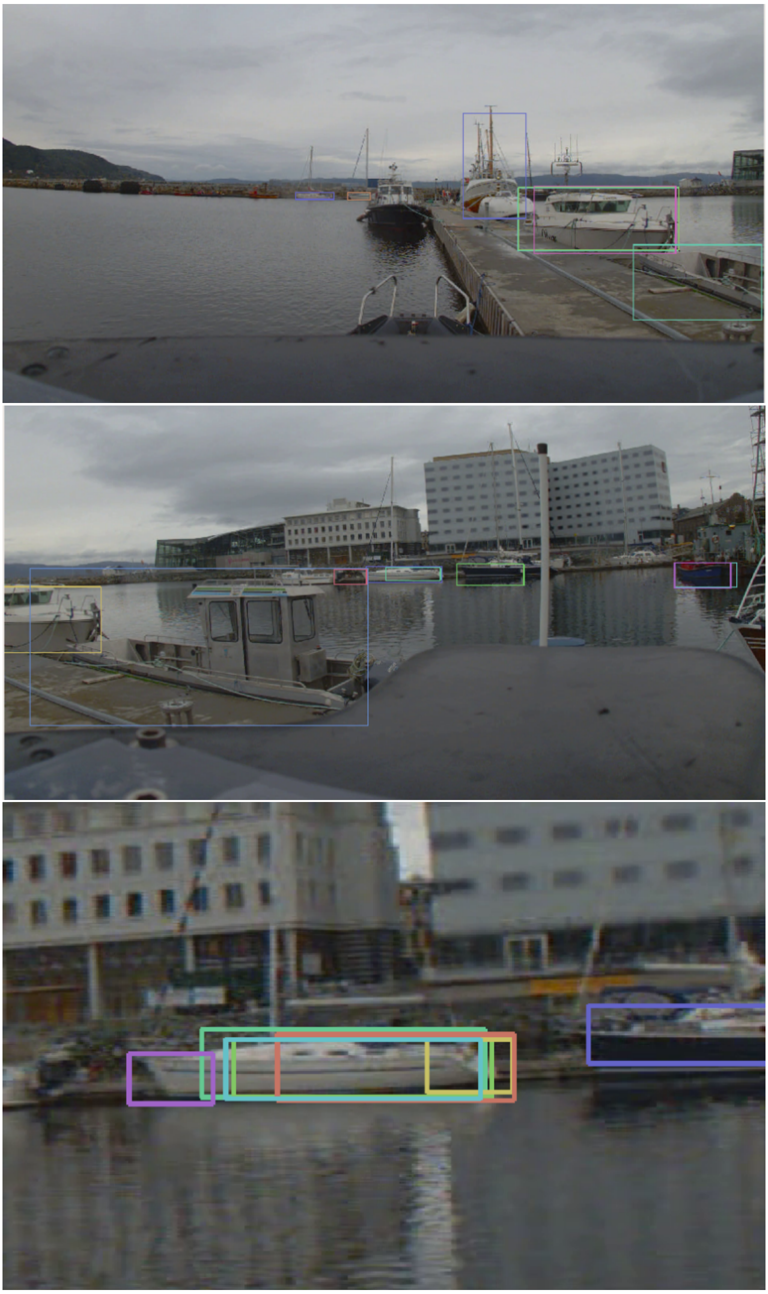


Figure 6.14: If the chosen IoU value for updating the tracker is chosen too low one might end up with overlapping bounding boxes on the same object. The tiny detector had trouble locking onto the whole boat.



(a) If the detections and existing tracks are not combined in the data association, one ends up with several overlapping tracks.

Figure 6.15: 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.

6.7 Discussion

A challenging part that came a bit unforeseen, was the large number of prerequisites that was needed to make the network run on a GPU with the required architecture and performance. Initially the tests had been performed using a laptop with only an integrated Intel GPU, see table 6.1, 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 6.2, which had quite good specifications even though it was about 4 years old and in a bad shape physically. This laptop has 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.

The idea with starting by getting the YOLO detector up to speed first was in itself good, but it was made clear, after a while, that Tensorflow, which is used as one of the backbones of the Re^3 tracker, did not support the latest version of CUDA and after trying to compile Tensorflow from source, and failing, it was back to start. After around 30 hours in total trying different versions of CUDA with its tools, Tensorflow and OpenCV it all started working together. OpenCV was in the end built from source. Here it was also an compatibility issue, where newer versions than 3.4.0 did not work.

As can be seen from table 6.1 and 6.2 there is an 540 fold increase in the speed of the YOLO detector when using the GTX 860M GPU. The Re^3 tracker has a more conservative increase, but the 7-8 fold increase in speed is still massive. Without CUDA the tracker on the new machine ran with 9 frames per second, which is similar to the detector with CUDA.

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 4 GB of GPU memory (RAM), 3 GB might work, but it will be problematic when tracking more than one camera simultaneously. The 2 GB GPU memory on the 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 figures 6.12 and 6.13.

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

Table 6.1: 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
YOLO:	9 frames per second
Re3:	60 frames per second

Table 6.2: Laptop used for further experiments

Chapter 7

Concluding Remarks and Further Work

This project has matured a lot while it has been in the making. The dual deep learning based pipeline that is proposed in this report shows good potential as a robust tracking method for detected boats. The fusion of the YOLO detector and the Re^3 tracker works well for single object tracking, while multi-object tracking still is not optimized. The program parameters must be tuned so that old tracks are pruned fast enough, but not too aggressively. The overall method works well and lays a good foundation to build upon.

Even though the relatively poor performing YOLOv3-tiny version worked well for updating the Re^3 tracker it is suggested to use the full size YOLOv3 for hardware that is capable of this. The much more reliable detections as well as detections of smaller or further away objects is important. To ease the demand for computing power, the detector could be used less frequently as long as it is reliable and consistent with the detection bounding boxes. The Re^3 tracker runs continuously in real-time.

The deep learning methods also have good potential to be further optimized and trained on the relevant boats and ship types to be encountered in this region. The number of object types to be detected by YOLOv3 can be adjusted to only boats and ships or to include other objects if needed. Training of the YOLO detector and the Re^3 tracker requires a large database of images with ground truth bounding boxes around the boats and ships visible in the images. Previous work has been done on the area and some data should be available. This update of the networks will be performed as soon as possible.

The full fusion algorithm including the IPDA based RADAR tracker [7] as shown in figure 5.1 remains to be implemented. The same is true for utilizing the INS data for position and attitude of the ASV.

Bibliography

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

[2] Veritas, Det Norske, “Ships Diving Support Vessels and Diving Systems,” July 2012.

[3] V. Kamsvåg, “Fusion between camera and lidar for autonomous surface vehicles,” Master’s thesis, NTNU, 2018.

[4] 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.

[5] 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.

[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] 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.

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

[9] SIMRAD, “Broadband 4G™ Radar,” 2017.

[10] D. Scaramuzza, *Omnidirectional Camera*. Springer, 04 2014.

[11] Geyer, Christopher and Pajdla, Tomas and Daniilidis, Kostas, “Short course on omnidirectional vision,” 2003.

[12] 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.

[13] 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.

[14] 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.

[15] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” *arXiv*, 2018.

[16] PointGrey, FLIR, “TAN2012009-Geometric Vision using Ladybug Cameras,” tech. rep., 2017.

[17] PointGrey, FLIR, “Overview of the Ladybug Image Stitching Process,” tech. rep., 2008.

[18] SONY-Semicon, “The Industry’s Smallest Pixel Size Class for Industrial Applications and ITS (Intelligent Traffic Systems) Applications A Variety of Functions,” tech. rep., 2018.

[19] G. Bradski, “The OpenCV Library,” *Dr. Dobb’s Journal of Software Tools*, 2000.

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

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

[22] J. Redmon, “Darknet: Open source neural networks in c.” `http://pjreddie.com/darknet/`, 2013–2016.

[23] 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,” 2014.

[24] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A Large-Scale Hierarchical Image Database,” in *CVPR09*, 2009.

[25] E. Brekke, *Fundamentals of Sensor Fusion*. 08 2017. Unpublished textbook for a future course at NTNU.

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

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

[28] J. B. Orlin and Y. Lee, “QuickMatch: A Very Fast Algorithm for the Assignment Problem,” tech. rep., 1993.

[29] A. Rahman and Y. Wang, “Optimizing Intersection-Over-Union in Deep Neural Networks for Image Segmentation,” tech. rep., 2016.

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

[31] 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.

[32] B. Wilking, S. Reuter, and K. Dietmayer, “Probabilistic data association in information space for generic sensor data fusion,” *International Conference on Information Fusion (FUSION)*, no. 1, pp. 317–323, 2012.

[33] D. K. M. Kufoalor, E. F. Brekke, and T. A. Johansen, “Proactive Collision Avoidance for ASVs using A Dynamic Reciprocal Velocity Obstacles Method *,” 2018.

[34] 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,” ICCPS ’18, pp. 287–296, IEEE Press, 2018.