FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Multi-Object Tracking with Multimodal Information for Autonomous Surface Vehicles

Diogo Ferreira Duarte

MSC DISSERTATION

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Andry Maykol Pinto

Co-Supervisor: Maria Inês Rodrigues Pereira

February 25, 2022

Multi-Object Tracking with Multimodal Information for Autonomous Surface Vehicles

Diogo Ferreira Duarte

Preparation for the MSc Dissertation

February 25, 2022

Abstract

Recently, research concerning the navigation of Autonomous Surface Vehicles (ASVs) has been increasing. However, a big scale implementation of these vessels is still held back by a plethora of challenges such as multi-object tracking.

Tracking systems are a key technology for many technical applications in areas such as surveillance, medicine, automation, robotics and autonomous driving. Although well studied, object tracking remains a complex problem due to its many influencing factors like the choosing of a model to define objects as landmarks and which parameters should be tracked, as well as the type of data from where information is taken. In order to better understand the possible areas of improvement, a brief study of current multi-object tracking methods and implementations is made. Regarding generic object tracking, the main challenges such as occlusion, identity switches and view-point variation were identified. Furthermore, investigation of strategies currently developed for tracking in maritime scenarios brought attention for several additional challenges that impair vision-based solutions such as water reflection, harsh illumination scenarios. Additionally, the inherent surface vibrations cause sway on the on-board sensors.

This dissertation presents the development of a full-scale detection and tracking model with an image-based CNN object detector trained trough transfer learning and a tracking module with two distinct sub-modules for association and trajectory prediction. The tracking model is designed work on-board the SENSE ASV, being able to detect multiple vessels in the field-of-view of the installed camera and tracing their position over time while resorting to its sensors, namely the camera and LiDAR, for feature extraction.

To train the detector module through transfer learning and for testing and evaluating the developed tracking model, data was collected with the SENSE ASV by sailing through two nearby ports: Leixões and Viana do Castelo and recording video frames through its on-board cameras, along LiDAR. GPS and IMU data. Images were extracted from the collected data, composing a manually annotated dataset with 9 classes of different vessels, along with data from other open-source maritime datasets.

The developed model, while having its limitations while tracking vessels whose extracted features fail to accurately describe the it, was somewhat successful at tracking in scenarios where the tracking module is consistently fed accurate detections. Examples of the tracking performance can be visualised in the video package provided at the end of this document.

Keywords: object tracking, ASV, object detection

Agradecimentos

Neste documento que marca o final de uma longa etapa de formação académica na minha vida, não poderia deixar de dedicar o primeiro parágrafo para agradecer a quem me acompanhou desde o primeiro suspiro. Aos meus pais que tudo fizeram para que nada me faltasse, à minha avó e aos meus padrinhos por sempre me terem enchido da confiança necessária para alcançar os meus objetivos. É à minha família mais próxima a quem devo a maioria do sucesso atingido e foi neles que encontrei sempre a força para enfrentar os meus momentos menos sorridentes.

Aproveito também para valorizar toda a ajuda e aconselhamento prestados pelo meu orientador, Prof. Dr. Andry Maykol Pinto, pelo incentivo e esforço despendido pela minha coorientadora Maria Inês Pereira. Sem esta ajuda nunca teria conseguido superar as adversidades encontradas no decorrer deste projeto. Agradeço também à restante equipa do CRAS, em especial ao Daniel Campos que sempre pôde dedicar parte do seu tempo a ajudar-me com problemas mais técnicos na implementação.

Por último, gostaria de deixar algumas palavras para os meus amigos. Para o Bruno Miranda que estimo como o irmão que nunca tive, para o Pedro Leite e a sua paciência de ouro a escutar todas as minhas reclamações e protestos, para o Daniel Barbosa que, apesar de muitos defeitos, nomeadamente em ser sportinguista, sempre foi alguém com quem pude contar. E todos os amigos que fiz no Porto, em Arcozelo, nos Carvalhos e em Itália, que compõem o meu círculo de amizades que deram o seu contributo para a minha formação pessoal e todas as memórias das nossas convivências que recordo com carinho.

Diogo Ferreira Duarte

Official Aknowledgements

This work was supported in part by the European Regional Development Fund (ERDF) through the Operational Programme for Competitiveness and Internationalization—COMPETE 2020 Programme, in part by the National Funds through the Portuguese Funding Agency, Fundação para a Ciência e a Tecnologia (FCT), under Project POCI-01-0145-FEDER-030010, and in part by the European Union's Horizon 2020—The EU Framework Programme for Research and Innovation 2014–2020, under Grant 871571.

Diogo Ferreira Duarte

"If you cannot understand why someone did something, look at the consequences and infer the motivation."

Jordan Bernt Peterson

Contents

List of Figures

List of Tables

Abbreviations

3D	Three-Dimensional
AIS	Automatic Identification System
ASV	Autonomous Surface Vehicle
BMPVA	Baird Maritime Passenger Vessel Accident
CNN	Convolutional Neural Network
CWNA	Continuous White Noise Acceleration
EKF	Extended Kalman Filter
FAST	Feature Accelerated Segment Test
GM-PHD	Gaussian Mixture Probability Hypothesis Density
GPS	Global Positioning System
IMO	International Maritime Organization
IMU	Inertial Measurement Unit
IR	Infra-Red
IoU	Intersection over Union
LiDAR	Light Detection And Ranging
MOT	Multiple Object Tracking
MSC	Maritime Safety Committee
R-CNN	Region-Based Convolutional Neural Network
RoI	Region of Interest
SORT	Simple Online and Realtime Tracking
SSD	Single Shot MultiBox Detector
SVN	Support-Vector Network
YOLO	You Only Look Once
k-NN	K-Nearest Neighbors
mAP	Mean Average Precision

Chapter 1

Introduction

1.1 Context

Looking at the current state of the road vehicles industry, one can speculate a bright future for autonomous driving. American companies such as Ford, General Motors, Waymo and the chinese Baidu have already tested millions of kilometers on the road[1], while Tesla is already enjoying success in the industry, delivering over a million vehicles with level 2 of autonomous driving[2]. A natural step would be to mirror the success of cars into ships. Considering the possibility of having autonomous vessels fully capable of handling the shipping business worldwide, a quick and logical thinking leads to savings in personnel costs. However, there is a need to make sure it is a viable option regarding regulations, in order to motivate investment and technological advance in the field.

A manned ship will always have a chance of sinking or simply having an accident due to human failure, while an unmanned one can ideally be unsinkable under normal conditions. Recent studies show grounding, collision, negligence and human error among the top five reasons why ships sink[3]. Figure 1.1 illustrates the main causes of fatal passenger vessel accidents. While the monitoring of ASVs still requires a human, the use of autonomous ships would have a direct impact on the presented statistics, decreasing errors attributed to human failure.

The relatively recent foundering of the Costa Concordia shows that even ships that are considered masterpieces of modern technology can be involved in disastrous accidents. Incidents such as the one referred motivate development of technology in the maritime industry, namely in the development of autonomous vessels. ASVs need regulations to operate safely and, although some regulatory aspects for manned ships may be compatible with unmanned ones, there is a need for specific international regulations taking into account the unique characteristics of unmanned ships. There have been recent efforts into developing such regulations by the International Maritime Organization (IMO) such as the Strategic Plan (2018-2023), which has a key strategic direction to

[1] Intelligent Mobility Xperience – 5 top autonomous vehicle companies to watch in 2020
[2] Number of Tesla vehicles delivered worldwide from 4th quarter 2015 to 3rd quarter 2020
[3] Marine Insight – Why Ships Sink - 10 Major Reasons

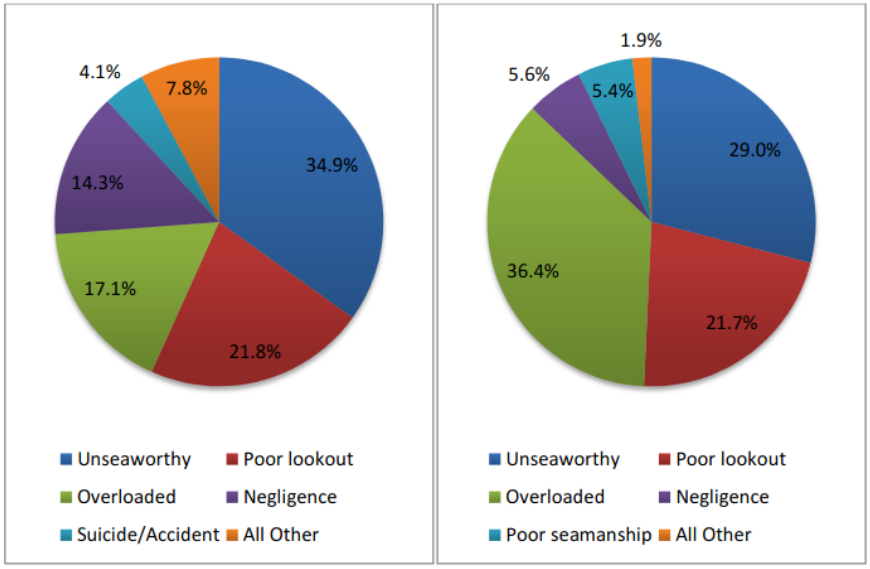


Figure 1.1: Causes of fatal passenger vessel accidents 1966-2015, by number of accidents on the left and by fatalities on the right (BMPVA database) [1].

"*integrate new and advancing technologies in the regulatory framework*" [4]. In 2017, following a proposal by a number of member states, IMO's Maritime Safety Committee (MSC) agreed to include the issue of marine autonomous surface ships on its agenda[5].

Table 1.1: Analysis of a voyage estimate, based on 1980 prices, for a typical 60 000 Dead-weight Tonnage (DWT) modern bulk carrier with a 16 knot service speed [2].

Items	**Percentage of total costs**
Fuel oil	29.6
Lubricating oil	0.9
Engine maintenance	1.2
Hull maintenance	3.9
Crew Costs	**24.2**
General administration	0.4
Insurance	7.6
Capital cost (depreciation, interest charges, etc.)	28.2

From an economic standpoint, research indicates that crew costs represent 24.2% of the costs in ship operations (see table 1.1). The constituents of the voyage estimate can vary by 10% owing to price changes and will differ according to the type and age of the ship and country of registration. For example, a passenger vessel will have a larger crew than a cargo ship, so crew costs will account for a greater proportion of the total [2]. The fact that no crew is required for an ASV brings risks for the ship in case of accident or hardware failure at sea. A good example would be a fire aboard the ship that could not only ruin the cargo but also break the hardware needed for

[4] IMO's Strategic Plan
[5] MSC, 98th session

the ship's navigation or monitoring, without a crew to put out the fire. A possible solution for this issue should incur prevention to avoid and/or mitigate the chances of having such accidents. It is also viable to develop standard procedures for the monitoring crew, in case of an unexpected event, such as allowing remote control to some degree. Crew costs aren't the only expenses that can be reduced with ASVs. Tracking systems often rely on information received from proximity sensors, motion detectors and optical sensors, mapping locations of different objects around the vehicle. Besides automated travelling, benefits such as fuel economy can be achieved by predicting object behaviours and choosing optimal routes to avoid fuel waste with speeding and braking, and insurance costs by reducing the risk of accident. The demand for renewable sources of energy, namely at sea, creates the need for cost-effective solutions capable of handling thorough tasks such as maintenance. Autonomous Vehicles are considered the prone solution to carry out said tasks in these harsh environments [3].

Regarding fuel economy, the National Renewable Energy Laboratory (NREL) partnered with Volvo to measure real world fuel consumption of autonomous vehicles versus manually operated cars[6]. The researchers found using adaptive cruise control resulted in a 5% to 7% drop in fuel consumption. Since cruise control means adapting the vehicle's speed regarding the surrounding objects, similar benefits can be assumed with object tracking inputs. Insurance costs also depend on the vehicle's safety features[7]. A higher level of autonomy in a vehicle translates into more security features which should reduce insurance prices.

1.2 Motivation

An Autonomous Surface Vehicle (ASV) is a vehicle with the ability to move and orient itself in an unknown and possibly hostile environment, with resource to on-board sensors in order to gain understanding of its spatial awareness, while being controlled by algorithms to make all the maneuver operations autonomous, ranging from collision-free navigation to the take-off and docking sequence.

Tracking systems are a key technology for many technical applications in areas such as surveillance, medicine, automation, robotics and autonomous driving. Although well studied, object tracking remains a complex problem due to its many influencing factors like the choosing of a model to define objects as landmarks and which parameters should be tracked, as well as the type of data from where information is taken. In order to better understand the possible areas of improvement, a brief study of current multi-object tracking methods and implementations has been made. The critical aspects to look for when developing such a system are:

- Object characteristics;
- Accurate placement of trackers in objects;
- Object tracking despite distractions such as occlusions and changes in illumination;

[6] Measuring Real-World Fuel Economy in Autonomous Cars
[7] Lower Your Car Insurance Premiums

- Algorithm speed and autonomy.

The tracking challenge differs depending on the environment and the type of objects detected. As such, reliable characterization of the environment obtained in real-time is essential to the execution of many tasks such as mapping and obstacle avoidance. In the context of ASVs, it is important to know what each detected object might be to determine their possible range of movement and to predict where they might be going. Different objects may also constitute different behaviours, so their individual characteristics (such as shape and size) should also be studied. Regarding the second item, the use of an array of different sensors to detect objects accurately in different scenarios is a common approach. Combining such varied information into a precise detector is a challenge that needs addressing to avoid the misinterpretation of data. Additionally, the maritime environment renders a layer of additional challenges such as a possibly clustered scenery and harsh illumination conditions (i.e. water reflection causing mirroring and overexposure effects, fog, lack of artificial illumination for night-time operations). The instability of the surface also causes vibrations, swaying on the on-board sensors. With these conditions, detection and tracking of dim or small targets on the water is very challenging. Figure 1.2 highlights some of these queries.

Foggy scenario.

Low-light scenario.

Overexposed scenario.

Water mirroring scenario.

Figure 1.2: Examples of harsh maritime scenarios that raise difficulty during detection and subsequent tracking.

The complexity of maritime object tracking requires a multi-versed solution capable of answering multiple questions. Which and how many objects need to be simultaneously tracked? Which sensors can be used to collect relevant information for detecting and tracking objects?

How will the collected information be processed? The challenge within the scope of this dissertation is to provide a tracking system capable of aiding the self-navigation of the autonomouS vEssel for multi-domaiN inSpection and maintEnance (SENSE) ASV [4]. Moving objects ranging from small boats to bigger ships, or even buoys, need to be tracked with enough precision to avoid collisions. As the testing scenario is often occupied by dozens of boats, the developed system should run in a machine capable of tracking those which are close enough to threaten a possible collision. Static elements like docks, walls or coast can be avoided with IMU and GPS information that feeds the system with precise location and orientation of the own vessel. Their static nature rejects the demand for complex tracking and, as such, those fall outside the ambit of this work.

1.3 Objectives

The main purpose of this dissertation is to ultimately produce a system able to detect and track objects (i.e. other moving vessels) for an ASV, enabling it to self navigate towards an end-point without accidents. The integration of the results in an ASV is expected to validate the system in a simulated scenario and, later, in a real environment. Information will be gathered from the ASV's different sensors, such as Global Positioning System (GPS), Inertial Measurement Unit (IMU), Light Detection And Ranging (LiDAR) and visual cameras, in order to detect, identify and continuously track the multiple objects resorting to object-oriented programming and machine learning.

Starting from this process of data acquisition, this thesis aims to:

- Collect a testing dataset, with the aid of a 3D simulation environment, based on data captured by the camera and LiDAR incorporated in the ASV. A dataset composed of diverse environments, with different vessel models and structures is ideal for better translation of real-life possible scenarios;

- Develop and implement a detection and tracking system for multiple objects;

- Integrate the developed work in a real ASV with the goal of testing the system in an authentic environment under conditions provided by INESC TEC (Institute for Systems and Computer Engineering, Technology and Science).

1.4 Scientific Contributions

This work originated an article presented at the OCEANS Conference 2021: San Diego – Porto:

- D. F. Duarte, M. I. Pereira and A. M. Pinto, "Multiple Vessel Detection and Tracking in Harsh Maritime Environments," *OCEANS 2021: San Diego – Porto, 2021* [5].

1.5 Document Structure

In addition to this introductory chapter 1, the present document contains four other chapters.

Chapter 2 proffers the research done regarding the state of the art. Section 2.1 includes generic definitions regarding the field of object tracking such as Convolutional Neural Networks, Kalman Filter and the Hungarian Algorithm. Section 2.2 focuses on real-time object detection methods and section 2.3 portrays several state-of-the-art models for multiple object tracking. Section 2.4 focuses on the detection and tracking models regarding ASVs. Concluding this chapter, section 2.5 provides an overview of the most relevant analysed methods and challenges to address in the field.

Chapter 3 provides a more detailed characterization of the problem, presenting a thorough description of the implemented tracking system. Section 3.2 entails the work done in assembling a maritime dataset used for training the detection module and developing the tracking module, while sections 3.3 and 3.4 further describe the development process of the detection and tracking modules respectively. The analysis of the obtained results for each module is detailed in chapter 4.

Finally, chapter 5 provides a conclusion to this work, elaborating on the future work opportunities derived from the developed work and describing some of the challenges along the process.

Chapter 2

State of the Art

The information gathered in the previous chapter demonstrates the convenience of autonomous navigation in the context of surface vehicles. In order to carry out the work in the context of this dissertation, an extensive study of subjects such as Object Detection and Tracking is essential for an assiduous understanding of concepts and current working methods. The following section will describe the results of the research on state-of-the-art models of object tracking.

2.1 Introduction to Object Tracking

Traditionally, object tracking was implemented using continuous object detection, starting with traditional computer vision methods in the late 90s. These approaches utilized classic feature detection algorithms like SIFT [6] and SURF [7] combined with machine learning algorithms like k-Nearest Neighbors (k-NN) [8] or Support-Vector Networks (SVN) [9] for classification. However, in certain conditions where detection would fail for a few frames, the tracked objects would be lost. Detection depends heavily on what the sensors can perceive, but what happens when they are occluded? Some of the major challenges for detection that need to be addressed separately in order to achieve successful tracing are listed below.

- Occlusion - The object in question is partially or completely obstructed.
- Identity Switches - After two objects cross each other, how to know which one is which?
- View-point Variation - An object may look very different when looked at from a different side. Identifying such objects would be very difficult relying solely on visual detection.
- Complex object shapes or motion.
- Scene illumination Variation.

In addition to the classification provided by a detector, a robust tracking module must retain information about each object, such as their location, trajectory prediction and pose estimation, allowing the algorithm to discern similar objects with the same classification in the same frame, and to solve the aforementioned challenges.

Figure 2.1 represents the block diagram of a typical Multiple Object Tracking (MOT) model which includes a tracking module responsible for handling predictions and producing position estimates for occluded objects, segmenting each detected object.

Figure 2.1: Block diagram of a typical Multiple Object Tracking model.

Currently, trackers such as TrackR-CNN [10] and Tracktor++[11] are popular models with state-of-the-art tracking accuracy while models such as ROLO [12] introduce lighter processing methods providing balance between tracking accuracy and execution speed. Furthermore, in the context of ASVs, object detection and tracking have a real-time execution requirement in order to compute the driving instructions and successfully navigate without accidents. The following subsections: 2.1.1 to 2.1.3 will summarize fundamental concepts regarding the basis of detection and tracking architectures. The object detection topic, which in itself is a preliminary stage for object tracking, is covered next, in section 2.2. Lastly, sections 2.3 and 2.4 will resume the methods implemented in the most popular state-of-the-art tracking models, with a hefty focus on real-time solutions.

2.1.1 Convolutional Neural Networks

CNNs are a type of artificial neural networks designed to take advantage of structures in images (or similar content) and 3D data. Their first successful use was to recognise handwritten digits, by LeCun *et al.* in 1990. The network, LeNet [13], consisted of five layers: two convolutional layers followed by a pooling layer and a fully connected layer, as shown in figure 2.2. Convolutional Neural Networks evolved into much more complex natures but the general structure of alternating convolutional and pooling layers, followed by fully connected layers, remains.

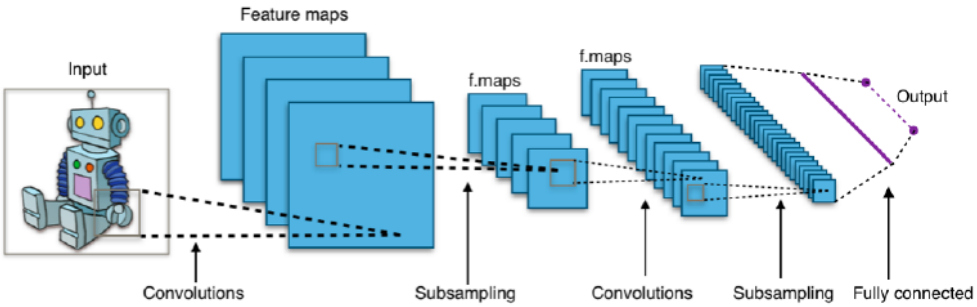


Figure 2.2: Basic structure of a CNN [14].

The convolution layer is the core building block of a CNN, carrying the main portion of the network's computational load. This layer performs a dot product between the set of learnable

parameters (the kernel) and the restricted portion of the receptive field. This produces a two-dimensional representation of the image, the activation map. The pooling layer replaces the output of the network at certain locations by deriving a summary statistic of the nearby outputs. A pooling operation is processed after convolution layers, decreasing the required amount of computation and weights while keeping the object recognizable regardless of where it appears on the frame. Fully connected layers have a weight associated with each pair of input and output. Their output is computed as the inner product of the weights and the input. These weights have to be learned, and since the number of weights is proportional to the number of inputs and outputs, the number of parameters quickly grows.

2.1.2 Kalman Filter

In 1960, Kalman [15] presented a theory to optimally estimate the state of a linear dynamical system, showing that the estimation minimises the mean of the squared error, given that the noise is normally distributed. The Kalman filter computes the optimal state estimate by recursively combining previous estimates with new observations. It consists of two phases: prediction, where the optimal state is processed prior to observing; and update, where the posterior state is computed after observing. Additionally, it calculated the prior and posterior estimate error covariance.

Several applications of the Kalman filter have been implemented in different computer vision areas, namely in trajectory prediction and correction of tracked objects.

2.1.3 Hungarian Algorithm

Multiple Object Tracking introduces an assignment problem. The goal is to find which detection corresponds to which object based on the predictions from the previous step, or alternatively, if a detection represents a new object. By comparison of the Euclidean distance between a detected object in two adjacent frames, the Hungarian algorithm is a viable method to do the linking process between identified objects by finding the optimal matching solution that minimizes the Euclidean distance in the assignment matrices [16].

2.2 Object Detection

Generic object detectors are mainly divided into one-stage and two-stage detectors. With one-stage methods, the output can be obtained after one CNN operation, obtaining both locations (bounding-box coordinates) and classifications (class probabilities) at once. This method was made popular by successful real-time models such as You Only Look Once (YOLO) [17], Single Shot MultiBox Detector (SSD) [18] and RetinaNet [19].

Two-stage detectors consist of a region proposal based framework which is a two-step process: initially it gives a coarse scan of the whole image, generating proposals, and then focuses on each of those regions of interest, classifying each one into different object categories [20]. This architecture was popularized by models such as OverFeat [21] and R-CNN [22]. The latter has

been improved through FasterR-CNN [23] and Mask R-CNN [24], making the current state-of-the-art in accuracy for object detection on MSCOCO [25], a dataset designed by Microsoft for object recognition with a collection of images containing common objects in their natural context that would be easily recognized by a child. The subjects performance is benchmarked by different measurements of mean Average Precision (mAP).

With this type of detectors, since the high score region proposals obtained from the first-stage CNN are fed to the second stage CNN for a final prediction, the inference time is equal to $T_{total} = T_{one} + mT_{two}$, where m represents the number of region proposals with a confidence score higher than a certain threshold. For real-time object detection, the inference time cannot be variable over time so live detectors use mostly one-stage methods.

Figure 2.3 provides a block diagram of both one-stage and two-stage detectors.

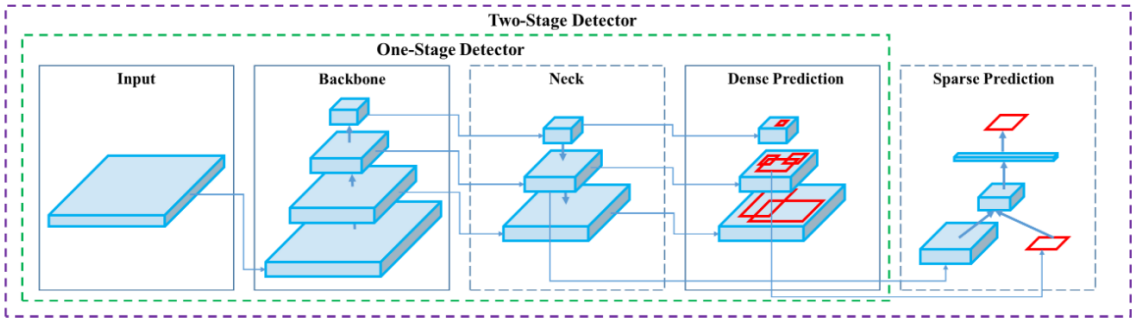


Figure 2.3: Block diagram of two-stage and one-stage detectors [26].

2.2.1 Real-time Object Detection

An algorithm that stands out in real-time object detection is YOLOv4 [26], an improvement of the original YOLO [17]. YOLOv4's different approaches have three top ten placements (including current second place). while the original YOLOv4, without extra training data, is still 36th [1]. Considering the need for real-time object detection, the model YOLOv4-CSP-P7 [27] is first.

YOLO uses the characteristics of a single image to predict multiple boxes, each containing one object. The original model, introduced in 2017 by J. Redmon *et al.*, consists in the following procedure (see figure 2.4):

- Division of the input image into a S×S grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object;
- Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and how accurate it thinks the box is. Confidence is defined as $Object * IoU$: if no object exists in the cell, the confidence score should be zero, otherwise the score should be equal to the intersection over union between the predicted box and the ground truth;

[1] Browsing State-of-the-Art in Object Detection

- Each grid cell also predicts C conditional class probabilities. These probabilities are conditioned on the grid cell containing an object. There is only one set of class probabilities predicted per grid cell, regardless of the number of boxes B;
- Multiplying the conditional class probabilities (C) and the individual box confidence predictions (B) gives the class specific confidence scores for each box. These scores determine both the probability of that class appearing in the box and how accurately the predicted box contains the object.

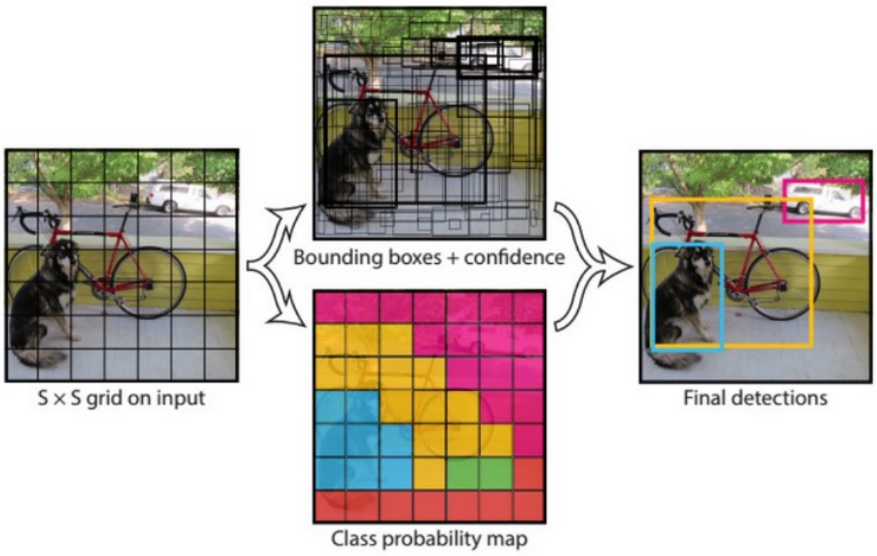


Figure 2.4: YOLO's visual representation [17].

The popularity and success of YOLO granted continuous work on the model, with updated versions such as YOLOv2 in 2016 [28] and YOLOv3 in 2018 [29] being introduced, both with novel strategies such as batch normalization, anchor boxes, multi-scale training (YOLOv2) and multi-label classification (YOLOv3) to improve both accuracy and speed.

The most recent, YOLOv4 [26], developed by Chien-Yao Wang *et al.* in 2020, is the basis of the current state-of-the-art real time object detectors. It uses the CSPDARKNET53 [26] neural network as backbone and the influence of state-of-the-art "Bag-of-Freebies" and "Bag-of-Specials" object detection methods during detector training, making it now more efficient and suitable for single GPU training. Figure 2.5 compares the current YOLOv4 with other state-of-the-art object detectors as well as the previous iteration, YOLOv3.

The model YOLOv4-CSP-P7 [27] by Wang *et al.* adds model scaling to the previous iteration with CSPNet, effectively cutting down 40% of computation of the backbone, increasing speed. It executes the Real-Time Object Detection benchmark with a mean AP score of 55.4 and execution time of 16 frames per second while not using extra training data [2]. Table 2.1 includes a

[2] Real-Time Object Detection on COCO

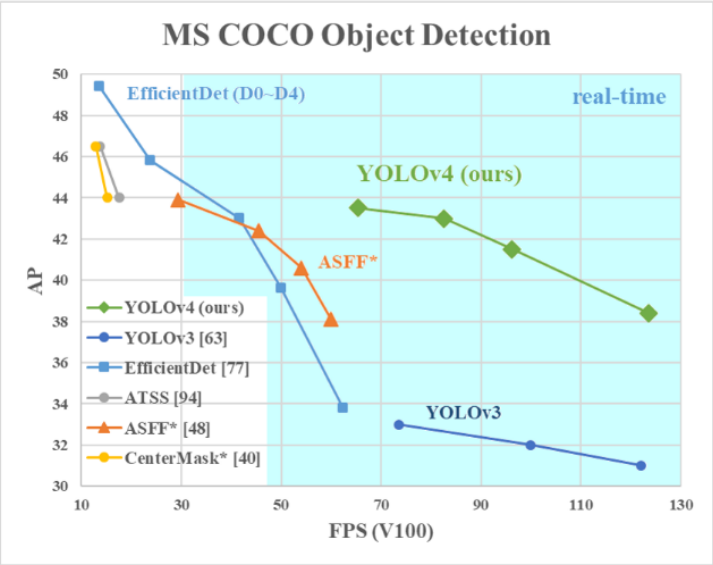


Figure 2.5: YOLOv4 testing under COCO benchmarks [26].

direct comparison of state-of-the-art object detectors, listing in bold the top two candidates of each comparing measurement.

Table 2.1: Comparison of state-of-the-art object detectors [27].

Method	Backbone	Size	FPS	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
EfficientDet-D0 [30]	EfficientNet-B0 [31]	512	97*	**34.6%**	**53.0%**	**37.1%**	**12.4%**	**39.0%**	**52.7%**
YOLOv4-CSP	**CD53s**	512	97/93*	**46.2%**	**64.8%**	**50.2%**	**24.6%**	**50.4%**	**61.9%**
EfficientDet-D1 [30]	EfficientNet-B1 [31]	640	74*	40.5%	59.1%	43.7%	18.3%	45.0%	57.5%
YOLOv4-CSP	**CD53s**	640	73/70*	**47.5%**	**66.2%**	**51.7%**	**28.2%**	**51.2%**	**59.8%**
YOLOv3-SPP [29]	D53 [29]	608	73	36.2%	60.6%	38.2%	20.6%	37.4%	46.1%
YOLOv3-SPP ours	D53 [29]	608	73	42.9%	62.4%	46.6%	25.9%	45.7%	52.4%
PP-YOLO [32]	R50-vd-DCN [32]	608	73	45.2%	65.2%	49.9%	26.6%	47.8%	57.2%
YOLOv4 [26]	CD53 [26]	608	62	43.5%	65.7%	47.3%	26.7%	46.7%	53.3%
YOLOv4 ours	CD53 [26]	608	62	45.5%	64.1%	49.5%	27.0%	49.0%	56.7%

[1] FPS value with * means overall latency, which include model inference and post-processing.
[2] APs value with **bold font** means the value is higher than all method which has higher FPS.

2.3 Real-time Object Tracking

Regarding multiple object tracking, MOTChallenge [33] provides a framework for fair evaluation of multiple people tracking algorithms, including challenges with subsets of data for specific tasks such as 3D tracking. The models are ranked using many evaluation measures, the most relevant ones being [3]:

- MOTA - Multi-Object Tracking Accuracy. This measure combines three error sources: false positives, missed targets and identity switches.

[3] MOTA, IDF1 and MOTP are subject to standard deviation, represented by ($\pm$ std_dev).

- IDF1 - The ratio of correctly identified detections over the average number of ground-truth and computed detections.
- MOTP - Multi-Object Tracking Precision. The misalignment between the annotated and the predicted bounding boxes.
- FN - The total number of false negatives (missed targets).
- Hz - Processing speed (in frames per second excluding the detector) on the benchmark. The frequency is provided by the authors and not officially evaluated by the MOTChallenge.

With ASVs, it is important to consider solutions with online methods (i.e. the solution is immediately available with each incoming frame and cannot be changed at any later time) in order to track in real life scenarios.

From the MOT20 Benchkmark Statistics [4], the three models with the highest reported processing speeds are GMPHD_Rd20 [34], Surveily and SORT20 [35].

GMPHD_Rd20

Baisa *et al.* developed in 2020 a tracking-by-detection model [34] that uses a Gaussian Mixture Probability Hypothesis Density (GM-PHD) filter with deeply learned CNN features to achieve linear complexity with the number of objects and observations and a real-time tracker able to track multiple targets in video sequences. This is achieved by integration of spatio-temporal and visual similarities obtained from bounding boxes of objects and their CNN appearance features at the update step of the GM-PHD filter and at the target labelling stage, which is done with the Hungarian Algorithm. A block diagram of the GMPHD_Rd20 model is displayed in figure 2.6. According to the authors, the GM-PHD filter is robust to false positives but very susceptible to miss-detections, being the original PHD filter designed for radar tracking applications where the observations collected could contain numerous false alarms with very few miss-detections. To counterbalance this characteristic, additional unassigned tracks predictions were included after the association step and deeply learned CNN appearance representations were used to re-identify lost objects, consistently labeling them.

A MOTA score of 44.7 ± 20.07 with run-time of 25.2 Hz was achieved in the MOT20 benchmark, making it the most accurate online tracker with a run-time over 10 Hz. This result was achieved running on a single core 3GHz machine.

SORT

The original Simple Online and Realtime Tracking (SORT) [35], developed by Bewley *et al.* in 2016, aims at efficient and reliable handling of the common frame-to-frame associations while exploiting recent advances in visual object detection to solve edge cases and detection errors.

[4] MMOT20 - Results & Criteria

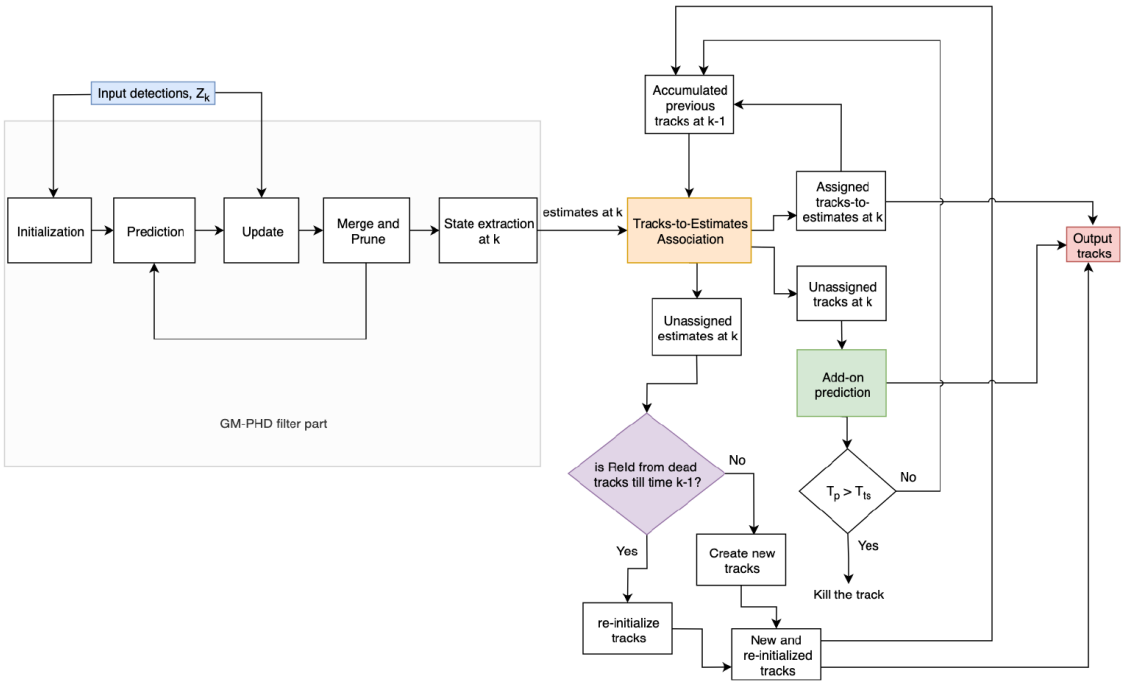


Figure 2.6: GMPHD_Rd20 Block Diagram [34].

A CNN based detector is used to improve accuracy while the minimalistic approach of using a combination of familiar techniques, such as the Kalman Filter and the Hungarian algorithm to predict and associate data respectively, ensures both efficiency and reliability for online tracking.

For the estimation model, when a detection is associated to a target, the detected bounding box is used to update the target state where the velocity components are predicted with a Kalman filter framework. If no detection is associated to the target, its state is predicted without correction using a linear velocity model. When objects enter and leave the image, unique identities are created or destroyed accordingly.

Any detection with an overlap lower than IoU_{min} signifies the existence of an untracked object. The tracker is initialised using the geometry of the bounding box with the velocity set to zero and the covariance of the velocity component is initialised with large values, reflecting the uncertainty of an unobserved velocity. The new tracker then undergoes a period where the target needs to be associated with detections to accumulate enough evidence in order to prevent tracking of false positives. Tracks are terminated if they are not detected for T_{Lost} frames. Should an old object reappear, tracking resumes under a new identity.

The presented methodology was able to achieve decently accurate results while being faster than other state-of-the-art trackers. A score of 42.7 ± 18.6 with run-time of 57.3 Hz was achieved in the MOT20 benchmark making it the fastest registered online tracker with a run-time over 10 Hz. This was achieved running on a quad-core 2.5GHz machine.

However, with SORT, the association metric with the Kalman Filter is only accurate when state estimation uncertainty is low. Occlusions which drastically increase uncertainty return a relatively high number of identity switches. DeepSORT [36], by Wojke *et al.*, attempts to correct

the issue by replacing the association metric with a more informed feature vector that combines motion and appearance information. The feature vector is obtained applying a CNN to compute bounding box appearance descriptors for the type of objects intended to track. The latter model is able to track objects through longer periods of occlusions, effectively reducing the number of identity switches by 45% while achieving overall competitive performance at high frame rates. An exemplary output of DeepSORT object tracking is illustrated in figure 2.7.

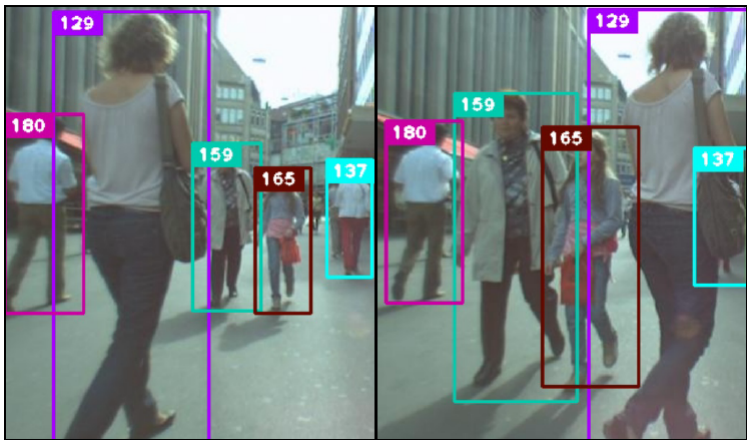


Figure 2.7: Exemplary output of DeepSORT on the MOT challenge dataset in a common tracking situation with frequent occlusion [36].

2.4 Object Tracking in the Context of ASVs

This section presents the research done for object tracking regarding ASVs. Considering the multi-modal approach to tracking developed in this work, a study has been made about different tracking models using different kinds of sensors typically available in an ASV such as cameras, LiDARs, AIS, GPS and IMUs.

2.4.1 Radar-based Tracking

In 2019, Freire *et al.* [37] published their model for obstacle avoidance combining a radar for shore detection, and an Automatic Identification System (AIS) to relay information regarding the locations of the bigger vessels that have an AIS transceiver, while using LiDAR to detect smaller vessels and debris. GPS and IMU data are relied on for extraction of the own ship's position and orientation.

For aggregation, considering that a full radar turn was of approximately two seconds, conventional methods such as K-means and DBScan [38], which during testing had execution times larger than the radar turn period, were not feasible. Instead, an algorithm that uses the same concept of DBScan but instead processes data scan-line by scan-line with an additional logic for evaluating the end of a cluster was proposed. This strategy revolves around the use of a timeout mechanism to periodically check if the difference between the current time and the time of the last

aggregation was bigger than a certain threshold. If so, the clustered data should have an adequate size to not be considered noise. Similarly, if the segment detections were bigger than two seconds, a timeout was applied, publishing every single point belonging to that cluster to the classification node.

The implemented classification node took a set of object measures, such as its centroid and weighted centroid, to form an elliptical model. Given that an ellipse is a 2D representation of a Gaussian distribution, the mean, variance and covariance values from the cluster's points characterize the object. The ellipse parameters were matched with the bounding box parameters: if the obstacle is an ideal ellipse inscribed within the rectangle and both parameters are similar enough, then the obstacle is classified as a vessel, otherwise, part of the shoreline. Segments which are approximately straight lines are merged to define the shoreline.

For tracking, firstly a Matching Filter was applied to receive object information from the object classification node. The weighted centroid, size and orientation of the objects were compared and, since the targets to be tracked had slow system dynamics, the Kalman filter could be used to accurately predict and track their trajectories. Figure 2.8 illustrates the results of the shore detection and the Kalman filter tracking techniques.

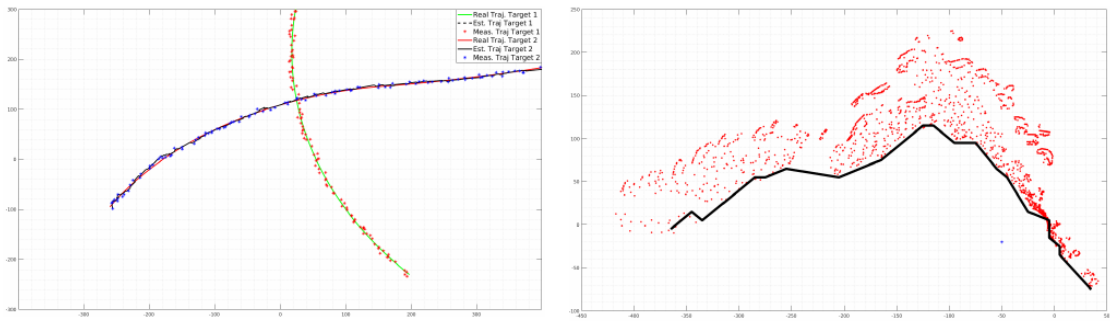


Figure 2.8: Output of the Kalman filter simulation tracking (left) and shoreline extraction algorithm (right). The image on the left plots the trajectory of two targets with their real trajectory traced with green and red lines respectively, the measured trajectories in blue and blue dots respectively, and finally, the estimated trajectories represented in black, closely following the real lines [37].

2.4.2 Visual-based Tracking

River Navigation

In order to provide a solution for river navigation when the GPS signal may be lost or blocked by thick and high canopy, Tan *et al.* [39] developed a system capable of detecting obstacles using a digital camera. The author concluded that even though IR and ultraviolet sensors are more common in autonomous robot navigation systems, the water environment brightness characteristic would be better captured by a digital camera.

For this model, each captured frame is converted to a gray-scale image, to which is applied a threshold filter to separate objects from water. A morphology operation is used to remove imperfections introduced during segmentation and a Canny filter algorithm is applied for accurate edge detection. Lastly, Hough Transform is used to detect the shoreline and to draw a straight line defining the boundary.

The GPS location is obtained from a GPS module and the latitude and longitude coordinates are plotted by using *Google Earth*. Figure 2.9 shows simulation results of shoreline tracking, confirming the efficiency of the simple implementation.

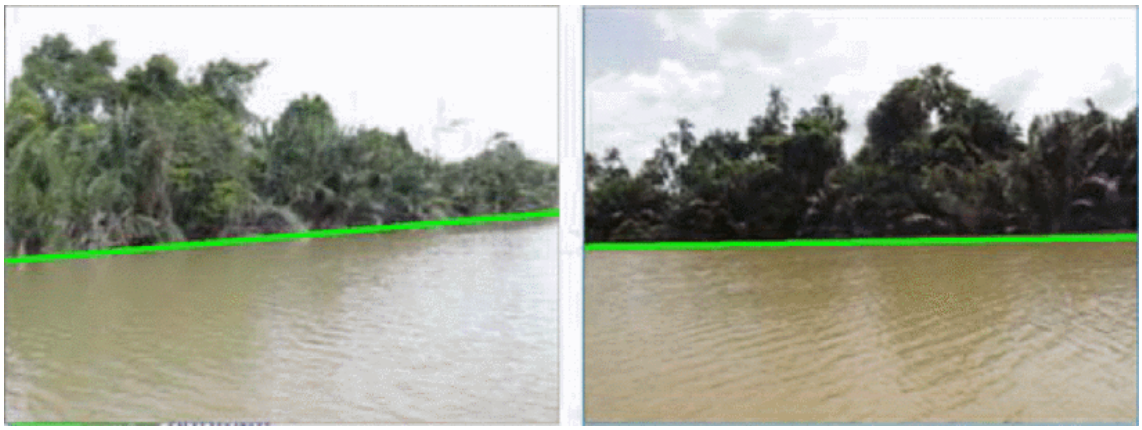


Figure 2.9: Simulation results of shoreline detection constrained Hough Transform [39].

Monocular Vision

In 2015, Cho *et al.* presented a vision-based real-time detection and tracking system [40] used to track a surface ship using a monocular camera in order to improve observability in congested waters with small boats. So as to minimize the typical noise of an inland water environment (e.g., reflections and glints of light from the water surface, ship wakes), computer vision techniques were tested. Feature Accelerated Segment Test (FAST) [41] was applied to extract the relative bearing and range measurements of a target ship from camera images. The measurements were processed using a filter based on the Extended Kalman Filter (EKF) to estimate the trajectories of detected targets with a Continuous White Noise Acceleration (CWNA) model.

Since the speed and maneuvering of the own ship caused observability issues, two approaches were taken to solve the problem. First, the own ship moves in a zigzag motion to ensure sufficient observability. Second, the range information from the horizon is used to improve the overall performance of the tracking filter, helping in prevention of an excessive divergence.

The tracking algorithm is comprised of five steps, as shown and described in figure 2.10. Firstly, the optical camera is used for searching the target ship. Once the target is detected, a way-point is assigned and the way-point tracking step is initiated. Feature detection is used to ensure the target ship is the one being tracked. This is a necessary step due to noise inherent in the water environment.

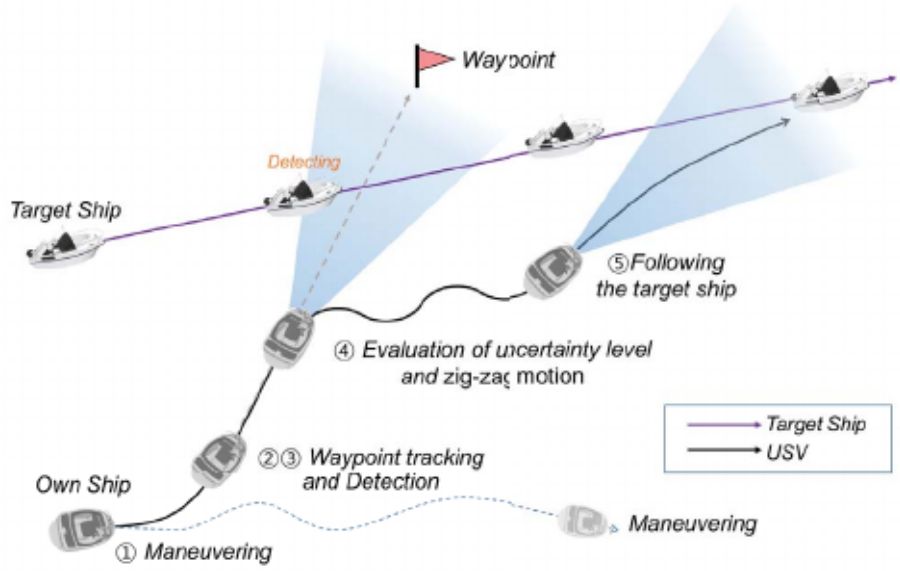


Figure 2.10: Strategy of a tracking target ship [40].

To reduce the noise inherent from water environments, image filtering algorithms were applied. Assuming that a ship or object must appear under the horizon line in the frame, the Region of Interest (RoI) for the search of a target ship is confined below the horizon, allowing for computation efficiency and faster detections. Once the RoI is defined, the feature from the FAST corner detector is applied to find salient target features in the image. While navigating towards the target, whenever the uncertainty level μ_L is higher than a predefined threshold γ_T, the ship changes to a zigzag motion until the uncertainty becomes lower than the threshold and then the ship keeps tracking but following the target in a straight line.

To enhance the accuracy of the tracking module, relative bearing and range measurements were taken from the frames. These measurements were determined using feature points of the target ship projected on the image plane considering the camera geometry and the relative coordinates in the own ship's reference frame, with the relative bearing measurement being proportional to the pixel distance from the center point of the target to the vertical line that is perpendicular to the horizon while passing through the center of the image. To find the central point of each feature extracted in the detection region, DBScan [38] was used. The relative range measurement is determined by the vertical pixel distance from the horizon to the lowest feature point of the target ship within the detection region. Figure 2.11 helps visualise the measurements taken.

In order to collect the own ship's position, speed, and heading angle, measurements were taken with GPS and IMU modules installed in the vessel.

Tracking By Detection Model (M^3C)

Despite state-of-the-art object detection and tracking for generic objects having recently demonstrated impressive performances, Qiao *et al.* deemed them inadequate for the complicated mar-

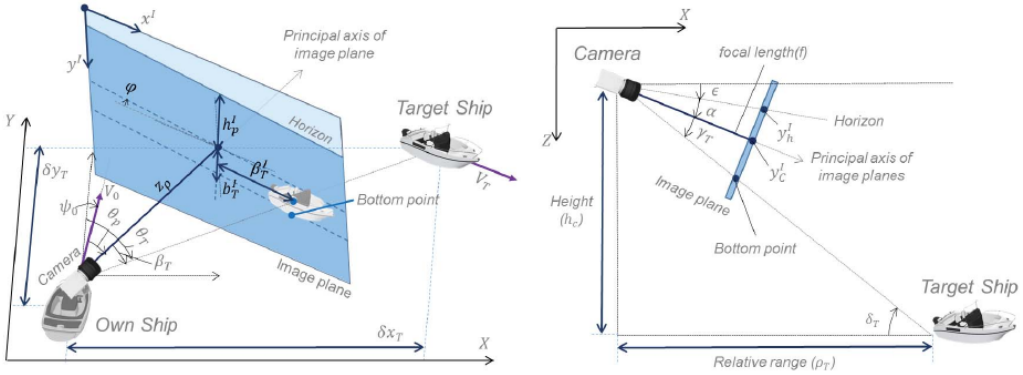


Figure 2.11: Camera geometry and coordinate systems [40].

itime environment and proposed a novel tracking by detection framework combining multi-model and multi-cue (M^3C) techniques to re-identify vessels during tracking [42]. The overall architecture is detailed in figure 2.12.

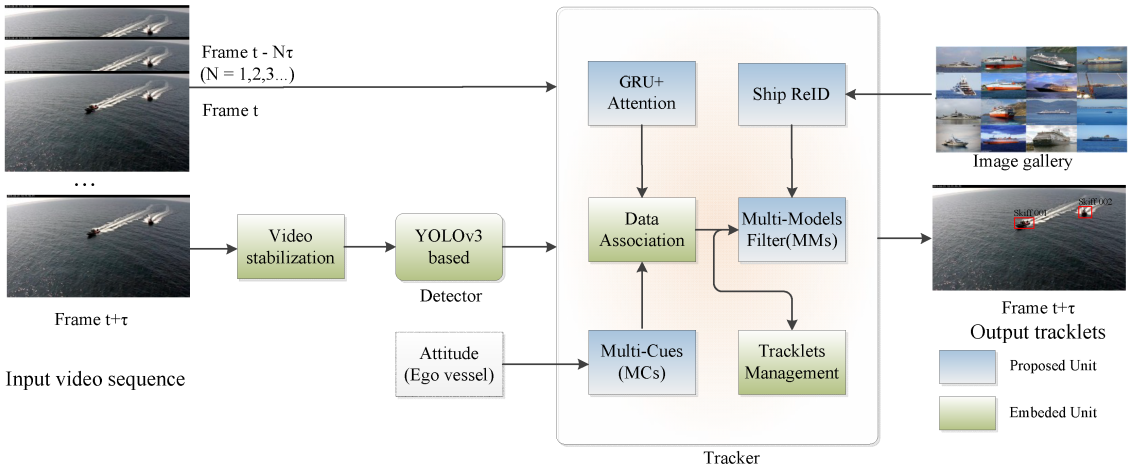


Figure 2.12: The architecture of the proposed M^3C tracking by detection pipeline [42].

It uses a YOLOv3 [29] based detector to obtain the bounding box and class of the objects by looking at the input each frame of the video input only once. When feeding this model a 608×608 pixel image, it can conduct three different scale predictions.

To compute the maneuvering prediction of a surrounding vessel, the estimation of the matching probability of each candidate model at the next moment based on the vessel's historical trajectory is calculated, while considering the ego vessel's posture via recurrent neural networks.

The kinematic behaviour of maritime vessels is predefined by three common kinds of motion maneuvers: constant velocity (CV), constant acceleration (CA) and curvilinear motion (CM), which are fed to a Kalman filter to predict the target vessel's motion state. This multi-modular approach is used to solve the problem of unstable tracking of a maneuvering target in the traditional single-model Kalman trackers applied in more popular scenarios such as tracking people.

The data association procedure has a hybrid affinity model based on multi-cues to evaluate the similarity between the detected vessels and existing tracklets, containing both long-term cues (appearance) and short-term ones cues (motion measurements and the dynamic attitude of ego vessel, such as pitch and roll). Adaptive association gate of appearance is used to confirm the validation of measurements before the data association algorithm is carried out.

Experiments have demonstrated its efficiency and robustness while still achieving real-time performance. Table 2.2 compares the model's performance with other popular tracking models.

Table 2.2: Quantitative evaluation of different trackers on two maritime datasets [42].

Dataset	Tracker	MOTA↑	MOTP↑	MT↑	ML↓	IDS↓	PFS↑
SMD (onshore) + PETS 2016	MDP	30.3%	71.3%	13.2%	38.4%	426	1
	SORT	59.8%	79.6%	25.4%	22.7%	631	**56**
	KCF	70.3%	80.2%	**37.8%**	22.3%	382	25
	POI	66.1%	79.5%	34.6%	**20.8%**	453	10
	M^{3}C (Ours)	**72.8%**	80.4%	37.4%	21.2%	**254**	20
SMD (onboard)	DeepSORT	60.4%	**79.1%**	32.8%	18.2%	56	**36**
	MOTDT	57.6%	70.9%	**34.2%**	28.7%	49	23
	M^{3}C (Ours)	**63.4%**	74.6%	26.2%	**17.9%**	**34**	16

2.5 Critical Review

This section contains a short overview of the researched material concerning the state-of-the-art methods described in this chapter. Regarding generic object tracking, the main challenges such as occlusion, identity switches and view-point variation were identified. Furthermore, investigation of strategies currently developed for tracking in maritime scenarios brought attention for several additional challenges that impair vision-based solutions such as water reflection, harsh illumination scenarios. Additionally, the inherent surface vibrations cause sway on the on-board sensors.

The subject of tracking, while popular and widely researched in aerial and terrestrial environments, hasn't yet seen similar growth for the maritime scene. As such, popular state-of-the-art tracking models that adopt mainly machine learning techniques for detection and re-identification of targets, which depend heavily on the availability of wide datasets to achieve their potential, won't see similar success on this maritime area with scarcer data. The biggest open datasets (i.e. ImageNet [43], PASCAL VOC [44] and MSCOCO [25]) contain over a million multi-purposed images but only one class related to a maritime scenario (boat) which translates into a bundle of only hundreds of actually usable annotated images. Vessels have can have different shapes and sizes. A row-boat is completely different from a cargo-ship so extracting features from those images without proper reclassification would be a very difficult task. There is a lack of sufficient work in gathering large amounts of annotated data for this context.

Generic object detection and classification are well covered topics with two main frameworks. As a preliminary sub-process of tracking, only the single-stage regression models are viable due

to their real-time application. Of those, YOLOv4 [26] stands out as the method with the best performance.

For the tracking algorithm itself, the problem can be separated in three parts: assignment or re-identification, trajectory prediction and positional update. The researched methods relied mostly on the Kalman filter to estimate the target's position, with varying motion equations. Qiao *et al.* demonstrated a multi-modal approach with three different kinetic states that showed a lot of promise, emerging as the better performing tracking model in maritime environments. The re-identification part is handled differently depending on the sensors utilized. The current state-of-the-art trackers, DeepSORT [36] and GMPHD_Rd20 [34], rely on the Hungarian algorithim while M^3C [42] adopts a multi-cue approach with both visual and motion information to minimize the issues encountered with the state-of-the-art trackers in maritime scenarios due to impaired vision. Comparatively the work of Freire *et al.*, using LiDAR technology to classify clusters through an elliptical model, resorting to a matching algorithm, established an effective method to deal with the previously mentioned unreliable visual conditions of an unmanned vessel in the middle of the sea.

Despite the results being quite impressive in the referenced approaches, the tracking topic remains quite unexplored. The state-of-the-art trackers aren't designed to account for the challenging seafaring environment as they rely on camera images for object detection, making association difficult when the observer is affected by rippling or water reflection. Alternatively, the maritime approaches tend to address specific scenarios. The work by Tan *et al.* [39] focuses on shoreline navigation, not addressing traffic tracking. Cho *et al.* worked on a tracking system with the goal of following a specific target [40], while the work of Qiao *et al.* is indeed successful at general tracking performance but the developed model for predicting the target's kinematic behaviour heavily depends on the observer being stationary [42].

Chapter 3

A Multi-Modal Multi-Object Detection and Tracking System

3.1 Introduction

This work proposes to develop a tracking model to work on-board the SENSE ASV which detects multiple vessels in the field-of-view of the installed camera and tracks their position over time while resorting to its sensors, namely the camera and LiDAR, for feature extraction. This vessel is $1.5m$ long, $1m$ wide and approximately $1.5m$ tall. With payload it can weigh up to $75kg$. Its representative model, developed by *INESC TEC* can be visualized in figure 3.1. A *Gazebo* simulated model is used for developing software to collect data from each sensor.

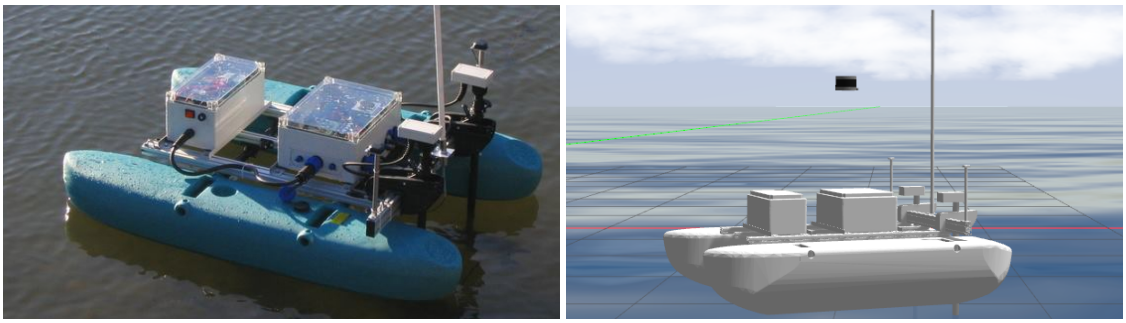


Figure 3.1: Picture of *INESC TEC*'s SENSE ASV and its representative model on the *Gazebo* simulator.

The SENSE ASV is equipped with a 3D LiDAR, a pair of stereo cameras, an IMU and a GPS. The spatial configuration of the described sensors is illustrated in figure 3.2 and the most relevant specifications from each of the sensors are as follows [45]:

- Camera - *Mako G-125* - Frame Rate: $30Hz$, Resolution: 1292×964, Field of View: $80°$ horizontal;
- LiDAR - *VLP-16* - Frame Rate: $4Hz$, Range: $100m$, Range Accuracy: $0.03m$, Field of View: $360°$ horizontal and $30°$ vertical;

- IMU - *MTi-30Xsens* - Frame Rate: 200Hz, Angular Accuracy: 0.2°/0.5°;

- GPS - *Swift Navigation* - Frame Rate: 20Hz, L1/L2 RTK, Accuracy: 0.01m horizontal, 0.015m vertical.

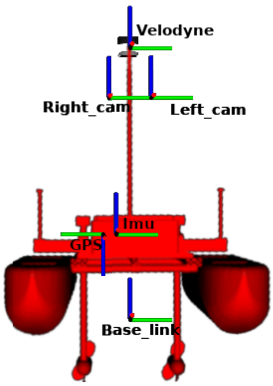


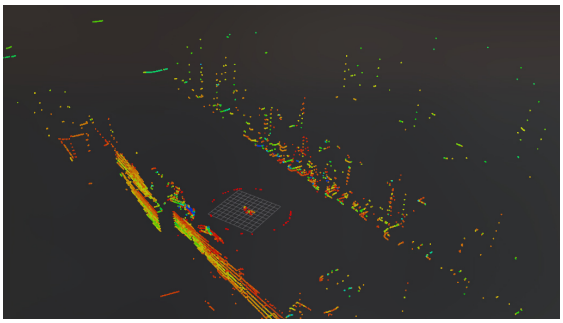
Figure 3.2: Visual representation of the SENSE ASV referentials for the camera and LiDAR sensors.

The data used for testing and evaluating the developed tracking model was collected during organised CRAS maritime missions in which the SENSE ASV was navigated around traffic in the Leixões and Viana ports, recording the camera and LiDAR information for further use. This recorded footage of about 100 minutes can be reproduced in real-time in the *ROS* environment, providing a simulation of a real testing scenario. Figure 3.3 pictures the data collection mission in the Leixões port. A similar procedure was put in practice during the Viana data collection mission.

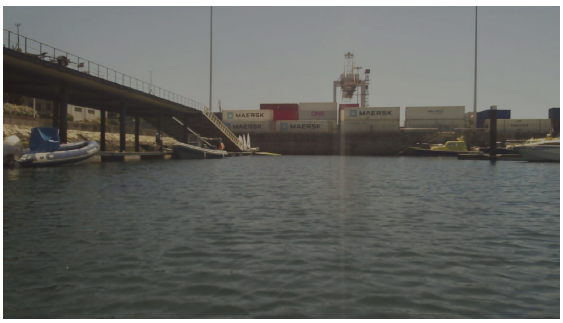
The developed tracking model consists of two primary modules. A detection module, responsible for handling the connection between camera and LiDAR information by identifying and classifying vessels in the camera's field-of-view and retrieving their respective 3D location from the LiDAR point cloud. For each single frame, a combination of 3D LiDAR distance measuring and camera vision must be sufficient for detecting and classifying each moving object. Traditional methods rely on hand-crafted features and probabilistic models. The advances in deep learning bypass this step, extracting important characteristics directly from the input [46]. Therefore, this step shall be handled by a detection module which employs a CNN based approach. Subsequently, the tracking module handles the tracing of the position of each detected object with two distinct sub-modules working in tandem. The association sub-module handles comparison and association of all identified objects in the current detection frame with their previous instances. To cover scenarios where previously detected objects are out of frame or undetected in the present frame (occlusions), a prediction sub-module maps and updates their position over time. A summary of the employed strategy is illustrated by figure 3.4.

(a) Hardware setup.

(b) Data collection stage.



(c) LiDAR captured frame.



(d) Camera captured frame.

Figure 3.3: Pictures taken of the Leixões mission. Figures (c) and (d) display the collected data visualized through *RViz* software.

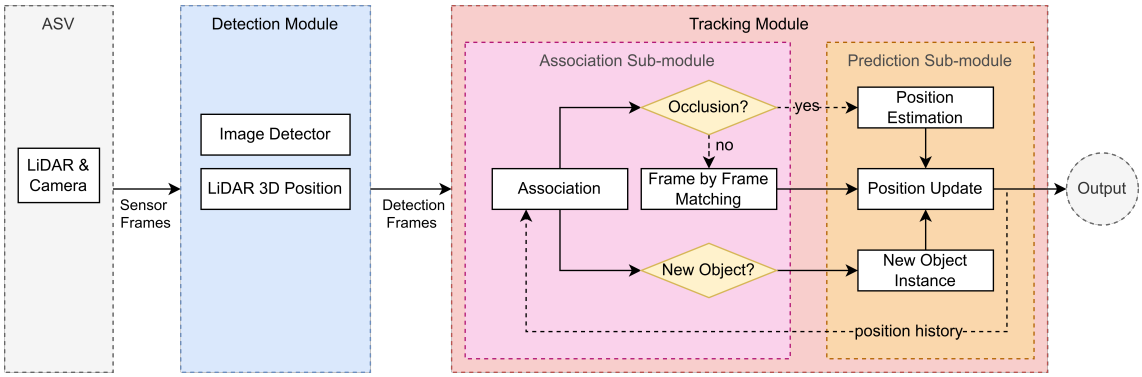


Figure 3.4: Block Diagram of object tracking in the context of ASVs.

3.2 Datasets and Annotation

In order to apply a CNN based approach in the proposed solution, firstly, an extensive dataset of marine vessels will be collected. This dataset shall include an amount of images with multiple boats and ships of different categories, that is suitable to train the detector module through transfer learning techniques. However, data concerning maritime objects with ground-truth and proper classification is not widely available. This section describes the gathering process of open datasets regarding maritime vessels, including manual annotation of images and footage gathering from the real testing scenario. The resulting dataset, contains 9044 images and their respective 21170 annotations, it is made available in the Datasense@CRAS repository [47]. Listed below are the

aforementioned datasets from which information was collected:

- The SeaShips Dataset, published by Shao *et al.* [48], contains a set of annotated real-world images covering six common ship types (ore carrier, bulk cargo carrier, general cargo ship, container ship, fishing boat, and passenger ship), designed for training and evaluating ship object detection algorithms. A sample version with 7000 of the 31455 claimed images is available in the author's website;
- The Singapore Maritime Dataset [49] provides on-shore and on-board recorded videos with matrix files of the bounding boxes of each object in every frame of all provided videos. The detector module requires a dataset composed of images and, as such, in order to use the Singapore dataset, frames were extracted each n seconds from On-Shore videos and a text file with the classification and bounding box of each object in the frame was generated, totalling 914 annotated images.
- On the Kaggle website, Clorichel released in 2018 a dataset of about 1,500 pictures of boats classified in 9 categories[1]. Only pictures were used, excluding images that don't represent real scenarios (i.e vector-arts, boat renders). The Kaggle dataset doesn't come with annotations, hence, the photographed objects have to be manually identified and classified;
- The aforementioned datasets provide images of different boats in different scenarios. However, to better represent the future testing scenario, the SENSE ASV sailed through two nearby ports: Leixões and Viana do Castelo, recording video frames through its on-board cameras, along LiDAR. GPS and IMU data. The processes of frame extraction and object identification provided around 200 annotated images.

An example of the gathered images can be visualized in figure 3.5.

3.2.1 Object Identification and Classification

The annotations for the Kaggle dataset and the collected data from the Leixões and Viana do Castelo missions with the SENSE ASV were taken through the open-source software *LabelImg*[2], available on *github*. Each vessel was delimited by a bounding box and categorized, as demonstrated in figure 3.6.

Labels were organised in 9 categories (ore carrier, bulk carrier, container ship, ferry boat, sail boat, fishing boat, small boat, uncategorized), grouping up some less represented categories from the previous datasets with similar characteristics (i.e. kayaks, gondolas are under small boat; buoys and surf/paddle boards which aren't boats get the uncategorized label). Annotations taken for the images in 3.5 can be viewed in 3.7.

The annotations were exported to *XML* files with information about every object in a single image, structure is detailed in 3.8. For the machine learning process, these annotations were

[1] Boat types recognition: About 1,500 pictures of boats classified in 9 categories
[2] Tzutalin. LabelImg. Git code (2015). Free software: MIT license

Figure 3.5: Examples of pictures that compose the collected dataset.

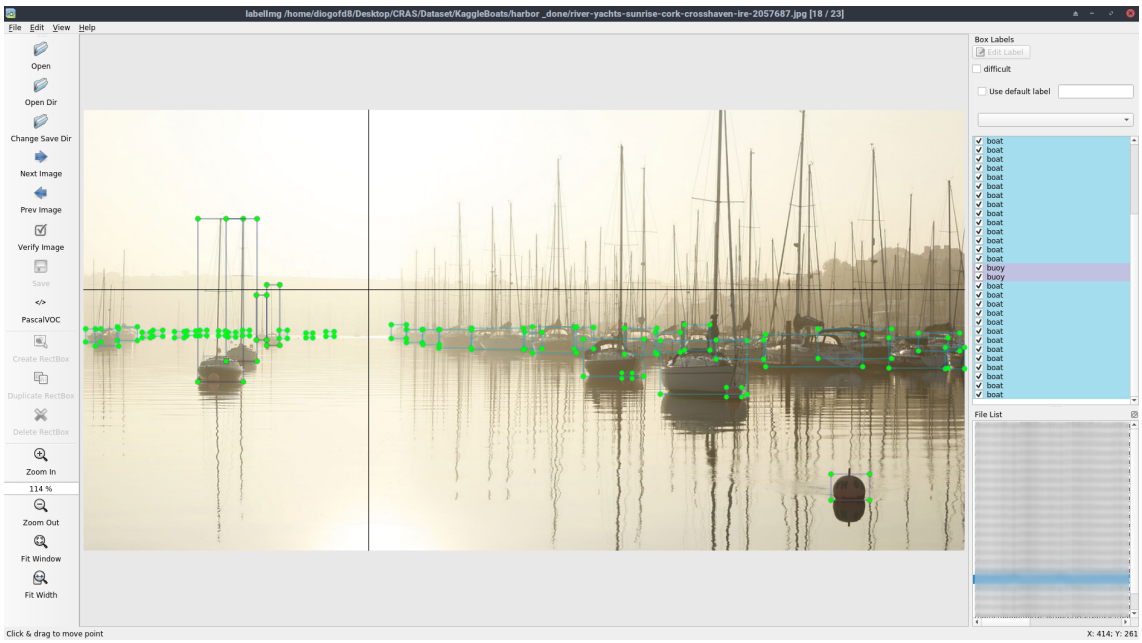


Figure 3.6: Framework of LabelImg loaded with an annotated image containing multiple objects. The green dots highlight the corners of the bounding boxes. On the right-sided panel, each annotated object is defined by its label.

Figure 3.7: Examples of annotations from the collected dataset.

converted to the *YOLO Darknet Text* format which replaces text labels with numerical identifiers and normalizes the bounding box coordinates within the range $[0, 1]$, making them easier to work with after scaling or stretching images.

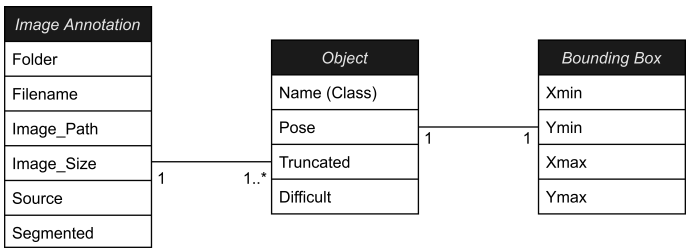


Figure 3.8: UML Class Diagram of a Pascal/VOC XML annotation file structure.

3.3 Detection Module

The main goal of this module is to accurately detect any maritime vessel. For this, two different training exercises will take place. The first is a binary classification method where the area inside

any bounding box is classified as a boat while the rest of the image isn't an boat. This attempts to train the neural network with general characteristics common in every maritime vehicle. The second exercise classifies every different type of vessel (i.e. sail boat, row boat, kayak, gondola). With binary classification, detections are expected to be more more reliable due to simplification. However, the advantage of having more information about the detected objects aids the association sub-module of the tracking module. With sufficient data, the more complex multi-class training would likely wield similar precision, proving to be a more suitable choice for a tracking scenario. As such, the accuracy of both training exercises should be measured in order to determine whether the drop in accuracy is significant enough to consider the binary classification method.

Lastly, LiDAR-based processing is implemented to enhance accuracy, purging false detections by projecting the 3D *point cloud* into each image frame and removing every bounding box whose object wouldn't fit a real detection. This step is accomplished by counting the amount of projected points inside the region of interest and determining whether the cloud is dense enough to be considered an object. LiDARs can be used during the day or night as they are a light emitting method, hence, not reliant on ambient light sources. Their ability to collect data in scenarios with poor lighting and without geometrical distortions makes them a very good complement for cameras.

3.3.1 Transfer Learning of the Detection Module

The scarcity of annotated maritime datasets for object detection and classification deemed unfeasible a performance comparison among popular object detectors with open-source implementations. However, state-of-the-art real-time object detectors such as YOLOv4 [26] have already proven to be effective in land vehicles detection as they come with pre-trained weights from extensive datasets containing popular categories. With transfer learning, the detector can be trained for different and more specific scenarios [50] [51]. In short, this technique reuses a pre-trained model on a new problem. The machine takes advantage of the knowledge gained from a previous task to improve generalization about another. This method has been popularized in deep learning as it can train deep neural networks with comparatively little data [52]. Figure 3.9 illustrates a simple block diagram of the transfer learning technique employed in this scenario.

Traditional augmentation techniques such as image rotation, flipping, distortion and mirroring are very effective in increasing the accuracy of classification tasks [53]. These techniques were applied during the learning process. With the purpose of minimizing the exposure issues created by the maritime environment, another image augmentation step was included, applying custom image filters such as histogram correction and sharpening. Figure 3.10 illustrates this process.

To suit the dataset to the training process, the acquired images from the Datasense@CRAS, detailed in 3.2 were split into three factions: training, validation and testing data. This allocation process must be carefully designed, especially for small datasets, to make sure that enough data is assigned to the training phase, allowing the algorithm to learn. The validation and test datasets cannot be too small either or they would not be fully representative of the collected data [54]. According to previous machine learning training exercises executed with similarly small datasets, a split of 90% of the available dataset was set aside for training purposes, from which 80% is

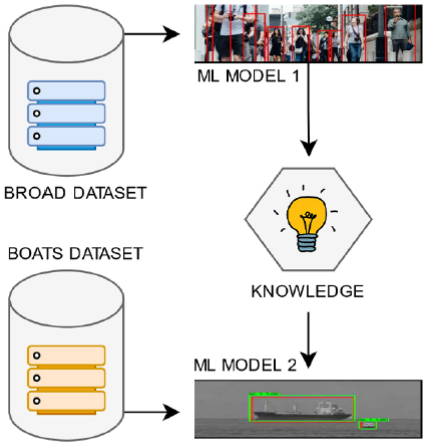


Figure 3.9: Block Diagram of the applied Transfer Learning technique.

used for the actual training process while 20% remains for validation, corresponding to 72% and 18% of the entire dataset, respectively. The remaining 10% are then used for testing the model after training, with novelty data to which the network hasn't been exposed before [55]. This split allocation process involved randomized selection of the images, to ensure that most conditions are equally represented in each data fraction. Figure 3.11 illustrates the structure of this dataset.

Optimisation of the Training Process

Optimisers are computational methods to update the weights of a neural network during a training instance. These algorithms update the parameters of a network through back-propagation to minimize the value of the loss function, finding the optimised value of the weights that produce the most accurate predictions. Three different optimisers were considered to train the network of the detector: Adadelta, Adam and Stochastic Gradient Descent (SGD). Each optimiser has different parameters, the most relevant one being the update rule. Table 3.1 sums up the update rules of the three selected optimisers:

Figure 3.10: Visual comparison of one of the applied filters to the original frame. The objects in the image are better exposed and contain more defined edges, allowing the detector to better identify their shape and colours.

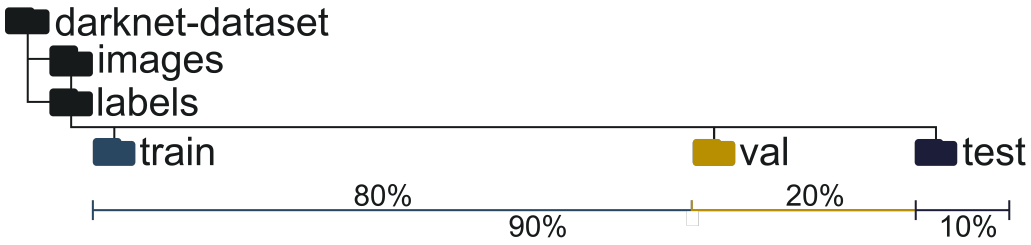


Figure 3.11: Diagram of the folder structure and dataset split where 10% belong to testing purposes while the remaining 90% are split between training, 80%, and validation, 20%.

Table 3.1: Update rule for the three optimisers.

optimiser	Update Rule
Adadelta [56]	$w_{t+1} = w_t - \frac{RMS[\Delta w]_{t-1}}{RMS[g]_t} g_t$
Adam [57]	$w_{t+1} = w_t - \frac{l_r \hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon_1}$
SGD [58]	$w_{t+1} = w_t - l_r g_t$

where l_r represents the learning rate, g_t the gradient of the loss function, m_t the moving average of the gradient, v_t the squared gradient. The w_t and w_{t+1} values are the weights in the $t-th$ and the $t-th+1$ iterations, respectively.

Different optimisers work best with different values of the learning rate and it is often necessary to experiment with several ones. The methodology employed to select the optimiser relates to the work of Pereira *et al.* in 2020 [55].

3.3.2 Camera and LiDAR 3D Position

Autonomous systems require a complete and accurate 3D perception of its surroundings to operate [59]. To acquire an image output containing the relative distance of every nearby object, an approach capable of fusing the information from the camera and LiDAR present in the ASV is required. A few aspects to keep in mind are:

- The camera topic provides a stream of 2D images while the LiDAR topic feeds a stream of 3D point clouds. As such, the 3D point clouds must be converted to 2D points before being masked into the camera image.
- As evidenced by figure 3.2, the two sensors aren't placed in the exact same spot, meaning that their point-of-view differs. The LiDAR point clouds require a translation operation to match both LiDAR and camera referentials.
- It is imperative that data from the same specific time-frame is collected. For that, synchronized reading of both inputs must be attained.

Data obtained through a LiDAR is also susceptible to environmental conditions. Tidal waves and wind cause constant bobbing of the ASV which can result in a very noisy point cloud [60]. An

effective approach is to crop the point cloud with a Region of Interest (RoI) that matches the field-of-view of the camera, while stabilizing the disturbances caused by the environmental conditions with GPS and IMU data. The point-of-view difference between sensors is handled by calculating the transform function that carries the relationship between the camera referential and the LiDAR referential, applying the transformation to the 3D point cloud. A comparison of both original and transformed point clouds is displayed in figure 3.12.

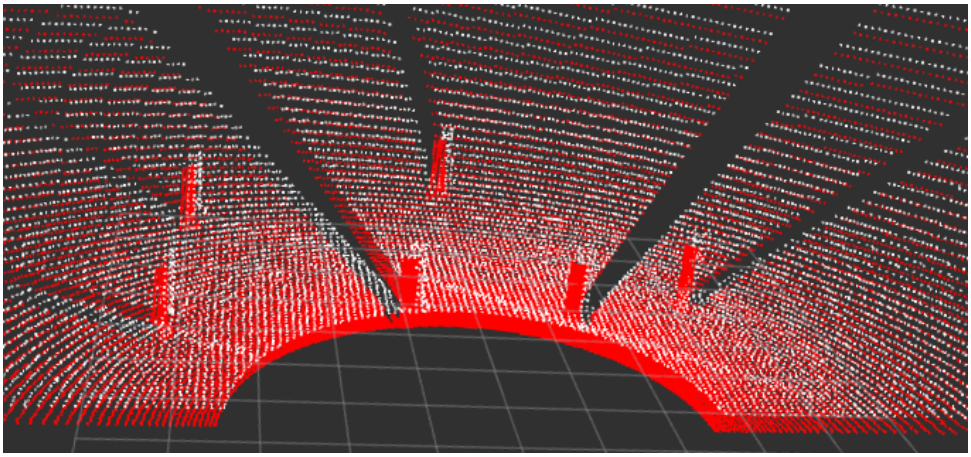


Figure 3.12: 3D plot of the point cloud through the software *Rviz*. The red dots represent the original point cloud from the LiDAR frame, the white dots illustrate the camera frame point cloud.

Lastly, to project the 3D point cloud into 2D coordinates, the intrinsic matrix K is applied. It contains the parameters of the camera and is used to apply the same translation and rotation between the camera's axis and the image plane by transforming 3D camera coordinates, $P_{3D} = (x, y, z)$ into 2D homogeneous pixel coordinates $P_{2D} = (w, h)$. This perspective projection is modeled by the ideal pinhole camera, illustrated below in figure 3.13.

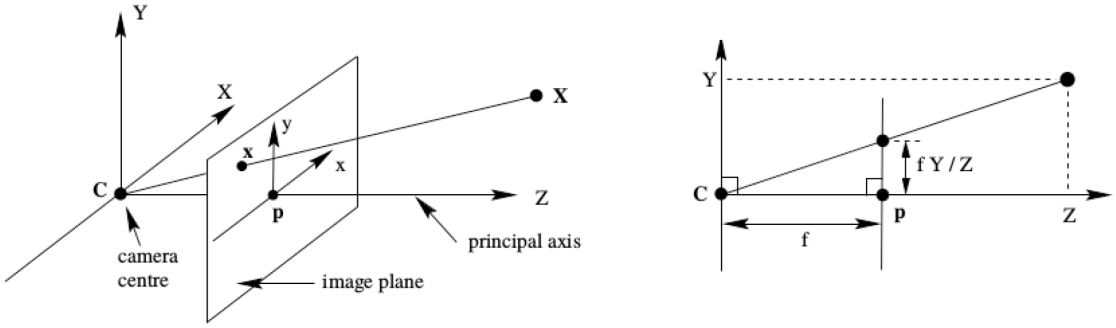


Figure 3.13: Pinhole camera model representation [61].

The center of the camera (pinhole) is represented by point C. An image plane is projected at a distance defined as the focal length, $f = (f_x, f_y)$, measured through the perpendicular line relative to the plane that passes through the pinhole. This line is the principal axis, Z. The intersection between the image plane and the principal axis defines the principal point, $p = (x_0, y_0)$. These parameters are divulged in the intrinsic matrix, presented in 3.1 (parameterized by Hartley and

Zisserman [61]). Resorting to the similar triangles created by the intersection of the principal axis with the ray that projects the point onto the image, such as the one represented in figure 3.13, the transformation from a $\mathbb{R}^3$ coordinate space to a $\mathbb{R}^2$ space can be computed using the equations presented in 3.2 and 3.3 [61]:

$$\begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} = \alpha \begin{bmatrix} p_u^{2D} \\ P_v^{2D} \\ 1 \end{bmatrix} \tag{3.1}$$

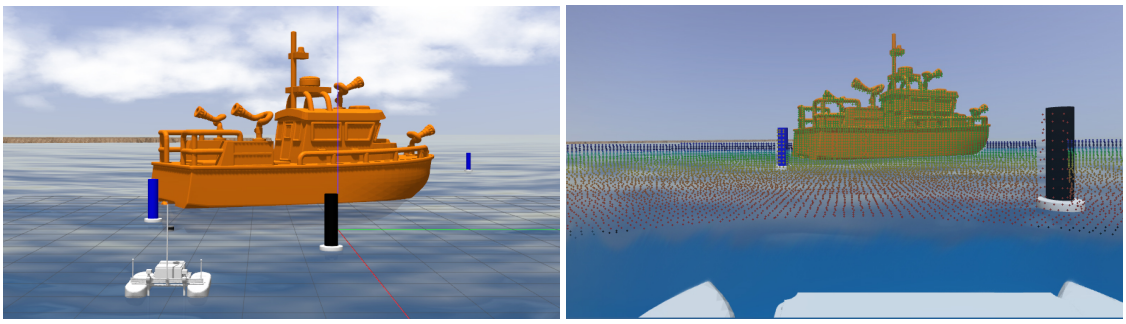
$$P_u^{2D} = \frac{f_x \cdot P_x^{3D}}{P_z^{3D}} + c_x \tag{3.2}$$

$$p_v^{2D} = \frac{f_y \cdot P_y^{3D}}{P_z^{3D}} + c_y \tag{3.3}$$

where P^{2D} is the projected point in pixels, P^{3D} the point belonging to the point cloud, f_x and f_y are representing the focal length and c_x, c_y are the distances from the origin to the principal point. The camera on-board the ASV has a 1292×964 pixel resolution. As such, the pixel values computed above must be filtered, discarding any pixel outside of the image frame (the RoI). Finally, to represent the depth of each 3D point in the reconstructed image, the pixel intensity is normalized according to equation 3.4 [62]:

$$I_{blue} = 255 \cdot \frac{P_z^{3D} - D_{min}}{D_{max} - D_{min}} \tag{3.4}$$

where I_{blue} is the value of the intensity given to the blue component of the RGB pixel, P_z^{3D} expresses the Z coordinate of the 3D point (frontal distance to the ASV), and D_{max}, D_{min} represent the maximum and minimum depth values of the points present in the cloud, respectively. This application is presented in figure 3.14.



(a) Third person view. (b) First person view.

Figure 3.14: 3D point cloud projection onto a 2D plane. The image (a) illustrates the 3D graphical simulation while the image (b) shows the point-of-view of the on-board camera. The colored dots are the 3D to 2D projected LiDAR points.

3.4 Tracking Module

In order to tackle the challenges listed in 2, the approach to the tracking module is composed of two different sub-modules: an object association module and a prediction module. For the association module, two lists of objects are considered. The first list, denominated *detected list*, corresponds to the objects detected at the current frame, in short the output of the detector module. For each of these objects, information about the class, 3D relative location to the observer, bounding box and predominant color, is extracted. The second list, or *tracked list contains* every object which was detected in previous frames. A distance matrix is then computed by calculating the euclidean distance between each element from the detected list and older objects from the tracked list, determining whether a detected object corresponds to the present instance of a tracked object or not. The prediction module resorts to motion equations and a Kalman Filter to estimate the position of each object for the next frame. This module predicts the position of each object in the future, allowing for its re-identification during occlusions.

3.4.1 Association Algorithm

The main goal of the association is to match objects detected in the current frame with their respective tracked instances, creating a new tracking instance whenever the association is determined impossible. The computed Euclidian distance measures the proximity or similarity between two objects. A number of measurements and detection characteristics can qualify to distinguish different objects, enabling an association of an object based on its similar features across frames. Such measurements are enunciated below:

- **LiDAR 3D location** - With the data fusion in 3.3.2, the same RoI generated by the object detector can be applied to the resulting data fusion map to select the points from the LiDAR point cloud belonging to the object. Their computed average distance introduces a three-dimensional position feature complementary to the two-dimensional bounding box location, adding a depth component which allows the tracker to distinguish similar objects passing through each other based on their 3D location history;

- **Vessel Classification** - As described in 3.2, the dataset used for training of the detector module has nine distinguished classes. This class information is used as input for the tracking module, reducing the amount of computations by restricting comparisons to objects with the same class label;

- **Predominant Color** - By averaging the (R,G,B) value of each pixel in the RoI of an object it is also possible to determine the predominant color value of a detected object, enabling comparisons according to the object's color characteristic. The average RGB value is converted to the Hue, Saturation, Value (HSV) colour space to have a single value (hue) determining the dominant colour type instead of the three values for each color channel, simplifying associations;

- **Bounding Box Aspect Ratio** - Each detected object is represented by a rectangular bounding box which comes with the respective image RoI. This rectangle should be proportionate between frames of the same object, hence, another metric used to compare objects in different frames is the aspect ratio of their bounding boxes: $\frac{Box_{height}}{Box_{width}}$. The metric is useful to distinguish nearby objects with overlapping boxes.

The listed features describe the state vector of each object, $X_k = [P_x, P_y, P_z, C_h, B_r]$, where P is the 3D location computed by the LiDAR RoI centroid, C_h the hue value of the object's RoI average color and B_r the object's bounding box ratio. To secure temporal persistence, for each tracked object, a vector of up to a maximum of ten X_k instances is stored. The metric used to compare each element from the detected list with the elements from the tracking list accounts on the distance between their respective state vectors. While the state of a detected object depends solely on the measurements at the current frame, for a tracked object the state vector depends on measurements between frame k and previous measurements from $k-n$, as depicted in algorithm 1. With this, a distance matrix with the distances between every detected object in frame k and the tracked objects from frames $k-n$ is created.

Algorithm 1 Association Distance Metric comparing each detected element at frame k with each element of the tracking list during $k-n$

Ensure: $Xk_{avg} \leftarrow [0,0,0,0,0]$
Ensure: $Xk_{detection} \leftarrow [P_x, P_y, P_z, C_h, B_r]$
  if $Class_{Detection} \neq Class_{Tracking}$ **then**
    $Distance \leftarrow -1$
  end if
  for $n = 0; n < X_{k-n}.size(); n++$ **do**
    $Xk_{avg} += \frac{X_{k-n}}{2^{n+2}}$
  end for
  $Xk_{avg} += \frac{X_k}{2^{n+1}}$
  $Xk_{avg} += \frac{LatestPredictionVector}{2}$
  $Distance \leftarrow |\det(Xk_{detection} \times Xk_{avg})|$

The flowchart in figure 3.15 exemplifies a scenario of the process behind the association module, considering a list of detected objects $A, B, C, (...), M$ and tracked objects $1, 2, 3, (...), n$ from which objects A and 2 are compared.

3.4.2 Prediction Algorithm

The Kalman Filter estimates the state of a system at time k, x_k, using the Kalman model true state equation 3.5, paired with the measurement model z_k that relates the state and the measurement at the current step 3.6.

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \tag{3.5}$$

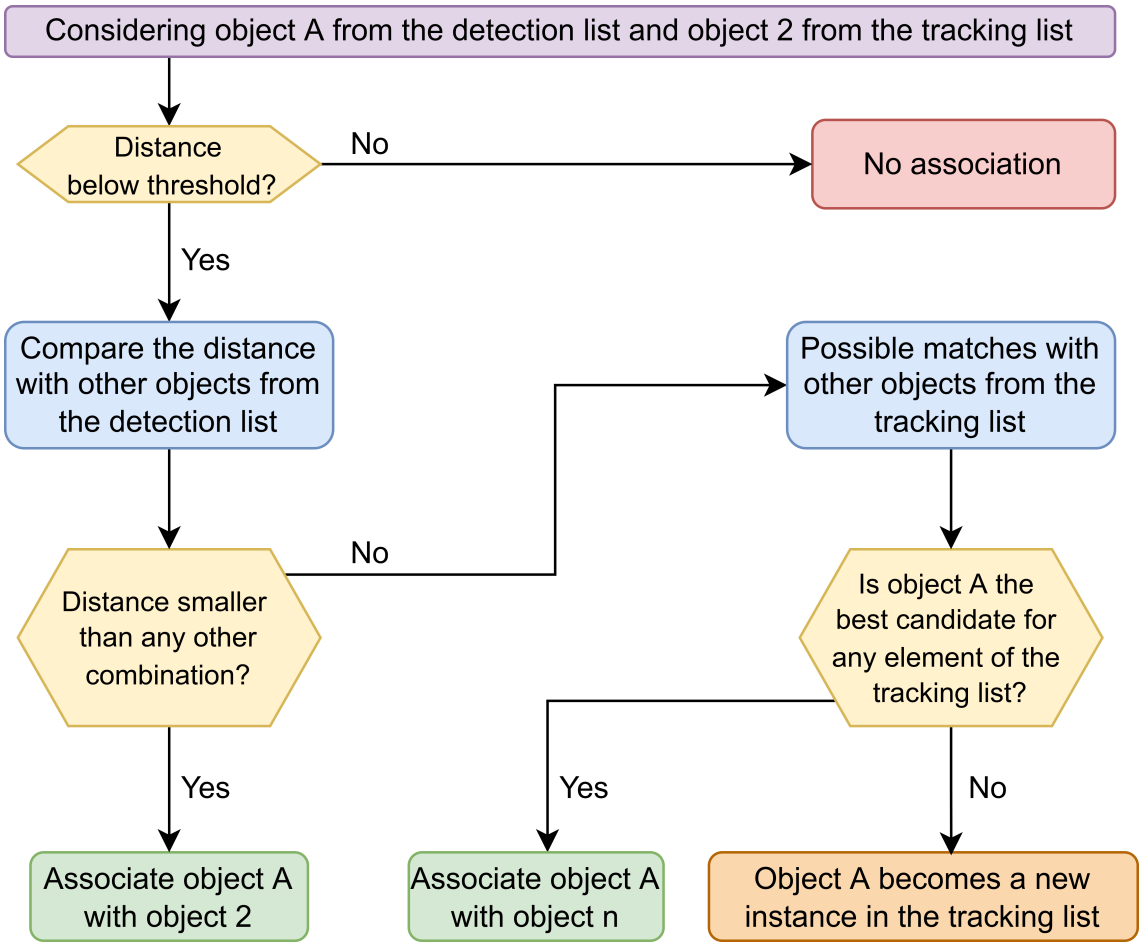


Figure 3.15: Flowchart with the general mindset for associating detected objects with previously tracked objects.

$$z_k = Hx_k + v_k \tag{3.6}$$

where:

- $A_{n\times n}$ is the state transition matrix, relating the previous time step $k-1$ to the current state k;
- $B_{n\times m}$ is a control input matrix applied to the optional control input u_{k-1};
- $H_{m\times n}$ is a transformation matrix that transforms the state into the measurement domain;
- w_k and v_k represent the process noise vector with covariance $Q_{n\times n}$ and the measurement noise vector with covariance $R_{m\times m}$, respectively.

Whenever a new tracking instance is computed, the Kalman filter enters the prediction state for each object, where the next position is predicted, along with its associated covariance error, described by equations 3.7 and 3.8, respectively:

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1} \tag{3.7}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{3.8}$$

where:

- $\hat{x}_k^-$ and P_k^- represent the *a priori* state estimate and error covariance matrix, respectively;
- $\hat{x}_{k-1}$ is the previous estimated state or *a posteriori* state;

If the tracked object is associated with a detection in the current frame, the Kalman filter enters its update stage in which the Kalman gain, displayed in 3.9, is computed. The measurement estimate calculation is updated relating the previous measurement estimate, the Kalman gain and the current frame measurement, as presented in 3.10, while the error covariance is determined through 3.11.

$$K_k = P_k^- H^T (HP_k^- H^T + R)^- 1 \tag{3.9}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{3.10}$$

$$P_k = (I - K_k H)P_k^- \tag{3.11}$$

where I is the identity matrix and K is the Kalman Gain.

Almost all maneuvering target tracking methods are model based, in that they assume that the target motions and observations can be mathematically modelled with sufficient accuracy [63]. Based on kinematic equation 3.12, the relation between the position x and velocity $\dot{x}$ of a moving object can be determined as the following:

$$x_k = x_{k-1} + \dot{x}_{k-1}\Delta t \tag{3.12}$$

Since the state vector x_k contains the position and velocity, the system can be modelled according to equation 3.13.

$$x_k = \begin{bmatrix} x_k \\ \dot{x}_k \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ \dot{x}_{k-1} \end{bmatrix} \tag{3.13}$$

For this tracking scenario the input coordinates are 3D, hence, the Kalman matrices were designed as shown in 3.14:

$$A = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

$$R = \begin{bmatrix} \sigma_x & 0 & 0 \\ 0 & \sigma_y & 0 \\ 0 & 0 & \sigma_z \end{bmatrix} \qquad Q = \begin{bmatrix} \frac{\Delta t^4}{4} & 0 & 0 & \frac{\Delta t^3}{2} & 0 & 0 \\ 0 & \frac{\Delta t^4}{4} & 0 & 0 & \frac{\Delta t^3}{2} & 0 \\ 0 & 0 & \frac{\Delta t^4}{4} & 0 & 0 & \frac{\Delta t^3}{2} \\ 0 & \frac{\Delta t^3}{2} & 0 & 0 & \Delta t^2 & 0 \\ 0 & 0 & \frac{\Delta t^3}{2} & 0 & 0 & \Delta t^2 \end{bmatrix} \sigma_a^2 \tag{3.14}$$

where σ_a is the tuning magnitude of standard deviation of the acceleration and $\sigma_x, \sigma_y, \sigma_z$ are the variances of the measurement noise, which were all set to $1e-4$ after experimentation.

3.4.3 Operation of the Tracking Module

The previous sections described the metrics and methods used to associate tracked objects to their new appearances and to predict the objects' behaviour. This section intends to clarify the whole tracking operation by describing the communication between the different tracking sub-modules.

Each tracked object can be defined at frame k by its state vector $S_k = [P_x, P_y, P_z, C_h, B_r]$, however, this state vector has different components in the prediction module. As mentioned 3.4.2, the Kalman model, 3.13, is based on the kinematic equation, 3.12, meaning that the Kalman state vector is $X_k = [P_x, P_y, P_z, V_x, V_y, V_z]$, with P being the position of the object and V its velocity. With the notion that the Kalman filter prediction state is executed synchronously and having access to the position differential, computing the object's velocity is a trivial matter. The other components of S_k are the vessel's physical attributes determined by shape and colour. As those shouldn't change over time, there isn't a need to predict their future state. These two components are taken from the detection state, D_k, the vector containing the data for a single detection.

During a single cycle of the tracking module, n state vectors are created for n detections fed from the detection module. These state vectors, S_k, contain the position vector from the LiDAR projection and the object's characteristics from the image detector. The prediction module updates the position vector of each previously tracked object, feeding this information to the association module which updates the object state with the predicted position and the previously detected visual characteristics (predominant color and bounding box aspect ratio). This vector is called *Object State*. Figure 3.16 depicts the described relationship.

Additionally, the association module measures the Euclidean distance between the *Object State* and the *Detection State*, matching the objects from the same class with the shortest distance, as detailed in 3.4.1.

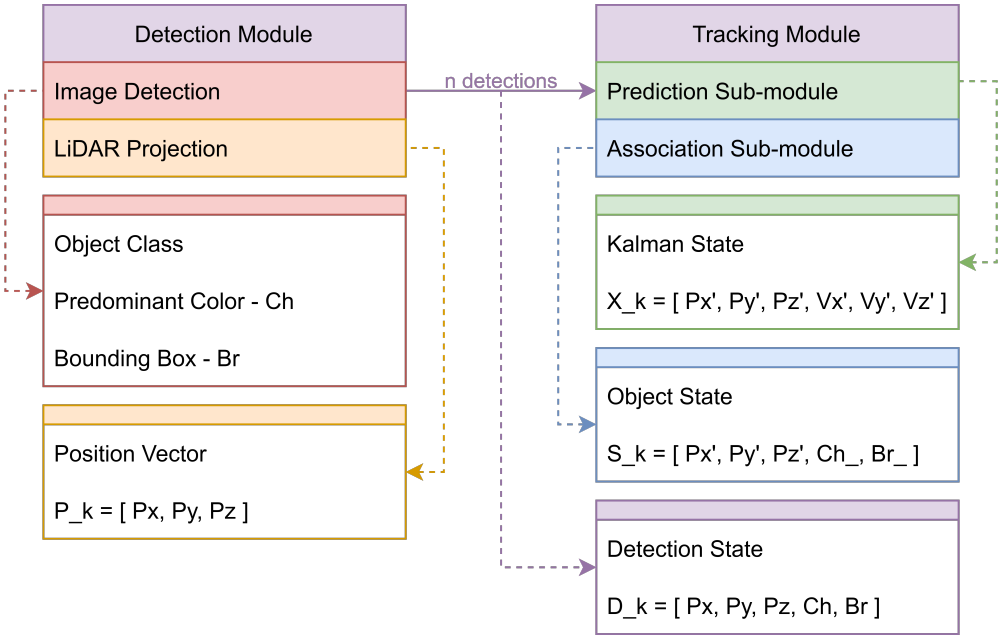


Figure 3.16: Block Diagram of the communication between the detection module and the tracking module for one object.

The *Object State* is then fed into the prediction sub-module to correct its position using the current detected position information, the *Position Vector*, and the position estimation from the *Kalman State*. The predominant color and bounding box aspect ratio values are also updated to match the detector information.

The implemented architecture reads the detections topic from the detection module and publishes a tracking topic with the contents of the tracking list for each tracking cycle (20*ms*). Whenever the detections topic returns an empty list, the association sub-module isn't called. The activity diagram, figure 3.17, illustrates the whole tracking operation.

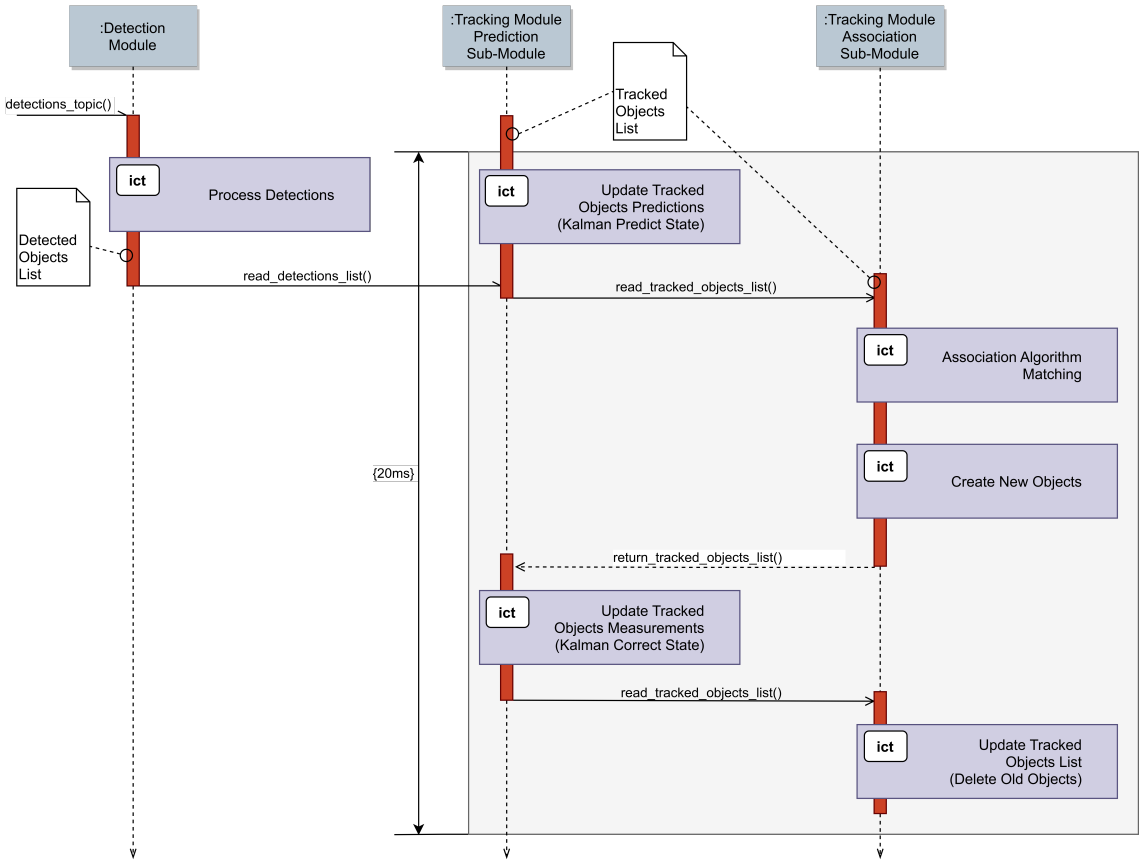


Figure 3.17: Activity diagram of the tracking module. The *ict* process is executed in a single *ROS* node.

Chapter 4

Results

4.1 Training of the Detection Module

The detection module was trained with the combined dataset exhibited in section 3.2. The performance of the trained detector is then measured through its Mean Average Precision (mAP). This section aims to expand on the training process results and the methodology used to measure performance.

4.1.1 Detection Module Performance Metric

The metric used to compare the accuracy of the training exercises was the Mean Average Precision (mAP). This metric calculates the mean of the precision metric (true positive detections over the total positive detections value) for each class of objects in the dataset. In order to compare the detections with the ground truth, the Intersection over Union (IoU) provides a quantitative score based on how concurrent are the areas of the two bounding boxes. It computes the intersection over the union of the two bounding boxes: the bounding box for the ground truth and the predicted bounding box. This process is described in the following equations 4.1, 4.2, 4.3, 4.4 and 4.5.

$$TL = \begin{bmatrix} \max(d_{TL_x}, o_{TL_x}) & \max(d_{TL_y}, o_{TL_y}) \end{bmatrix} \tag{4.1}$$

$$BR = \begin{bmatrix} \min(d_{BR_x}, o_{BR_x}) & \min(d_{BR_y}, o_{BR_y}) \end{bmatrix} \tag{4.2}$$

$$A_{intersection} = (BR_x - TL_x) * (BR_y - TL_y) \tag{4.3}$$

$$A_{union} = A_{d_{box}} + A_{o_{box}} - A_{intersection} \tag{4.4}$$

$$P_{IoU} = \frac{A_{intersection}}{A_{union}} \tag{4.5}$$

where TL and BR are the pixel coordinates of the two respective edges of the yellow box in 4.1, which represents the intersection area between the detected object, d and the ground truth object o, with $n \in \mathrm{N}$, and A represents the areas, $box_{width} * box_{height}$.

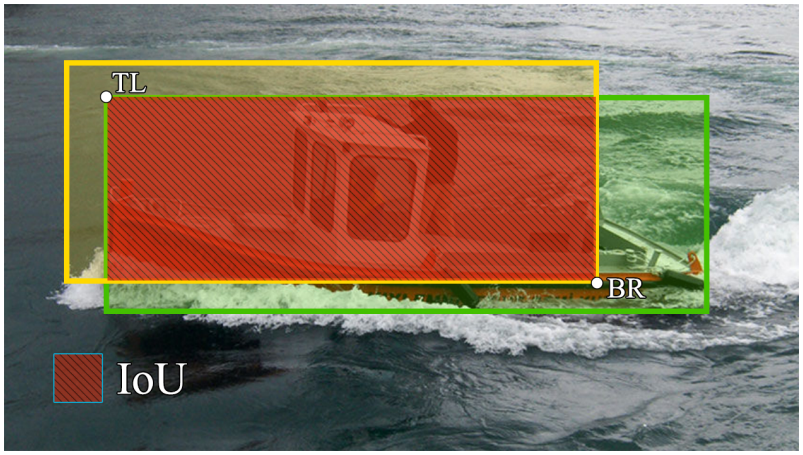


Figure 4.1: Visual demonstration of the IoU algorithm enunciated in 4.1 to 4.5. The green box represents the ground truth, the yellow box represents the predicted area and the orange box represents the intersection area. If the yellow and green boxes completely overlap, the detection is perfect and the IoU value is 1. When both boxes are completely separate, the orange box does not exist, making the $IoU = 0$.

Defining an IoU threshold will provide a metric to classify the object detection in three ways:

1. False Negative (TN) - when a ground truth is present in the image but the model failed to detect the object;
2. True Positive (TP) - if $IoU \geq threshold$;
3. False Positive (FP) - if $IoU < threshold$.

The confidence score for each detected object is also considered, calculating the precision and recall values for confidence intervals of 10%, according to equations 4.6 and 4.7, and including $\varepsilon = 1x10^{-}6$ in the numerator and denominator for linearity purposes and to avoid divisions by zero respectively.

$$Precision = \frac{TP}{TP + FP} \tag{4.6}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.7}$$

For each image, the algorithm will match each ground truth instance with the prediction with the best candidate IoU score and count it as a TP until there aren't any more possible matches (i.e. there aren't any more predictions with enough accuracy to satisfy the IoU threshold or every object already has one associated prediction). The leftover ground truth are counted as FN while the remaining predictions belong to the FP. This process is computed for each confidence interval

and also for each class, resulting in a matrix of $confidence \times class$ with the calculated precision and recall values for each element.

The average precision for each class is then estimated by calculating the area under the plot of the recall/precision graph for all confidence intervals, with the mean average precision being the sum of the average precision for each class, divided by the number of classes, as depicted in equation 4.8. Figure 4.2 better illustrates the average precision calculation.

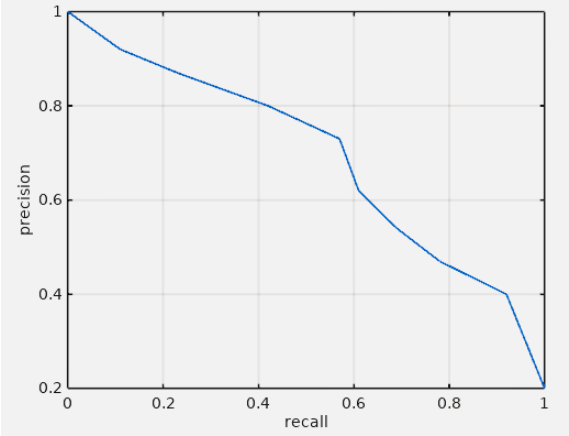


Figure 4.2: Typical plot of the recall/precision values for each confidence iteration. The area below the line is the average precision.

$$mAP = \frac{\sum_{i=0}^{Classes} AP_i}{Classes} \tag{4.8}$$

Another popular performance comparison metric involves iterating the mAP for an interval of IoU thresholds, typically iterations of $0.05@[0.5,1]$. For this, the aforementioned algorithm is run for each IoU threshold iteration, computing the weighted average, $IoU\_threshold \times AP_{class}$, per class before estimating the mAP.

4.1.2 Preliminary Training Results

A training exercise with the same conditions (batch size, image size, number of epochs) was made for each optimiser, varying only the initial learning rate, l_{r0}, and momentum parameters, each for 3 arbitrary values. The training results are then plotted and compared to determine which optimiser works best for the provided dataset and which parameters potentiate the performance of the best optimiser.

For the first training exercise, only the mAP with IoU of more than 50% was considered. For the second and third exercises the mAP[.5:.95], which iterates the average of the mAP with IoU bigger than 50% in intervals of 5%, is also included. Measurements of the mAP are displayed in table 4.1. Overall, the Adam and Adadelta optimisers achieved consistent over 80% mAP@.5 with learning rates of 0.001 and 0.1 respectively. With only one result ($lr0 = 0.05$ and $momentum = 0.8$) achieving similar performance, the SGD optimiser was deemed less fit for this specific training scenario.

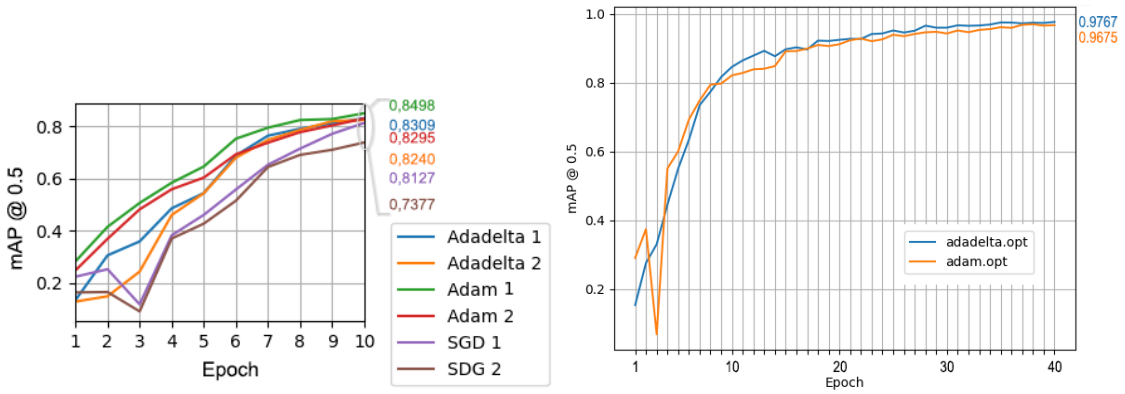
Table 4.1: Measured mAP@.5 final results using the Adadelta, Adam and SGD optimisers. Best results in bold.

Adadelta		Momentum		
		0.8	0.95	0.5
lr0	0.05	0.7249	0.7538	0.6611
	0.001	0.0945	0.1388	0.0782
	0.1	**0.8309**	0.824	0.8114

Adam		Momentum		
		0.8	0.95	0.5
lr0	0.05	1.010e-5	0.2769	4.510e-5
	0.001	**0.8498**	0.8295	0.8165
	0.1	0.1166	0.1033	4.170e-5

SGD		Momentum		
		0.8	0.95	0.5
lr0	0.05	**0.8127**	0.3363	0.7377
	0.001	0.4025	0.6973	0.2738
	0.1	0.3604	0.4150	0.4995

Out of a total of 27 training exercises, the top 2 of each optimiser are plotted in figure 4.3 (a). Although the best result was achieved with the Adam optimiser with the parameters $lr0 = 0.001$ and $momentum = 0.8$, the top three results come very close. Furthermore, the training wasn't extensive enough to stale out the evolution curves. While the SGD optimiser is clearly inferior, it is still hard to tell whether Adam or Adadelta would come out on top. To rule out the uncertainties, a second training exercise was done, testing out the best $lr0$ and *momentum* combinations for each of the top two results.



(a) Plotted mAP@.5 of all training exercises.

(b) Plotted mAP@.5 of the two best results.

Figure 4.3: Mean Average Precision of detections with IoU over 0.5. The figure (a) includes every 10 epoch training exercise while the figure (b) has a comparison of the two best results (a), trained for another 30 epochs.

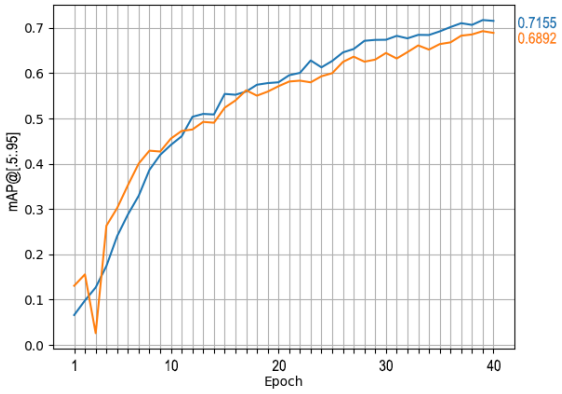
Figure 4.3 (b) shows the results of 40 epochs of training. Adadelta ended up coming on top after the 10th epoch, remaining superior through further stages. This excercise achieved a 17.5% increase in mAP@.5 compared to the previous best Adadelta result, while managing a 14.9% boost with the Adam optimiser. Both values, however, are awfully close with a less than 1% difference,

Table 4.2: Mean Average Precision comparison with the state-of-the-art for a single boat class.

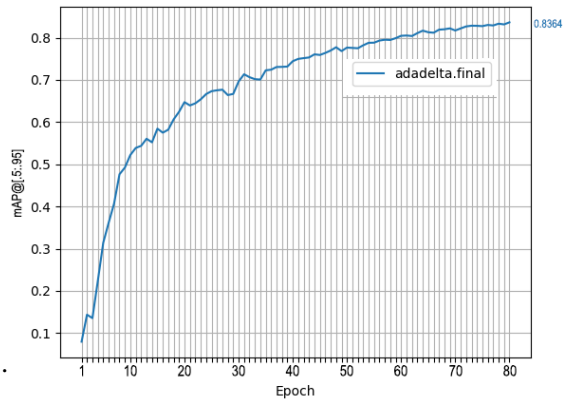
	mAP @.5	mAP @ [.5 .95]
Pre-trained YOLOv4	0.812	0.643
Our model	0.998	0.928

meaning that there isn't a significant difference between detections with both optimisers.

In order to better compare Adam and Adadelta, the evolution of the mAP[.5:.95] metric was observed throughout the training process. This metric better exhibits the quality of the detections, meaning that higher values point to detections with a higher IoU. Figure 4.4 (a) displays the evolution of the mAP[.5:.95] metric over epochs, illustrating a 3.8% difference between the two optimisers, having the Adadelta optimiser with momentum of 0.8 and $lr_0 = 0.1$ achieve the best result.



(a) Comparison between the optimal Adadelta and Adam optimised configurations.



(b) Densely trained Adadelta setting.

Figure 4.4: Evolution of the mAP[.5:.95] metric.

Furthermore, in order to explore the full potential of the Adadelta configuration, a third training instance was done until the mAP@.5 evolution stabilized as depicted in figure 4.4. Through this last training, the mAP@.5 values did not increase further than 1%. However, the mAP[.5:.95] improved 16.9% over the last 40 epochs.

4.1.3 Transfer Learning Results

The open-source YOLOv4 detector includes a pre-trained model on the COCO dataset. This model has been trained for 80 classes, one of them being boats. In order to fairly measure the improvement with the transfer learning, the classes of the objects in the combined dataset were suppressed so both models are tested for only one class of objects: boats. The results for this first scenario are displayed in table 4.2, showing a 22.9% and 44.3% increase in the mAP with an IoU threshold of 50% and the mAP@[.5 .95] respectively.

With the previous results confirming the effectiveness of the transfer learning technique and a clear improvement in boat detection compared to a multi-purposed state-of-the-art detector, another training instance has been executed, for the collected dataset with its defined nine classes. This resulted in a class mAP@[.5 .95] of 89.5%, achieving a detector capable of detecting and classifying different kinds of maritime vessels while compromising 3.7% in average precision. Figure 4.5 shows the mAP[.5 .95] values per class.

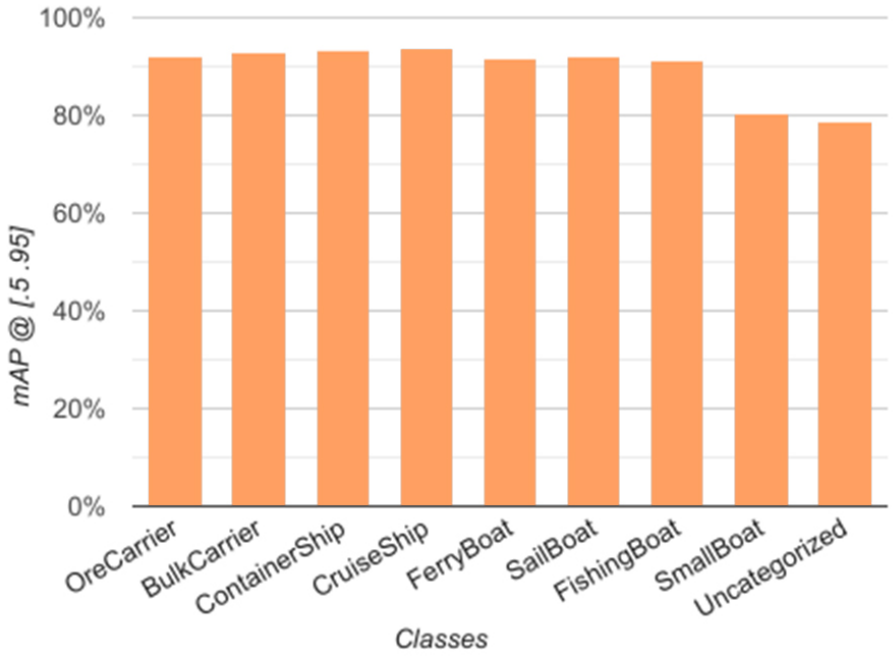


Figure 4.5: Mean Average Precision per class, with 10 increments of 5% of the IoU threshold, starting at 50%.

4.2 Object Tracking Performance

The basic idea behind multiple object tracking is to assign unique tracking ID's to every object presented in a frame and try to maintain the same ID in subsequent frames associating objects across time. The research done for the state-of-the-art highlighted the MOT metrics as the primary performance indicators when evaluating tracking algorithms 2.3. Such metrics would classify objects between frames, determining whether they are missed detections, false positives or mismatch errors by comparing the results of the tracking algorithm with the ground truth information of a testing scenario. The MOT performance indicators would not only provide an accurate and numeric display of the tracking results, but also enable performance comparisons with other known trackers whose open-source development is compatible with these tests.

With the developed multiple object tracking model, evaluating the performance through the described metrics and comparing it to other trackers isn't feasible due to the following reasons:

- The MOT metrics focus exclusively on the tracking performance with camera input. This tracking model heavily depends on the LiDAR position information and discards any camera detection without sufficient LiDAR data to measure the relative distance of the object to the observer, considering those detections to be false positives from the detection module.
- Such metrics can only be computed when there is sufficient testing data with ground truth information. The Singapore Maritime Dataset, studied in section 3.2, may provide video with ground truth information but such information is also exclusively camera input, hence, not useful as testing data for the developed tracking model. In order to work with this tracker, both camera video frames and LiDAR point cloud frames need to be captured.

4.2.1 Tracking Results

The collected data from the aforementioned missions was vital for exhaustive testing of the developed tracking algorithm, however, not sufficient for proper measurement of the tracking performance as the only method for obtaining ground truth would be similar to the one described in section 3.2, which would not be feasible considering the amount of frames and time dedicated for the development of this project. Without the ability to resort to the MOT performance metrics, the performance can only be evaluated through footage observation, considering the objects with sufficiently accurate detections and determining whether the tracking module is associating objects between frames correctly or predicting their trajectory when the detections aren't consistent.

For this, a simple visualization tool was built, drawing a bounding box around the tracked objects and identifying each instance with a numerical tag. The box is colored according to the object's class, turning gray whenever the tracking isn't active (the object wasn't detected in the current frame).

The considered metrics are as follows:

- Tracked Objects - Number of different objects tracked;
- Average Association Accuracy - The association accuracy of each object is measured by counting the frames where the same tracking identifier is attributed to a detected object, divided by the total number of frames where the object is visible in the footage. The average association accuracy computes the sum of the association accuracy for each tracked object, divided by the number of tracked objects;
- Best Performing Object - This metric displays the object with the highest association accuracy;
- Successful Re-associations - The number of occurrences where an object becomes inactive due to not being detected at frame t (i.e occluded or outside of the field-of-view), being correctly re-identified with the same tracking identifier when it is detected $t+n$;
- Lost Objects - Inactive objects that are re-identified with a different tracking identifier when detected at a posterior frame.

Table 4.3: Tracking performance results.

Video Name	Tracked Objects	Average Association Accuracy (%)	Best Performing Object [ID](%)	Successful Re-Associations	Lost Objects
leixoes_4	20	68,52	[11] 95,42	429	9
leixoes_5	38	42,89	[4] 76,31	519	20
leixoes_7	10	17,32	[9] 99,89	81	9
leixoes_8	11	40,64	[1] 81,49	25	11
leixoes_9	13	12,81	[4] 31,05	107	13
leixoes_12	27	10,74	[8] 46,44	183	27
leixoes_15	15	19,48	[1] 77,42	189	13
leixoes_16	38	36,00	[1] 83,30	1084	35
viana_1	26	22,99	[1] 73,91	106	24
viana_2	14	18,78	[2] 54,32	67	14
viana_3	37	23,86	[23] 49,05	282	35
viana_5	41	23,64	[6] 52,27	293	39

From the analysed data, a package of twelve videos recording the output of the tracking module was assembled and made available [1]. Table 4.3 displays the aforementioned metrics for each recorded video.

As expected, whenever the tracking module is consistently fed accurate detections, the association finds some degree of success, being able to successfully identify the same object throughout the footage. Good examples of acceptable tracking performance can be viewed in object 1 from *leixoes_15* and object 9 from *leixoes_7*. The figure 4.6 displays multiple frames from *leixoes_15* where object 1 is accurately tracked from different perspectives. The association task becomes harder and less accurate when multiple objects of the same class are detected simultaneously. Video *leixoes_4* exemplifies association issues where the same tracking identifier is associated with different vessels next to each other. Possible solutions to address this issue, such as using IMU and GPS data to obtain the location and orientation of the observer or improving the prediction sub-module by including bounding box coordinates in the Kalman filter model, are listed in the following chapter.

Further analysis of the footage and the performance measurements, the developed tracking module clearly has its limitations and large room for improvement. Poor detection performance can greatly hinder the tracking performance. When the extracted features fail to accurately de-

[1] https://drive.google.com/drive/folders/1pYmqCzDDe_elRnT9nt8PrQC_VfUZvz4W?usp=sharing

(a) Initially detected from a rear perspective, while cornering left.

(b) Successfully associated while concurrent with another object.

(c) Successful association of the same vessel from a left-sided perspective.

(d) Returned to a rear perspective, being tracked for longer than 2 minutes.

Figure 4.6: Successful tracking of object 1 in video *leixoes_15*.

scribe the object, it can lead to wrong associations. Examples of poor detection performance are given in figure 4.7. To circumvent this issue, a tracked object is deleted whenever the number of inactive frames is bigger than the number of active frames (frames where it was identified in the detector). This approach has the drawback of re-identifying not only the intended miss detections, but also some successful detections with poor confidence (i.e. out of 60 frames, only 2 have accurate detections), as evidenced in figure 4.8.

Figure 4.7: Examples of poor detection performance evident in video *leixoes_12*, where rocks are detected as *SmallBoat* and a flag pole as a *SailBoat*.

Figure 4.8: Example of a failed re-identification from video *leixoes_7* where the same *SmallBoat* is associated with tracking identifiers 2 and 3.

4.2.2 Hardware Performance

Development of this tracking model focused on the ability to run in real-time while resorting to hardware that can be installed on-board of an ASV in order to enable future real-world testing and application. The bottleneck of the whole architecture regarding processing power happens during the computation of the detections for each image frame. As such, the detection and tracking modules run separately, allowing for the tracking algorithm to process asynchronous detections and to be compatible with different faster detection modules. Each tracking instance, including the execution of both prediction and associated modules, as described in figure 3.17, has an execution time of $20ms$ or $50Hz$ per tracking instance, updating at least 100 objects simultaneously. For this maritime application, such performance should be plentiful.

The hardware specs in which this model was tested are listed as follows:

- CPU - *Intel Core i5-6300HQ* $2.30GHz$;
- GPU - *nVidia Corporation GeForce GTX 960M* - $2GB$ VRAM (of which $1900MB$ are used for processing the detection module);
- RAM - $16GB$ (of which $300MB$ are used for processing the tracking module).

Chapter 5

Conclusions & Future Work

This dissertation proposes a complete tracking system for detecting different classes of maritime vessels and tracing their trajectory over time with the aim of fulfilling an important requirement for autonomous navigation at sea. The developed system encompasses an image-based CNN object detector trained trough transfer learning and a tracking model with two distinct sub-modules for association and trajectory prediction. The association sub-module measures the Euclidean distance between feature vectors, associating the nearest candidates while the trajectory prediction is done by resorting to state estimation of an object's relative position coordinates with a Kalman filter.

The lack of available data for training of the detection module required time-consuming research for usable maritime datasets. Within the scope of this dissertation, a maritime dataset is collected to provide a maritime dataset with real-time images with nine different types of vessels properly identified and classified, used for the transfer learning of the detector module. It involved a re-classification of data from available online datasets and footage acquisition and manual annotation of vessels in thousands of pictures. With this, a clear improvement in boat detection compared to a multi-purposed state-of-the-art detector, YOLO-v4, with a 22.9% and 44.3% increase in the mAP with an IoU threshold of 50% and the mAP@[.5 .95] respectively.

Testing and development of the tracking module was done resorting to acquisition of data from navigating the SENSE ASV around different vessels in the Leixões and Viana do Castelo ports. Initially, this stage was thought to be done resorting to simulated data, considering the accessibility to a well simulated environment of the SENSE ASV in the *Gazebo* simulator. However, populating the simulated environment with distinct 3D models of different vessels wasn't feasible. There were issues and incompatibilities with the hardware used in development of this dissertation, failing to produce a sufficient simulation. As such, the whole development stage had to be executed resorting solely to the aforementioned acquired data in a real environment.

The tracking module was able to continuously track vessels in real-time whenever detections are consistent and accurate, but future work is necessary to achieve a level tracking performance suitable for fully autonomous navigation. Regardless, the developed system achieved good results when the module is consistently fed accurate detections, being able to successfully identify the same object throughout the footage.

Furthermore, the lack of ground truth in the footage used for testing the tracking module dismissed relevant comparisons with other tracking modules and further analysis which could aid development as it isn't possible to determine whether the trajectory prediction is correct without real information about an object's trajectory. Future work can be conducted, such as the following:

- Collection of a bigger maritime dataset with annotated images of maritime vessels enabling further development of the detector module;
- Development of a complete 3D simulated environment with accurate simulations of different vessels to conduct experimental tests of the developed system with different navigation instructions, allowing for ground truth information vital for performance measurements of the tracking module;
- Acquisition of data with ground-truth information for testing and further developing the tracking module;
- Using IMU and GPS data to obtain the location and orientation of the observer. This step would allow for determining the absolute location of the vessels to be tracked, minimizing miss-associations where the relative location is heavily dependant on observer movement;
- Improving the detection module classification and location information through clustering of the LiDAR to extract more features from objects such as shape and size;
- Improving the prediction sub-module by including bounding box coordinates in the Kalman filter model.

References

[1] N. Baird, "Fatal ferry accidents, their causes, and how to prevent them," 2018.

[2] A. E. Branch, "Economics of Shipping Practice and Management," 1988.

[3] R. J. Silva, P. Nuno Leite, and A. M. Pinto, "Multi-agent optimization for offshore wind farm inspection using an improved population-based metaheuristic," in *2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pp. 53–60, 2020.

[4] N. Cruz, A. Matos, S. Cunha, and S. O. d. Silva, "Zarco - an autonomous craft for underwater surveys," *7th Geomatic Week*, 2007.

[5] D. F. Duarte, M. I. Pereira, and A. M. Pinto, "Multiple vessel detection and tracking in harsh maritime environments," in *OCEANS 2021: San Diego – Porto*, pp. 1–5, 2021.

[6] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2, pp. 1150–1157, 1999.

[7] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-Up Robust Features (SURF)," *Computer Vision and Image Understanding*, vol. 110, pp. 346–359, June 2008.

[8] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[9] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[10] P. Voigtlaender, M. Krause, A. Osep, J. Luiten, B. B. G. Sekar, A. Geiger, and B. Leibe, "Mots: Multi-object tracking and segmentation," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 7934–7943, 2019.

[11] P. Bergmann, T. Meinhardt, and L. Leal-Taixe, "Tracking without bells and whistles," in *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2019-Octob, pp. 941–951, 2019.

[12] L. Tan, X. Dong, Y. Ma, and C. Yu, "A Multiple Object Tracking Algorithm Based on YOLO Detection," in *Proceedings - 2018 11th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics, CISP-BMEI 2018*, 2019.

[13] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, "Handwritten digit recognition with a back-propagation network," in *IJCNN-91-Seattle International Joint Conference on Neural Networks*, pp. 396–404, 1990.

[14] S. Murray, "Real-Time Multiple Object Tracking A Study on the Importance of Speed," *arXiv*, 2017.

[15] A. Giffin and R. Urniezius, "The Kalman filter revisited using maximum relative entropy," *Entropy*, vol. 16, no. 2, pp. 1047–1069, 2014.

[16] B. Sahbani and W. Adiprawita, "Kalman filter and Iterative-Hungarian Algorithm implementation for low complexity point tracking as part of fast multiple object tracking system," in *2016 6th International Conference on System Engineering and Technology (ICSET)*, pp. 109–115, 2017.

[17] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, December 2016.

[18] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9905 LNCS, pp. 21–37, 2016.

[19] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal Loss for Dense Object Detection," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, pp. 318–327, 2020.

[20] Z. Zhao, P. Zheng, S. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, pp. 3212–3232, Nov. 2019.

[21] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," 2014.

[22] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," 2014.

[23] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[24] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," 2018.

[25] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8693 LNCS, pp. 740–755, 2014.

[26] A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," 2020.

[27] C. Y. Wang, A. Bochkovskiy, and H. Y. M. Liao, "Scaled-yolov4: Scaling cross stage partial network," 2020.

[28] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 6517–6525, 2017.

[29] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," in *arXiv*, 2018.

[30] "Efficientdet: Scalable and efficient object detection," 2020.

[31] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," 2019.

[32] X. Long, K. Deng, G. Wang, Y. Zhang, Q. Dang, Y. Gao, H. Shen, J. Ren, S. Han, E. Ding, and S. Wen, "Pp-yolo: An effective and efficient implementation of object detector," 2020.

[33] A. Milan, L. Leal-Taixe, I. Reid, S. Roth, and K. Schindler, "MOT16: A Benchmark for Multi-Object Tracking," 2016.

[34] N. L. Baisa, "Occlusion-robust Online Multi-object Visual Tracking using a GM-PHD Filter with a CNN-based Re-identification," *arXiv*, 2019.

[35] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *Proceedings - International Conference on Image Processing, ICIP*, vol. 2016-Augus, pp. 3464–3468, 2016.

[36] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," *Proceedings - International Conference on Image Processing, ICIP*, vol. 2017-Septe, pp. 3645–3649, 2018.

[37] D. Freire, J. Silva, A. Dias, J. M. Almeida, and A. Martins, "Radar-based target tracking for obstacle avoidance for an autonomous surface vehicle (asv)," in *OCEANS 2019 - Marseille*, pp. 1–6, 2019.

[38] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pp. 226–231, 1996.

[39] C. S. Tan, M. R. Arshad, and R. Mohd-Mokhtar, "River navigation system using Autonomous Surface Vessel," *USYS 2016 - 2016 IEEE 6th International Conference on Underwater System Technology: Theory and Applications*, pp. 13–18, 2017.

[40] Y. Cho, J. Park, M. Kang, and J. Kim, "Autonomous detection and tracking of a surface ship using onboard monocular vision," *2015 12th International Conference on Ubiquitous Robots and Ambient Intelligence, URAI 2015*, pp. 26–31, 2015.

[41] E. Rosten, R. Porter, and T. Drummond, "Faster and better: A machine learning approach to corner detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 1, pp. 105–119, 2010.

[42] D. Qiao, G. Liu, J. Zhang, Q. Zhang, G. Wu, and F. Dong, "M3c: Multimodel-and-multicue-based tracking by detection of surrounding vessels in maritime environment for usv," *Electronics*, vol. 8, no. 7, 2019.

[43] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

[44] M. Everingham, L. Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *Int. J. Comput. Vision*, vol. 88, p. 303–338, June 2010.

[45] A. R. Gaspar, A. Nunes, A. M. Pinto, and A. Matos, "Urban@cras dataset: Benchmarking of visual odometry and slam techniques," *Robotics and Autonomous Systems*, vol. 109, pp. 59–67, 2018.

[46] P. N. Leite and A. M. Pinto, "Exploiting motion perception in depth estimation through a lightweight convolutional neural network," *IEEE Access*, vol. 9, pp. 76056–76068, 2021.

[47] D. Duarte, M. I. Pereira, and A. M. Pinto, "An annotated and classified maritime dataset aimed at machine learning," 2022.

[48] Z. Shao, W. Wu, Z. Wang, W. Du, and C. Li, "Seaships: A large-scale precisely annotated dataset for ship detection," *IEEE Transactions on Multimedia*, vol. 20, no. 10, pp. 2593–2604, 2018.

[49] D. K. Prasad, D. Rajan, L. Rachmawati, E. Rajabally, and C. Quek, "Video processing from electro-optical sensors for object detection and tracking in a maritime environment: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 8, pp. 1993–2016, 2017.

[50] F. J. P. Montalbo, "A computer-aided diagnosis of brain tumors using a fine-tuned yolo-based model with transfer learning.," *KSII Transactions on Internet & Information Systems*, vol. 14, no. 12, 2020.

[51] J.-a. Kim, J.-Y. Sung, and S.-h. Park, "Comparison of faster-rcnn, yolo, and ssd for real-time vehicle type recognition," in *2020 IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia)*, pp. 1–4, 2020.

[52] Q. Yang, Y. Zhang, W. Dai, and S. Pan, *Transfer Learning*. Cambridge University Press, 2020.

[53] L. Perez and J. Wang, "The effectiveness of data augmentation in image classification using deep learning," 2017.

[54] V. Subramanian, *Deep Learning with PyTorch: A practical approach to building neural network models using PyTorch*. Packt Publishing, 2018.

[55] M. I. R. Pereira, "A machine learning approach for predicting docking-based structures," Master's thesis, Faculdade de Engenharia da Universidade do Porto, 2020.

[56] M. D. Zeiler, "Adadelta: An adaptive learning rate method," 2012.

[57] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[58] J. Kiefer and J. Wolfowitz, "Stochastic Estimation of the Maximum of a Regression Function," *The Annals of Mathematical Statistics*, vol. 23, no. 3, pp. 462 – 466, 1952.

[59] P. N. Leite and A. M. Pinto, "Exploiting motion perception in depth estimation through a lightweight convolutional neural network," *IEEE Access*, vol. 9, pp. 76056–76068, 2021.

[60] P. Leite, R. Silva, A. Matos, and A. M. Pinto, "An hierarchical architecture for docking autonomous surface vehicles," in *2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pp. 1–6, 2019.

[61] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521623049, 2000.

[62] P. N. B. Leite, "A self-guided docking architecture for autonomous surface vehicles," Master's thesis, Faculdade de Engenharia da Universidade do Porto, 2019.

[63] X. Rong Li and V. Jilkov, "Survey of maneuvering target tracking. part i. dynamic models," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, no. 4, pp. 1333–1364, 2003.