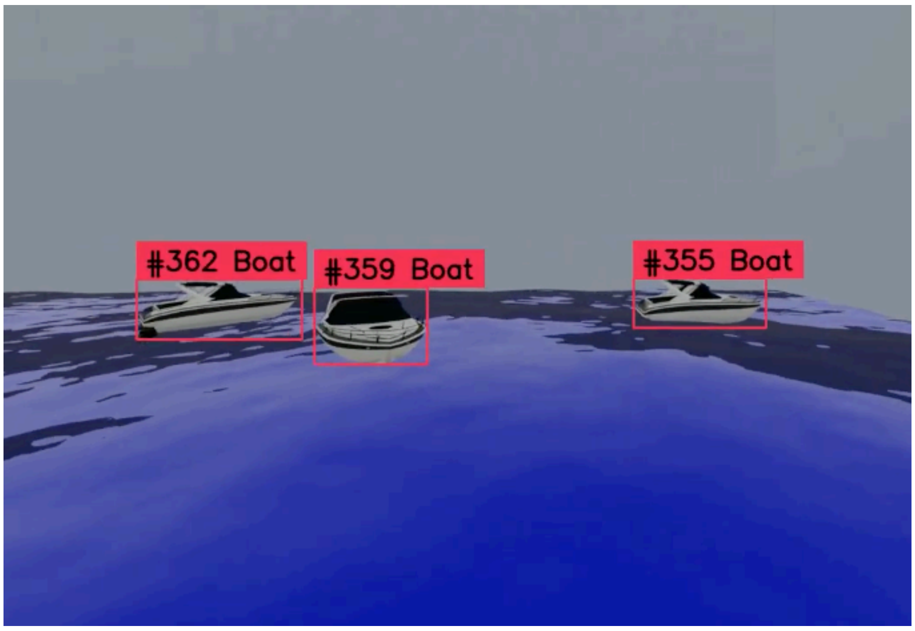
Emil Wiersdalen Hjelle
Sondre Molnes Kanstad

An Approach to First-Level Situation Awareness for Autonomous Surface Vehicles

Master's thesis in Informatics
Supervisor: Rudolf Mester
June 2023



Emil Wiersdalen Hjelle
Sondre Molnes Kanstad

An Approach to First-Level Situation Awareness for Autonomous Surface Vehicles

Master's thesis in Informatics
Supervisor: Rudolf Mester
June 2023

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

Abstract

There has been extensive research on how to recognize different objects in images, but fewer research articles explain how to utilize the information from these detections. For autonomous vessels to be safe and efficient, it is crucial that the data they receive accurately represent their surroundings. By employing a system capable of extracting information about the position of boats or obstacles in the surrounding environment, the situational awareness of the autonomous vessel is enhanced. In order to achieve first-degree situational awareness in the autonomous vessel, several aspects must be implemented, including object detection, tracking, and localization.

The project began by visualizing simulated data used to experiment and evaluate various methods. The visualization was created in Blender and aimed to replicate a busy harbour with boats. In the architecture of the system, established methods for object detection and tracking were combined with fundamental methods for estimating depth, bearing, and, subsequently, position. A YOLOv8 model was trained to detect boats in the video sequences generated in Blender. During the model's training, datasets were also created on which the model was trained. Furthermore, the tracking algorithm ByteTrack was used to track and identify the various detections over time. After the boats were detected, the detections were utilized to estimate the angle, depth, and, ultimately, the position of the respective boats. These estimates were made based on the detection from a stereo camera setup generated in Blender. In order to enhance the interpretability of the results of the situational awareness system, the outcomes were visualized from a bird's-eye view with the system at the centre. These visualizations were generated to analyze the end-to-end system and generate videos qualitatively.

By utilizing state-of-the-art methods in object detection and tracking, as well as fundamental techniques for bearing and depth estimation, the maritime situational awareness system has demonstrated a high level of accuracy in representing its dynamic surroundings. The situational awareness system achieved a precision rate of 63% with an error margin of 6 meters for detected objects throughout the track. However, the system struggles with complex situations, such as occluded and partially visible ships. The links to the tracking videos and the situation awareness system's predictions can be found in chapters 3 and 4.

Sammendrag

Det har blitt forsket mye på hvordan man kan gjenkjenne ulike gjenstander i bilder, men det finnes færre forskningsartikler som forteller hvordan man kan bruke informasjonen fra disse gjenkjennelsene. For at autonome fartøy skal være trygge og effektive er det viktig at dataen den får inn representerer omgivelsene presist. Ved hjelp av et system som klarer å ekstrahere informasjon om posisjonen til båter eller hindringer i omgivelsene øker situasjonsbevisstheten til det autonome fartøyet. For at det autonome fartøyet skal nå det første nivået av situasjonsbevissthet er det flere aspekter som må implementeres blant annet objekt deteksjon, sporing og lokalisering.

Prosjektet startet med å visualisere simulert data som ble brukt til å eksperimentere og evaluere forskjellige metoder. Visualiseringen ble laget i Blender og skal imitere en travel havn med båter. I arkitekturen til systemet blir etablerte metoder for objekt deteksjon og sporing kombinert med grunnleggende metoder for å estimere dybde, vinkel og deretter posisjon. Det ble trent en YOLOv8 modell for å oppdage båter i video sekvensene som var generert i Blender. I forbindelse med treningen av modellen ble det også laget datasett som modellen trente på. Videre ble sporingsalgoritmen ByteTrack brukt til å spore de ulike deteksjonene og identifisere dem over tid. Etter at båter har blitt detektert så blir deteksjonene brukt til å estimere vinkelen, dybden og, til slutt, posisjonen til de ulike båtene. Disse estimatene har blitt gjort basert på detekteringen fra et stereokameraoppsett generert i Blender. For å forbedre tolkbarheten av resultatene til situasjonsbevissthetsystemet ble resultatene visualisert i et fugleperspektiv med systemet i sentrum. Disse visualiseringene ble brukt til å kvalitativt analysere systemet ende-til-ende og generere videoer.

Ved bruk av de siste og beste metodene innenfor objekt deteksjon og sporing samt grunnleggende metoder for vinkel- og dybde estimering har det maritime situasjonsbevissthet systemet vist en høy grad av nøyaktighet for å representere sine dynamiske omgivelser. Situasjonsbevissthetsystemet hadde en treffsikkerhet på 63% med en feilmargin på 6 meter for detekterte objekter for hele sporet. Derimot, sliter systemet ved komplekse situasjoner som tildekkete og delvis synlige fartøy. Lenkene til sporingsvideoene og situasjonsbevissthetssystemets prediksjoner kan bli funnet i kapittel 3 og 4.

Acknowledgements

Firstly, we want to thank our supervisor, Rudolf Mester, for all the feedback and time he spent helping us with the thesis. We would also like to thank our fellow master's student, Henrik Fjellheim, for providing us with the pose files we used for our project. Finally, we would like to thank Zeabuz for the collaboration in the initial phase of the project.

Table of Contents

List of Figures

List of Tables

Code Listings

List of Acronyms

2D two-dimensional. 10, 22, 24, 76

3-DOF three degrees of freedom. 27, 29, 35

3D three-dimensional. iv, viii, 10, 11, 22, 27, 29, 32, 33, 76

6-DOF six degrees of freedom. 11, 27

AABB Axis-Aligned Bounding Box. vi, viii, x, 7, 8, 10, 11, 17, 26, 37, 38, 41, 44–48, 51, 54, 69, 71, 75

AP Average Precision. 19

ASV autonomous surface vehicle. iv–viii, 1–4, 6, 7, 11, 12, 16, 19, 21–29, 35–37, 41, 43, 44, 54, 57, 59, 63–65, 67, 69–71, 74, 76, 77

BEV bird's-eye view. vi, viii, ix, 3, 26, 55–60, 65, 68, 71

CLI command-line interface. 9

CNN Convolutional Neural Network. iv, 8, 9, 11

COLREGs Convention on the International Regulations for Preventing Collisions at Sea. 5

FOV field of view. vi, viii, 31, 32, 44–46, 53, 54, 56, 57, 62, 63, 67, 69, 71, 73

IMU Inertial Measurement Unit. 22

IoU Intersection over Union. v, vi, viii, x, 16–19, 42, 44, 45, 51, 64

mAP Mean Average Precision. v, viii, 19, 38, 39

ML machine learning. 21, 24

MS COCO Microsoft Common Objects in Context. 23, 37, 38

MSE mean squared error. v, 16

MSFBD Maritime Sensor Fusion Benchmark Dataset. 22, 45

NED North, East, Down. vi, viii, 43

NTNU Norwegian University of Science and Technology. 22, 28

ReID re-Identification. 12, 16, 42, 75

RMSE root-mean-square error. x, 16, 47, 62, 63

SA situation awareness. iv–viii, 1–7, 11, 19, 26–29, 35–37, 41, 43–45, 52, 54, 55, 57, 59, 60, 63–65, 67, 69–71, 73–77

SDG synthetic data generator. v, vi, viii, ix, 30, 32–35, 37, 38, 41, 42, 44, 51, 57–59, 64, 65, 68, 71, 74, 76

SYNTHIA SYNTHetic collection of Imagery and Annotations of urban scenarios. 24

YOLO You Only Look Once. i, ii, iv–vi, viii, 3, 9–11, 19, 26, 36–39, 41, 48, 64, 67

Chapter 1

Introduction

This thesis is about a first-level situation awareness (SA) system for an autonomous surface vehicle (ASV). This chapter begins by discussing the motivation and background behind this project, followed by a description of the project. Next, it outlines the scope and limitations of the thesis. Finally, it summarizes the main contributions of the thesis.

1.1 Background and Motivation

1.1.1 Overview of ASVs and their Applications for Society

An ASV is defined as a floating vessel that operates with varying degrees of human control or interaction, ranging from remote-controlled to fully autonomous ships. Due to rapid improvements in artificial intelligence, the interest in research of ASVs has increased due to the possible improvements of safety and efficiency in the shipping industry [Bai+22]. Furthermore, other commercial and research domains, such as the use of ASVs for public transport, are flourishing and have achieved huge milestones, such as the launch of the world's first commercial self-driving passenger ferry [dmb23]. Increasing safety within the maritime domain is crucial since a significant part of accidents at sea is due to human error. Almost 60% of accidents from 2014 to 2021 were caused by human action [Age22]. Well-functioning autonomous ships or decision support systems could reduce the number of accidents and the impact of them.

1.1.2 The Importance of Perceiving and Understanding the Surrounding Environment for an ASV

Assuring safety is arguably the most crucial factor when developing an autonomous vehicle [War06] not only for the vehicle itself but for the people and vehicles close to it. To safely act in an environment, it is essential to perceive and understand the surrounding environment as well as possible. This understanding of the environment and the safety assessment of a situation is defined as situation awareness (SA) and is something humans inherently possess. However, SA needs to be implemented in an autonomous vehicle to assess the risks in an environment before acting. Further stated by the main conclusion of the paper "*The Role of Situation Awareness in Assuring Safety of Autonomous Vehicles*" SA is the key factor in assuring safety in autonomous vehicles [War06]. Therefore, implementing a good SA system is essential for a successful and safe ASV, which can further reduce costs and safety risks within the maritime domain.

1.2 Project Description

Our project aims to develop a novel method that utilizes visual sensors to implement SA as a dynamic scene representation in an ASV. A dynamic scene representation contains vital information about the surrounding environment. Depending on the domain or situation, the importance of information types varies. For an ASV, information about reefs, nearby boats, weather updates and maritime regulations are deemed important information. Other important information is the attributes of vessels, such as their position, identification and path. Additionally, supporting the ability to track the boats' identities across various frames gives valuable information about their path.

1.3 Scope of the Master's Thesis

In order to implement SA in an ASV, simulated data was utilized to represent the environment of the ASV. However, it is important to state that creating a realistic and comprehensive simulator is outside the scope of this thesis. Creating a simulator able to generate video sequences of boats with a simplified motion model will be sufficient for the scope of the thesis. As defined in the project description, the ASV should utilize visual sensors. Therefore data commonly used in computer vision tasks such as object detection and tracking data must be generated. However, generating high-quality object detection and tracking data is outside the scope of the thesis. This is because much work has been published on improving object detection and tracking algorithms. Today there are publicly available state-of-the-art detection and tracking algorithms that can detect [He+22] and track a high percentage of objects in images. However, there is less literature to be found on how to create a dynamic scene representation for an ASV. Therefore, the scope of the thesis is about generating sufficient object detection and tracking data from a simple simulator and then implementing and testing methods to convert the data into a dynamic scene representation.

1.4 Summary of Main Contributions

Throughout the development of this thesis, various types of contributions have been made to the SA and ASV domains. An overview of the contributions is listed below. Each contribution will be described in further detail in chapter 3.

- A Blender script that generates stereo- or mono-video sequences of boats in a maritime environment.
- A synthetic dataset containing the maritime images with ground truth bounding boxes.
- Trained a You Only Look Once (YOLO)v8 model to detect boats from the synthetic dataset.
- Generated tracked stereo sequences with ByteTrack utilizing the trained YOLOv8 model for object detection.
- A novel modularized SA architecture for a stereo camera system.
- An implementation of the SA architecture that represents an ASV's surroundings from tracked stereo video sequences.
- A script that generates positional, temporal, dimensional and identifiable ground truth from the synthetic data.
- A script that visualizes the dynamic environment representation from a bird's-eye view (BEV) perspective.

Chapter 2

Relevant Background Topics

This chapter will present relevant background theory for developing the SA system. It will explain more deeply what terms like SA, object detection and tracking means and compare synthetic data with data from real-world experiments.

2.1 A Broader Definition of SA

SA is a cognitive process that plays a vital role in a wide range of domains, from aviation and military operations to healthcare, emergency management, and even everyday decision-making. SA refers to understanding one's environment, including its dynamic elements, and the ability to anticipate future events or changes. It involves actively gathering information, processing it effectively, and forming a representation of the current situation to make informed decisions and take appropriate actions. To sum up, understanding the environment is necessary to achieve the desired outcome in decision-making.

2.2 What are Endsley's Three Levels of SA

In "*Toward a Theory of Situation Awareness in Dynamic Systems*", Mica Endsley defines a model for SA, which suggests that it can be broken down into three levels. These levels can classify the degree of SA in systems [End95]. Each level in Endsley's model is further described and exemplified within the maritime context below.

2.2.1 First Level of SA

The first level is the perception of elements in the current situation. An element in the maritime domain is defined as everything relevant to the ASV like other boats, humans, and weather information. Furthermore, elements can also be static such as coastlines and docks. To achieve the first level of SA, one needs to know the relevant elements' status, attributes, and dynamics. Such as their location, velocity, trajectory, size, and other essential characteristics for the SA. The first level is decisive in reaching the second and third levels [End95].

2.2.2 Second Level of SA

The second level of SA is about comprehending the current situation. To achieve the second level, one must understand the significance of the different elements and their impact on reaching the desired outcome [End95]. Suppose the goal is to park a ferry at a dock, and another boat is approaching the same dock. In this situation, it's crucial to determine how this will affect the ferry's ability to park at the dock. A sound system would comprehend the situation and move towards another available dock.

2.2.3 Third Level of SA

The final level is about projecting future situations, meaning using current and previous situations to predict future situations. To achieve the third level of SA, future actions of the elements in the environment have to be accurately predicted [End95]. Suppose a ferry is heading toward another boat. The ferry should be able to expect that the ship likely will make room at the port side as is regulated by the Convention on the International Regulations for Preventing Collisions at Sea (COLREGs).

2.3 Why SA is Necessary for an Autonomous System

In order to make optimal actions in decision-making problems, a system needs a good perception of the environment. A possible architecture for an autonomous system could be divided into four parts: sense, comprehend, plan, and act. The described architecture can be seen in Figure 2.1. In this architecture, the sense module is responsible for retrieving all the necessary information as accurately as possible. In contrast, the comprehend module aims to get the most out of the sensed data to describe the environment as precisely as possible. The SA is limited to what information the system can retrieve from the surroundings and the quality of the processing and comprehension of the data. Therefore, these modules can be grouped into a SA part of the system [End95]. Based on the SA from the previous modules, the planning module creates possible actions for the system. Then the act module executes the desired action. The planning module is responsible for the decision-making of the system. However, the act module restricts the planning module to physically executable actions.

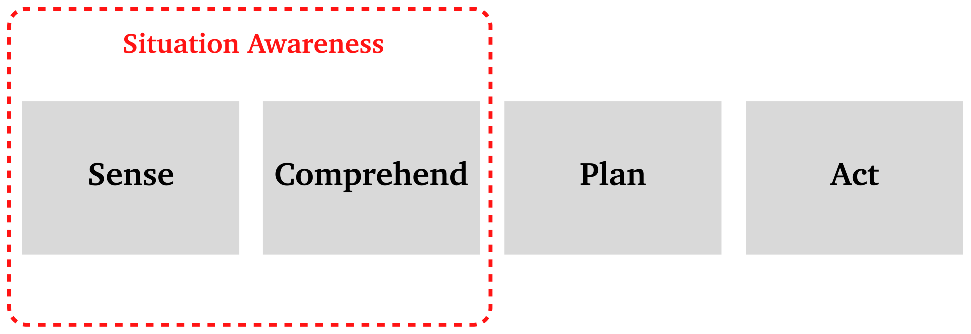


Figure 2.1: A possible architecture for an autonomous system.

2.4 Sensors for SA

In order to achieve SA, it is necessary to sense the environment as stated in section 2.3. An asv can utilize a variety of sensors to perceive its environment. Different sensors vary in functionality and have distinct advantages and disadvantages. The price, range, and size might be important when considering what sensors to use on a ASV. The sensor performances can be measured and are discussed in 2.7. This section presents some of the more common sensors used for SA for an ASV.

2.4.1 Radar and LiDAR

Radar and LiDAR are two sensors that are similar in terms of functionality. Radars utilize radio waves, which result in a sensor capable of measuring at a long range with high reliability. Whilst LiDAR uses light waves to get more accurate results but has a shorter range. Furthermore, LiDAR is more sensitive to environmental influences like fog and rain. Both radar and LiDAR data can be converted to a point cloud. A point cloud is a set of points used to represent the surroundings. The coordinates of the different points can be used to detect potential obstacles. The radar has traditionally been the most common sensor in maritime applications [Hel+22], whilst LiDAR is used for more specific applications such as autonomous vehicles. Both sensors are considered to have a high-cost level [RG05].

2.4.2 Camera

Camera sensors are commonly utilized for autonomous vehicles. In contrast to radar and LiDAR sensors, camera sensors are visual sensors capable of extracting colours and textures. This increases the autonomous system's perception capabilities and subsequently its SA. Cameras can also be used for short, medium, and long-range applications and are often cheaper than radars and LiDARs. Similarly to LiDAR, camera sensors are sensitive to environmental influences. With a camera sensor, an ASV could detect and classify different boats, buoys, or lighthouses. Cameras do not explicitly give any information about the distance to any objects. However, methods exist to extract information about the distances of the different objects. [Cam+18]. Methods for calculating distance using camera sensors are further discussed in section 2.8.

2.4.3 Conclusion

For an ASV, it would be advantageous to utilize all of the aforementioned sensors to create a robust SA system. Combining the sensors lessens their disadvantages and increases their perception capabilities. Even though radars and LiDAR are widely used and have advantages that could make an ASV more robust and reliable, this thesis will, as stated in section 1.2, focus on utilizing visual sensors for an ASV. The following sections will further explain how different methods extract useful information about the surroundings from camera sensors.

2.5 Object Detection with Visual Sensors

The object detection problem can be defined as classifying and finding the positions of objects in an image. In order to classify an object, different classes need to be pre-defined. Classes in the maritime environment could be ships, kayaks, or sailboats. A limitation of pre-defined classes is that when encountering objects that aren't represented in these classes, the object is either classified as an unknown object or not classified at all. In SA for an ASV, object detection provides crucial information about visible objects and a better understanding of the environment.

2.5.1 How to Represent Objects

There are different methods to represent a detected object. The most common approach is to use Axis-Aligned Bounding Box (AABB) or instance segmentation. Examples of AABB and instance segmentation are shown in Figure 2.2. Both methods have their pros and cons. AABB is a simpler representation where detections are represented as boxes surrounding the detected objects. In instance segmentation, a more exact representation is created with an overlay of the object. When choosing between AABB and instance segmentation, there is a trade-off between speed and how detailed the representation is [Ahr22]. In real-time applications, AABB is often considered sufficient and preferred because of its speed and simplicity.

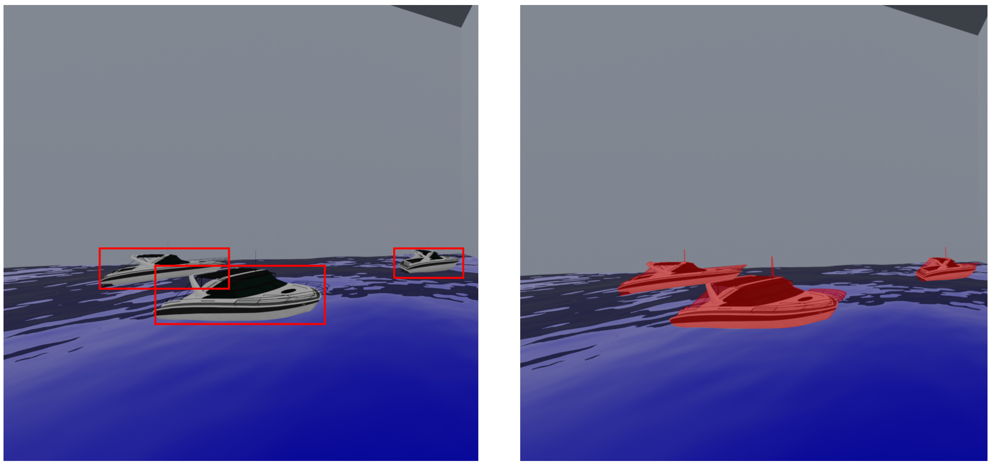


Figure 2.2: Examples of AABBs (left image) and instance segmentation (right image).

2.5.2 Convolutional Neural Network (CNN)

CNN is one of the most representative neural networks within deep learning and has been making brilliant achievements in multiple machine learning problems. CNNs can be used to solve object detection problems efficiently, and many state-of-the-art object detection models are based on CNNs [Ahr22]. Important components of a CNN are presented in the text below.

Image Features

A CNN is able to extract features in an image through convolutional layers. Features can be described as the characteristics of an object. Furthermore, there are different levels of features reaching from low-level features, including edges, corners, and simple shapes, to more complex shapes, like the shape of a person or a boat.

Convolutional Layers

In the convolutional layers, a kernel extracts a specific feature. A common technique is to add padding to the image to prevent loss of information on the image borders. Stride is the number of places the kernel is moved while iterating through the input and is used to control the density of the convolution. The input size for each iteration equals the size of the kernel. The output is the sum of the scalar product of the fixed-size input and the kernel. Then the process is repeated with the entire input shifting the number of places equal to the size of the stride until all the input has been processed. The output of a convolutional layer is a feature map for each kernel in the layer. [Li+21]. A convolution process is illustrated in Figure 2.3.

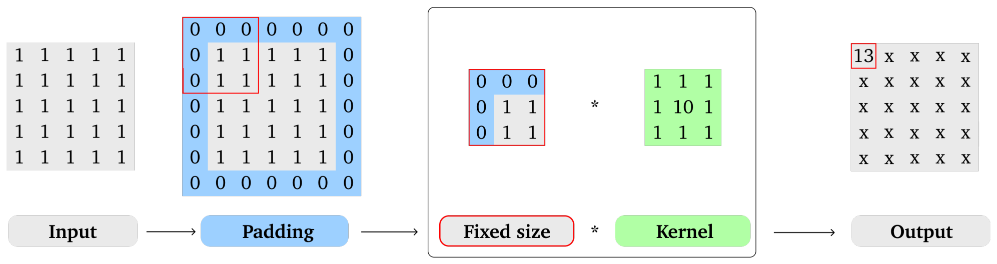


Figure 2.3: An example of how the output of a convolutional layer is calculated.

Pooling Layers

To avoid overfitting from a large number of features, pooling is used. Pooling is a way to generalize the convolutional layer output and is also called downsampling [Li+21]. In the pooling layer, the pooling operation is performed on the feature map. Then the feature map is shifted by the length of a given stride and repeated for the rest of the spatial dimensions. A standard pooling operation is max pooling. In max pooling, each channel's maximum element of the fixed-size areas is returned. Average pooling is another operation where the average of each fixed-size area is returned. The illustration in Figure 2.4 is an example of a max pooling operation, where each colour represents a fixed-size area.

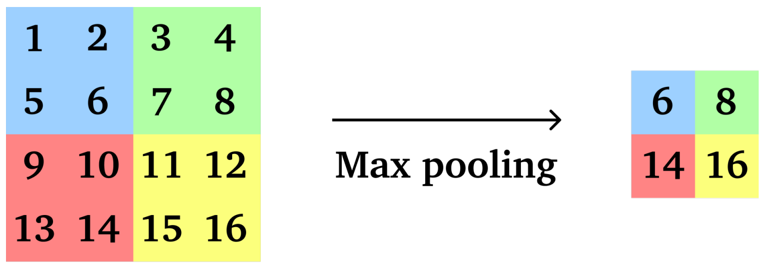


Figure 2.4: An example of a max pooling operation.

2.5.3 You Only Look Once (YOLO)

YOLO is a state-of-the-art, real-time object detection system built on the CNN architecture. The framework is called "You Only Look Once" because it can complete the detection task with a single pass through the network. YOLO is used to generate bounding boxes in many real-time applications today and is often chosen because of its combination of high speed and great accuracy. The framework has been through several iterations from the initial version to the newest version YOLOv8 with improvements in both speed and accuracy. YOLOv8 is publicly available and can be run from the command-line interface (CLI) or installed as a PIP package [TC23].

How a Detection in YOLO is Represented

The detections made by YOLO are represented as AABBs with additional information about the class the detections belong to and a confidence score. The confidence score is a number between 0 and 1 and is made up of the multiplication of two factors. Firstly, how likely it is that the classified object is in the AABB. Secondly, how accurately do the dimensions of the AABBs cover the actual dimensions of the object in the image [Tri+22]. How to calculate the confidence score for a given bounding box can be seen in the formula below:

$$\text{Confidence Score} = \Pr(\text{Object in AABB}) \cdot \text{Accuracy of AABB}$$

YOLO has a threshold for generating detections based on the confidence score. The default value in YOLOv8 is a threshold of 0.25, which entails that any detection made with a confidence score under 0.25 is discarded. When considering different thresholds, there is a trade-off between having fewer detections with high confidence or several with lower confidence. This trade-off will be further discussed in 2.7.3.

2.5.4 The Difference between 2D and 3D Object Detection

Two-dimensional (2D) object detection can detect an object in an image and represent it as AABB or instance segmentation. However, these representations lack information about the object's dimensions and rotations. three-dimensional (3D) object detection is introduced to include valuable information about objects in a 3D context. The difference between AABBs and 3D can be seen in Figure 2.5. However, 3D object detection methods are still in early development and require refinement to improve their detection accuracy, lower computational complexity, and reduce costs. In contrast, 2D methods are relatively more established [GK22].

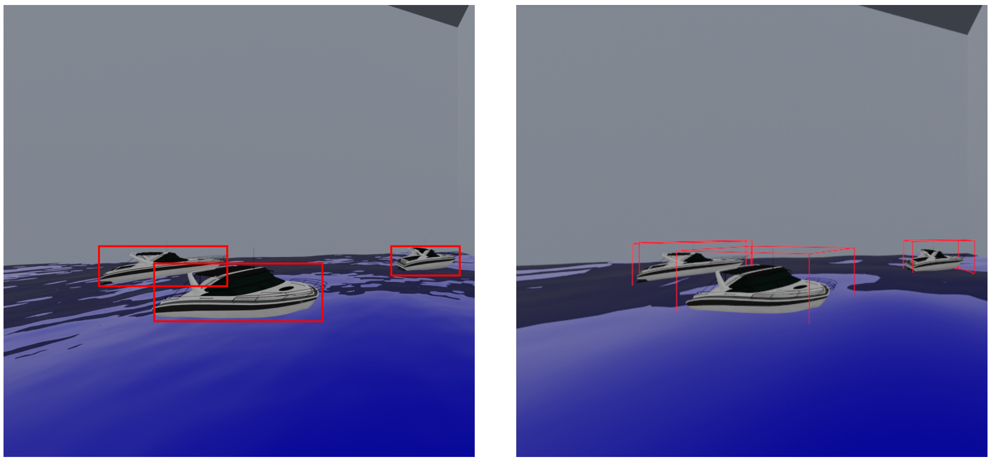


Figure 2.5: Examples of AABBs and 3D bounding boxes.

2.5.5 How an Object can be Represented in a 3D Context

Six degrees of freedom (6-DOF) refers to the six mechanisms of movement in a 3d space. An object could move and rotate across the x-, y- and z-axis. For 3D bounding boxes to represent an object with 6-DOF, the representations must include height, width, length, yaw, pitch and roll. A way to represent the bounding box is to represent it as four points. From the front of the object, the bottom left corner, bottom right corner, and top right corner are used. From the back of the object, the bottom right corner is used. All the information about rotation and dimension from the 4 points can be subtracted. Height, width and length are calculated as the distance between the bottom right and top right corner, bottom right and bottom left in the front, and the bottom right corners in the front and the back.

2.5.6 Conclusion

Object detection is crucial for an ASV that utilizes visual sensors. The balance between speed and accuracy is vital for real-time applications, influencing how to represent objects and what object detection models to choose. Object detection models from the YOLO family are based on the CNN architecture and produce impressive results in terms of speed and accuracy.

2.6 Target Tracking on Image Sequences

Detecting and classifying objects is crucial for SA, but it does not indicate the paths of the detected objects. However, tracking algorithms are used to associate objects through multiple frames and retain their identities and can be used to find the paths of objects. When multiple detections at different points in time have been assigned an identification, it is defined as a track. Tracking increases the SA of the ASV as ships can be distinguished from each other and identified over time.

2.6.1 Single- and Multi-Target Tracking

Single-target tracking is defined as identifying an object and keeping track of the object's identification over a series of image frames, assuming that only one object exists in the frame. Multi-target tracking is a variation of target tracking. The tracking implementation has to detect multiple objects and then create and remove tracks as the objects enter and exit the frame. In the context of an ASV, there can be multiple actors; therefore, multi-target tracking is preferable.

2.6.2 How to Associate Objects with Tracks

In order to know what track a detection should be associated with, there has to be a method to evaluate how suitable a track is for a given detection. Association costs are predictions of how likely a detection is an already existing track detected in a previous frame. Choosing how to create the association costs is a crucial part of multi-target-tracking and can range from simple to complex implementations. The costs are dependent on a set of variables that describe the object's appearance and characteristics. These variables are then compared to the existing track and usually calculated with a weighted average to produce the association cost between the target and the preexisting track. There exist many implementations of association costs, and the context of the task should be considered before choosing since this could be a factor in the accuracy of the costs [Ahr22].

2.6.3 Keeping Track of Reemerging Objects

Re-Identification (ReID) aims to keep track of objects across non-overlapping cameras or when an object disappears from the camera view and reappears later. Tracking algorithms often utilize the fact that an object will have a position close to its position at previous timestamps. But when an object disappears and reappears at a later timestamp, it is difficult to associate it with the position of previous detections alone. Instead of only looking at the similarity in position, a common approach is to analyze feature similarities, which are often a key factor in high ReID performance. The importance of ReID varies depending on the application of the multi-target tracking algorithm, and in the case of an ASV, one could argue that it is acceptable if an object is assigned a new id when reentering the frame. [Ahr22].

2.6.4 Improving Track Predictions by Using a Kalman Filter

When tracking dynamic objects in an image frame, an essential task is to accurately predict the current state of each previously existing track [Ahr22]. However, when estimating the state of a track, there would always be some noise present in the estimation. In addition to noisy estimations, there would also be measurement noise when working with sensors in real-world applications. The Kalman filter is an algorithm that combines estimations with measurements to improve the state estimations continuously. It consists of two sets of equations called prediction equations and correction equations. The prediction equations are used to estimate the state for the next timeframe, while the correction equations combine the predicted estimations with the current measurements to estimate the state of the current timeframe [WB+95]. The Kalman gain (K) is a matrix calculated to improve how predictions and measurements are weighted continuously. The matrices and vectors used in the equations in the Kalman filter are listed in Table 2.6.4.

Symbol	**What it represents**
$\hat{x}^-$	A prediction of the state
$\hat{x}$	A correction of the state
A	Matrix A relates the state at timestamp k to the state at timestamp k+1 if noise is ignored and no driving function is used.
u	The control input
B	Matrix B relates the control input to a state
P^-	A prediction of the error covariance
P	A correction of the error covariance
Q	The variance in the prediction noise
K	The Kalman gain
z	The noisy measurement
H	Relates the state to a measurement
R	The variance in the measurement noise
I	Identity matrix

The prediction equations used in a discrete Kalman filter are given below:

$$\hat{x}^-_{k+1} = A_k \hat{x}_k + B u_k$$

$$P^-_{k+1} = A_k P_k A_k^T + Q_k$$

The correction equations used in a discrete Kalman filter are given below:

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1}$$

$$\hat{x}_k = \hat{x}_k^- + K_k (z_k - H_k \hat{x}_k^-)$$

$$P_k = (I - K_k H_k) P_k^-$$

The results from the correction equations are used in the prediction equations, and the results from the prediction equations are used in the correction equations and denoted as the Kalman Filter cycle. An illustration of the Kamlan Filter cycle can be seen in Figure 2.6.

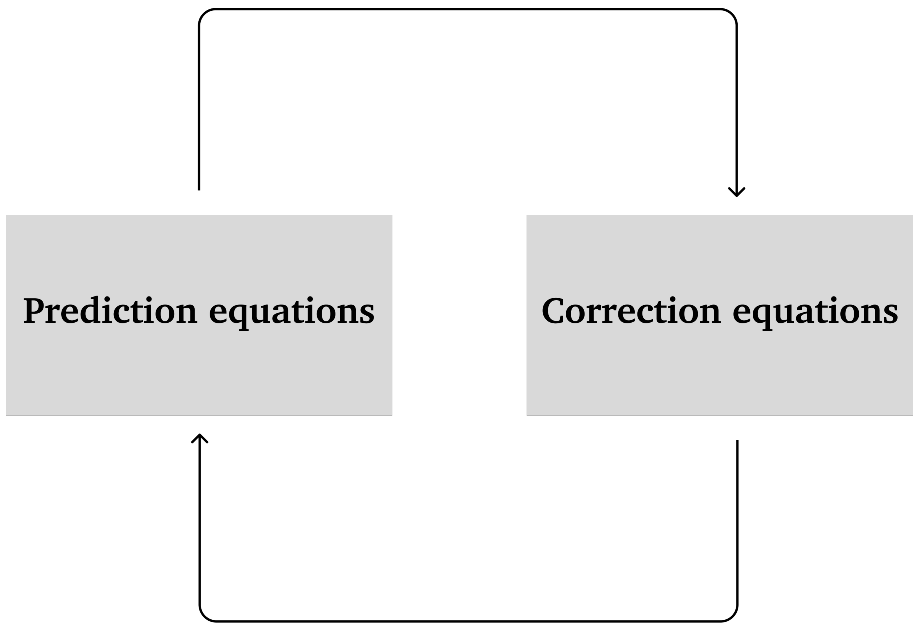


Figure 2.6: The continuous Kalman Filter cycle.

One thing to notice is that the final estimate of the state is somewhere between the predicted state and the measurement. If K equals 0, the state estimate is equal to the prediction. And if K is equal to I, the estimate is equal to the measurement [WB+95].

2.6.5 How to Associate Tracks with Detections

The problem of associating tracks with detections is a variant of the assignment problem. The assignment problem is represented as a nxn matrix and is zero-padded if the input matrix is not squared [Ahr22]. In the matrix, the rows represent workers, and the columns represent tasks. The cell values are defined as the cost of the task for the worker. The assignment problem is to select a task for each worker with the lowest cost possible. However, a worker can only be assigned one task, and a task can only be assigned to one worker. The Hungarian algorithm can solve the assignment problem and, thus, associate tracks with detections. Below is an example of how the Hungarian algorithm can solve an assignment problem for a $3x3$ matrix.

A $3x3$ input matrix representing costs for the assignment problem

$$\begin{bmatrix} 8 & 4 & 7 \\ 5 & 2 & 3 \\ 9 & 4 & 8 \end{bmatrix}$$

1. Find the minimal element in each row and subtract it from each element in that row.

$$\begin{bmatrix} 8 & 4 & 7 \\ 5 & 2 & 3 \\ 9 & 4 & 8 \end{bmatrix} \rightarrow \begin{bmatrix} 4 & 0 & 3 \\ 3 & 0 & 1 \\ 5 & 0 & 4 \end{bmatrix}$$

2. Find the minimal element in each column and subtract it from each element in that column.

$$\begin{bmatrix} 4 & 0 & 3 \\ 3 & 0 & 1 \\ 5 & 0 & 4 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 2 \\ 0 & 0 & 0 \\ 2 & 0 & 3 \end{bmatrix}$$

3. Cover all zeros by marking as few columns and rows as possible. A solution can be found if the number of rows and columns equals n. However, if the number of rows and columns is less than n, continue to the next step.

$$\begin{bmatrix} 1 & 0 & 2 \\ 0 & 0 & 0 \\ 2 & 0 & 3 \end{bmatrix}$$

In the example, all zeros can be covered by marking one column and row. Since the number of rows and columns selected does not equal the dimensions, $2 \neq 3$, the algorithm continues with step 4.

4. Find the lowest unmarked value, subtract it from all unmarked cells, and add it to all cells marked by a row and column. Then repeat step 3.

$$\begin{bmatrix} 1 & 0 & 2 \\ 0 & 0 & 0 \\ 2 & 0 & 3 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 2 \end{bmatrix}$$

Repeat step 3 and mark all the zeros by choosing as few rows and columns as possible.

$$\begin{bmatrix} 0 & 0 & 2 \\ 0 & 1 & 0 \\ 2 & 0 & 3 \end{bmatrix}$$

Now the number of rows and columns equal the dimensions, $n = 3$, and a solution can be found.

To find a possible solution select n zeroes from the cells in the marked rows and columns without selecting a zero from a previously selected row and column.

$$\begin{bmatrix} 0 & 0 & 2 \\ 0 & 1 & 0 \\ 2 & 0 & 3 \end{bmatrix} \rightarrow \begin{bmatrix} 8 & 4 & 7 \\ 5 & 2 & 3 \\ 9 & 4 & 8 \end{bmatrix}$$

2.6.6 ByteTrack

ByteTrack is a state-of-the-art tracking algorithm for objects in videos. While many tracking algorithms filter out detections with low confidence scores, ByteTrack is designed with a simplistic association method which keeps almost all detections. Then ByteTrack separates between bounding boxes with a high and low confidence score for a given threshold. After the separation, the Kalman filter is applied to predict the new locations of the tracks in the current frame. Detections with a high confidence score and the predicted tracks for those detections are associated. The similarity is measured by Intersection over Union (IoU) or ReID feature distance. IoU will be described in subsection 2.7. The Hungarian algorithm is used to associate the detections with the tracks. The detections with a low confidence score are then associated with the remaining tracks by computing their IoU [Zha+22].

2.7 Common Evaluation Metrics

This section will cover common evaluation metrics utilized in object detection, target tracking and other domains. Different evaluation metrics evaluate different system characteristics, and for an ASV, there are strict requirements for safety and functionality. Therefore, it is essential to choose the correct evaluation metric for a given task and understand what the metric measures.

2.7.1 Mean squared error (MSE)

A simple and common way to measure the performance of a numeric prediction is to look at the MSE. The formula to calculate MSE is show below:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

The square root of the MSE gives the mean squared error (MSE). Both MSE and RMSE generate large values for significant errors and are thus prone to outliers.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

2.7.2 IoU

In general, IoU measures the overlap of two areas. The result is a number from zero to one, where one indicates a perfect overlap, while zero indicates no overlap at all. An example of different IoU results are illustrated in Figure 2.7.

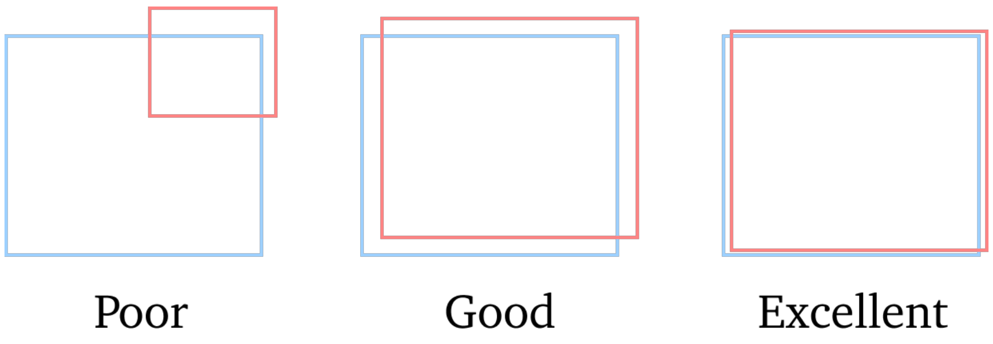


Figure 2.7: A qualitative example of IoU evaluations.

Furthermore, IoU can be used to evaluate the performance of an object detection method by comparing the generated AABB or instance segmentation with the ground truth. The formula for calculating IoU is given below and illustrated in Figure 2.8:

$$\text{IoU} = \frac{\text{Area of overlap}}{\text{Area of union}}$$

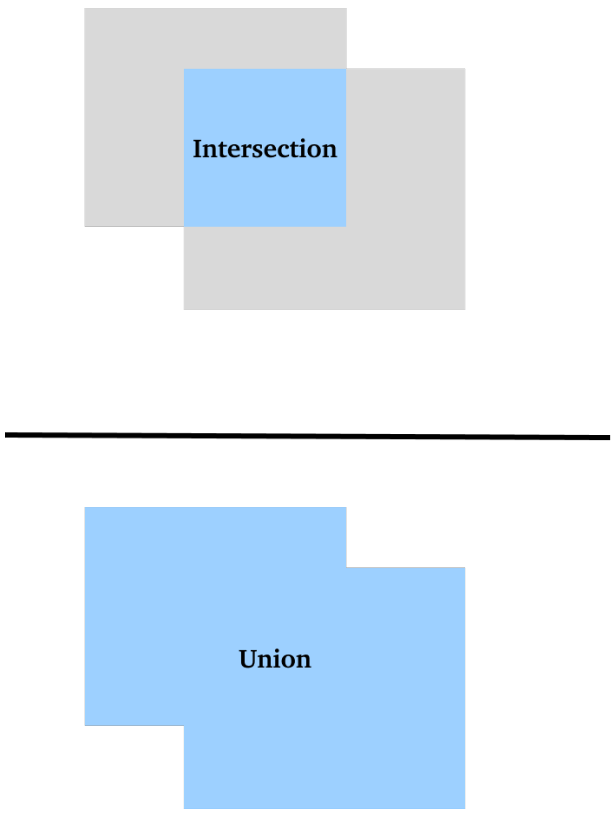


Figure 2.8: A figure illustrating how IoU is calculated.

2.7.3 Precision and Recall

Precision indicates how many of the detections are correctly associated with a target. Recall indicates how many of the targets are detected. There is a trade-off between precision and recall when deciding on a confidence threshold in an object detection method. The precision-recall curve indicates the trade-off between precision and recall at different thresholds. To calculate precision and recall, the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) are considered. The formulas are given below.

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + TN}$$

Clutter is defined as a detection that is not associated with a target. Additionally, a miss-detection is defined as a target not associated with any detection. High recall can result in a high clutter rate, while high precision can lead to a high miss-detection rate in object detection.

2.7.4 Mean Average Precision (mAP)

Mean Average Precision (mAP) is a common evaluation metric for evaluating different object detection methods. For instance, when YOLO models are compared, mAP is an important factor when evaluating the models. MAP uses the precision-recall curve to balance the clutter and miss detection rate. To determine how well a class is predicted, the Average Precision (AP) is calculated. To calculate the AP, the average precision of a class for a given IoU threshold is calculated. A detailed explanation of how mAP is calculated can be seen below [TC23]:

1. The precision-recall curve is calculated.
2. Compute each category's AP using a given number of interpolations.
3. The mAP metrics are often calculated using different IoU thresholds.
4. For each of the IoU thresholds, take the mean of the APs for all the classes.
5. The mAP is given as the average across all IoU thresholds.

2.8 Camera Depth Estimation

For an ASV, the relative distance to other objects is vital information for several aspects of the SA. For example, to ensure safe navigation and planning, detecting objects and estimating their location is essential to avoid collisions. When utilizing radar and LiDAR sensors, calculating the distance to detected objects is straightforward. However, calculating the distance to objects can be a complex task when using camera sensors. Camera sensors can be set up in different ways, and two common approaches are monocular- and stereo-camera setups. When using camera sensors, depth is often calculated to gain information about a pixel in an image. Depth is not the exact distance to the pixel but the distance projected by the z-axis of the camera. Figure 2.9 illustrates the difference between depth and distance. This section will present how to calculate the depth of objects in images for these camera setups and how to calculate distance from depth estimations.

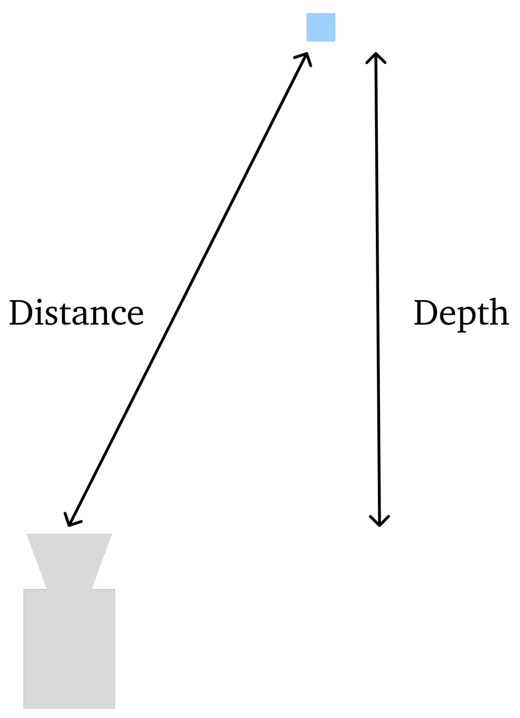


Figure 2.9: An illustration of the difference between distance and depth.

2.8.1 Monocular Camera Depth Estimation

Monocular camera depth estimation calculates the depth of an object in an image from a single camera. Early research utilized depth cues to calculate depth maps from monocular images. Using cues is similar to how humans judge the depth of objects in images, and could, for example, be vanishing points, occlusion, shadows etc. However, in recent years many deep-learning methods have been created to generate better depth estimations from monocular images [Min+21]. Some advantages of monocular depth estimation instead of stereo depth estimation are that it is more robust to sensor failures and that objects only need to be present in a single camera.

2.8.2 What is Stereo Camera Depth Estimation?

A stereo camera setup can be defined by displacing two cameras by a given distance defined as the baseline. When the two cameras observe an object in the image plane, the distance between the location of the pixels in the two images is known as the disparities. By using the disparities of an object, several methods can be used to calculate its depth [JKS95]. Despite significant progress in monocular depth estimation models, there is a substantial gap between stereo- and monocular-depth estimation accuracy. Stereo depth estimation models perform better than monocular depth estimation due to the fundamental limitations of monocular depth estimation [SKB18].

2.8.3 How to Convert Depth Estimations to Distance Predictions?

The following formula can be used to calculate the distance of objects in images from monocular or camera setups. For a monocular setup, the distance is computed by dividing the depth by the cosine of the angle (θ) between the object in the image and the camera. However, the cosine of the mean angle (θ) for each camera is used for the stereo camera setup. The formula is given below.

$$\text{Distance} = \frac{\text{Depth}}{\cos\theta}$$

2.9 Data Used in an ASV

2.9.1 Why Data is Crucial for an ASV

Most ASVs employ a variety of machine learning (ML) methods for the decision-making and perception of the system. These ML methods could require substantial amounts of training data to achieve adequate results. Furthermore, the data needs to be compatible with the task and domain of the ML model with a high degree of variation within the data. Additionally, for supervised ML methods, the methods necessitate accurate ground truth values for the dataset. Thus, acquiring high-quality data is a fundamental requirement for developing an ASV system that utilizes ML methods. Data can be divided into two categories: captured from the real world and synthetically generated. The following subsection presents how an ASV can utilize synthetic data. In contrast, the use of synthetic data for an ASV is discussed in subsection 2.9.3.

2.9.2 Real Data for Training Autonomous Vehicle Systems

Data Captured from Experiments in the Real World

From this point on, we will refer to data collected from real-world experiments as "real data". Real data in the context of autonomous vehicles could consist of information from the surrounding environment captured by sensors on the vehicle. In addition, the sensor data can vary from radar or LiDAR point clouds, videos, and positional data that cover various time intervals and situations.

What are the Advantages of Using Real Data

Utilizing real data for training an autonomous vehicle increases the authenticity and reliability of the system. Real data could reflect reality better than simulated data, as there could be biases in simulated data that do not exist in the real world. However, real data is also limited by how it is collected and the limitations of the sensors used to gather it. Furthermore, due to the complexity of the real world, a simulator will only replicate it somewhat. Therefore real data inherently has better authenticity and is more similar to the input data of an ASV operating in the real world.

What are the Challenges of Collecting Real Data

There are quite a few challenges in collecting data from the real world. Accurately collecting data depends on the accuracy of the acquisition methods and tools. It is well known that there is a lot of noise in gathering real data due to sensor malfunctions or human error. Furthermore, conducting experiments for generating data could be limited by money and risk to human and object safety. Another challenge with real data that include information about humans is to avoid breaching data protection laws such as GDPR. Lastly, after creating a dataset, it is, as mentioned earlier, sometimes necessary to generate accurate ground truths for the input data, which again is a time-consuming and complex task.

Datasets for Autonomous Vehicles Containing Real Data

A variety of excellent open-source datasets contain real data for autonomous driving. An example of this is the KITTI dataset which is one of the benchmarks for testing a variety of autonomous vehicle tasks. The KITTI dataset comprises camera images, GPS/Inertial Measurement Unit (IMU) data and laser scans collected in Karlsruhe, Germany [Gei+13]. It is widely used to test many machine-learning methods and autonomous driving tasks, including 2D object detection, tracking and 3D object detection.

Datasets for ASVs Containing Real Data

In the maritime context, there are not currently any datasets used as a benchmark to evaluate the performance of ASV. There are, however, publicly available datasets which exist that can be used similarly to the KITTI dataset, such as the *the Maritime Sensor Fusion Benchmark Dataset (MSFBD)* by Zeabuz [Hel+22]. The MSFBD dataset contains data from multiple sensors such as radar, LiDAR, IR and RGB cameras. The data in the MSFBD dataset comes from the sensors on MilliAmpere. MilliAmpere is an autonomous ferry developed by Norwegian University of Science and Technology (NTNU). In addition to the sensing system, a navigation system provides position and heading information. The testing was conducted on two distinct testing environments in Trondheim, Norway. One of the environments was a big open area, while the other was in a narrow channel. The MSFBD dataset can be used to train supervised machine-learning models for several ASV tasks. However, the dataset has some limitations, for instance, the limited size of the dataset. Furthermore, a lack of variation in weather, the surrounding environment, and scenarios can also be considered a limitation of the dataset [Hel+22].

2.9.3 Synthetic Data for Training Autonomous Vehicle Systems

What are the Advantages of Using Synthetic Data?

Due to the challenges of collecting real data, attention has been turned to using synthetic data to support or even replace datasets with real data. Synthetic data generation can range from imitating real datasets to using sophisticated simulators that replicate the real world as accurately as possible. Utilizing software to generate data makes it possible to create a mass amount quickly. Furthermore, the software can generate photo-realistic faces that do not violate any privacy laws. In general, synthetic data removes many ethical and safety issues that arise during real data collection. In contrast to real-world driving experiments, when running a car simulator, it is possible to generate many more scenarios, including those that could pose a high risk to the safety of human beings and vehicles. Furthermore, crashes and near misses are vital for training a model to increase the robustness and safety of an ASV. Perhaps the most significant advantage of synthetic data generation is the mass generation of precisely annotated labelled data. Not only is manually annotating time-consuming and costly, but it also usually lacks a high degree of accuracy. Even in the industry standard dataset of Microsoft Common Objects in Context (MS COCO) dataset, the instance segmentation mask of several objects are crude outlines and misses significant features [Nik21]. When generating ground truths, for instance, segmentation or positional data of objects, several factors can reduce the quality of the truth values, such as noise in sensors or tedious manual labour, such as creating instance segmentation masks for thousands of images. When using, for example, a car simulator, it is possible to access the exact location of all the agents in the environment, including information about their relative angle, type of vehicle, velocity and much more information. Another example is generating datasets for image classification where the segmentation masks will be pixel-perfect to the rendered synthetic image. By generating synthetic data, the dataset will be practically perfectly annotated and can be easily mass-produced.

The Challenges of Using Synthetic Data

One key limiting factor when generating synthetic data is the simulator's realism. Creating a simulator capable of real-life physics and photo-realistic graphics is complex and costly regarding time and money. As a result, simulators will depict an abstract imitation of the real world. Thus, when training a model solely on synthetic data, the model will create biased generalizations from the data due to a possible lack of variability and correlation in the data [Dil22]. Even though the ground truths can be calculated precisely, it is still prone to human error. Furthermore, human verification is still necessary to evaluate the quality of the ground truth values.

Simulators for Autonomous Vehicles

There exist some high-end simulators for autonomous cars with ground truth bounding boxes for vehicles, advanced physics and the possibility of generating custom maps. The cars can be equipped with various sensors such as LiDARs, multiple cameras, depth sensors, and GPS [Dos+17]. These simulators have the necessary variety of sensors and ground truth data needed to train a ML method for tasks within autonomous driving. However, they do not support boats or other maritime vehicles. There is a minimal selection of publicly available ASV simulators with a much lower degree of sophistication in comparison to the car simulators.

2.9.4 Comparing Real and Synthetic Data for Training Autonomous Vehicle Systems

ML methods trained on real data have achieved good results in many autonomous vehicle tasks. However, a study analysed the effect of synthetic data for training an object detection algorithm and testing the performance on the KITTI Benchmark dataset. The study concluded that "training on a simulation does not translate to generalizability on real-world data and performance depends on a number of factors" [Tal+20]. The study compared the results of a model trained on synthetic data from two distinct simulators with a model trained on real data. The model trained on real data did significantly better on the KITTI dataset. The study findings surmise that a lack of diversity in simulated landscapes and insufficient graphical quality to imitate realism are responsible for poor object detection performance. Thus, simulators must be capable of generating a broad range of scenarios representative of those encountered in the real world with an adequate degree of realism. Furthermore, the study discussed that combining the synthetic data with real data could improve generalisation and, in turn, the performance of the 2D object detection model.

2.9.5 How Synthetic Data can Complement Real Data

Research has been conducted on combining real and synthetic data to get the advantages of both. In a study by the authors of SYNTHetic collection of Imagery and Annotations of urban scenarios (SYNTHIA), the performance of the models increased in almost all classes when combining real and simulated data compared to only using real data. When combining real and synthetic data, it could either be mixed in the training set, or one could conduct transfer learning. Transfer learning is when a model is pre-trained and then fine-tuned. In this case, the model would be pre-trained on the synthetic data and then fine-tuned on the real data. Lastly, there are drawbacks to combining real and synthetic data as well. One major concern is that using synthetic images in training may lead to a decrease in performance on real images, referred to as domain shift. To limit the effects of domain shift, the synthetic data needs to possess a high degree of variation, for example, different types of lighting, texture, weather and object types in simulators. By generating a more varied and subsequently difficult synthetic dataset, the model performs better when shown real images, as these also have a great variation in similar parameters. [SLW20].

2.9.6 Conclusion

In domains lacking high-quality datasets, such as the maritime domain, it can be necessary to generate synthetic data. However, models only trained on synthetic data perform poorly when tested on real data. This can be solved by training on synthetic data and then fine-tuning on real data to produce great results. Furthermore, an essential factor for the quality of an ASV is the safety of the vehicle and its surroundings. Therefore, generating data which contains dangerous and uncertain situations is also necessary to create a robust model. Such data is difficult, costly and dangerous to collect in the real world, re-identifying the need for quality synthetic data in the maritime domain.

Chapter 3

Situation Awareness Architecture: Implementation and Experimentation

3.1 Introduction

This chapter will discuss the steps and methods for developing a ASV SA system. Initially, the chapter describes the creation of the synthetic video generator that will be used to create synthetic datasets for ASV SA tasks. Next, the chapter takes on the architecture of the final ASV SA system. Then the implementation of the different parts of the architecture will be presented, such as how we trained a model with the YOLOv8 algorithm described in subsection 2.5.3. And how the ByteTrack algorithm was used on the generated AABB for tracking. Then, the use of stereo depth estimation and bearing estimation to calculate the depth and bearing of the tracked objects will be described. And lastly, how a bird's-eye view (BEV) was generated to visualise the tracked objects with their corresponding distance.

3.2 How we Generated Synthetic Data for ASVs in Blender

In order to implement a ASV SA system utilising visual sensors, there is a need for maritime video sequences to train and test on. We decided to generate our own maritime synthetic dataset due to a lack of available large labelled maritime datasets containing video sequences. Furthermore, generating realistic synthetic data adds value to the thesis and further development of SA in ASVs. Video sequences were generated with Blender to create object detection and tracking data to reach the final goal of creating a dynamic scene representation.

3.2.1 What Degree of Realism Should be Replicated?

When generating synthetic data, an important aspect to consider is to what degree the synthetic data should replicate the real data. During the prestudy, several synthetic datasets were investigated across domains, and we wanted to emulate the success of the state-of-the-art simulators in the automotive industry. These simulators are three-dimensional (3D), and the vehicles drive in realistic patterns. Furthermore, the simulators have a semi-realistic visual appearance, generating more convincing visual data. Photo-realistic video sequences with a high degree of variation could be extracted from these simulators. Therefore, our synthetic data should strive to meet these standards.

Limiting Vehicle Dynamics to Three Degrees of Freedom

In the real world, an object such as a boat can move in 6-DOF as it can move forwards, backwards, from side to side and up and down in the ocean. Due to waves, the boat also rocks sideways and back and forth. Additionally, it can be steered in any direction. Therefore, by replication the 6-DOF, it is possible to simulate the movements of a real ship accurately. However, due to the time constraints of the thesis, we decided to simplify certain aspects of the synthetic data. Since the movements of a boat are complex, we decided to limit the boat's possible movements to three degrees of freedom (3-DOF), which means that the boat can move forward, backwards, and steer in any direction. Furthermore, this limitation removes the possibility of the boat moving up and down in the water, and it is not subject to rotations due to waves and other forces. This, in turn, abstracts the synthetic data from real boat movements, and a model trained on this data will, in turn, have trouble generalising if tested on real data. However, by simplifying the boat dynamics, several ASV SA tasks such as object detection, tracking and depth estimation become easier. Whilst limiting the dynamics to 3-DOF is an abstraction, the synthetic data generated is viable to utilise for the implementation of a novel ASV SA system.

How Photo-Realistic Should the Synthetic Data be?

If the synthetic data is to be used on computer vision tasks such as object detection and tracking, it could require some degree of photo-realism to achieve sufficient results. Therefore, the video sequences should contain maritime imagery, for example, boats and water. The objects should have textures or images that appear as the objects they imitate. However, the quality does not have to be completely photo-realistic but should be generalisable with real imagery.

How Dynamic Should the Environment be?

State-of-the-art simulators in the automotive industry have a high degree of variations in object classes, weather, scenery, environment and other environmental factors. Having a high degree of variation increases the realism of the simulator and, subsequently, the data generated from it. However, due to the scope and limited time of the thesis, we decided to limit the degree of variety in the synthetic data. As generating high-quality synthetic data with a high degree of variation is a complex and time-consuming task. However, adding the variation in the aforementioned environmental factors is an important task if the synthetic data is to be used for further research.

3.2.2 How to Simulate the Movement of Vessels

For our synthetic data, the boats should behave realistically and not just move in the same directions or have completely random movements. Generating realistic movements of objects that also consider the movements of other objects in a scene is a complex and time-consuming task. Therefore, using an already existing method of simulating data was a great alternative considering the limited time of the project. We were provided with pose files generated as part of another NTNU student's master thesis. A pose file includes the bearing of an object in addition to its position. This information can be used to represent a simulated vehicle. This subsection will discuss the advantages and limitations of using the provided pose files.

A Description of the Vehicle Dynamics

The pose sequences are generated using the motion model of a car given the velocity, direction and heading of the vehicle. These pose sequences dictate how the vehicles move, and as a result, the movements are realistic to how a car would drive in real life. The simulated vehicles are reactive agents, meaning they have pre-defined behaviour similar to reflexes. Realistic movement is considered sufficient, and the need for intelligent movement is considered outside the scope as the task is only to gain first-level SA. Additionally, dangerous situations are more likely without intelligent behaviour, which is important data for an ASV to train on to generate robust systems to ensure safety. To better understand the generation of the pose sequences, we recommend reading Henrik Fjellheim's master thesis.

Generating Mass Amounts of Data

An advantage of the pose sequences is that mass amounts of the sequences can be generated quickly. One time step includes all the positional information for each vehicle in the sequence, and it is possible to generate vast amounts of these sequences. Furthermore, it is possible to alter the number of vehicles, the size of the environment and the size of the objects in the environment. In turn, making it possible to generate numerous pose sequences with a high degree of variation in the data.

Perfectly Labelled Ground Truth

One of the major benefits of using the pose sequences is that the sequences can be used for perfect positional ground truth in the synthetic dataset. For each time step in the dataset, each vehicle's x- and y-coordinates in the environment and their heading and velocity are accessible. Furthermore, we have access to both the width and the height of each vehicle. With the agglomeration of this data, it is possible to generate perfect ground truth about the relative angles and distances from an ego-ship from these labelled data.

Limitations of Pose Sequences

There are a few limitations to the provided pose sequences for use in an ASV simulator. Firstly, the pose sequences are generated using the motion model of a car. Generating the pose files with the motion model of a normal leisure motorboat would increase the realism of the boats' movements. Boats turn much slower than cars; an accurate motion model would create a more precise simulation of their movements. However, for the scope of the thesis, the use of boats with the motion model of a car is sufficient, considering the time constraints. Another limitation is that the pose sequences only generate movement with 3-DOF. Lastly, even though reactive agents generate exciting and important data for a SA system, it is not representative of how boats move in the real world. Creating more advanced path planning and navigation in the pose sequence generation would increase the quality of the simulation and subsequent synthetic datasets generated.

3.2.3 What Software Tools Exist to Create Synthetic Data for an ASV

To create semi-photo-realistic synthetic data, software tools such as game engines or other 3D modelling software can be used. Two of the most popular and advanced game engines are Unity and Unreal Engine. These can be used to generate advanced 3D projects with a high degree of photo-realism. Unlike Unity, the Unreal Engine is open-source but has a complex interface and a steep learning curve. Blender is a free and open-source 3D creation suite. It supports the entirety of the 3D pipeline—modelling, rigging, animation, simulation, rendering, compositing, motion tracking, and even video editing and game creation [Fou]. One of the major advantages of utilising Blender for generating synthetic data is that it has simulation libraries which can, for example, simulate complex water dynamics. Furthermore, Blender has an extensive Python API, allowing developers to generate simulations by scripting instead of utilising a user interface. With the possibility of writing scripts, the provided pose sequences can be more easily utilised to generate synthetic data. Due to the preceding reasons, Blender was chosen as our software tool. However, both Unity and the Unreal Engine are viable alternatives for synthetic data generation.

3.2.4 How to Use POSE Files for Synthetic Data Generation

The script which parses the pose sequences and uses Blender for visualisations is, from this point on, denoted as the synthetic data generator (SDG).

JSON Format

The provided pose sequences were stored as JSON files, and the format is further described in this section. Furthermore, a pose sequence is split into timeframes containing pose data for all the vehicles in the scene. A sequence contains information about a vehicle's velocity, steering angle, centre position, and heading direction. The units for the different parameters are radians for steering angle and heading direction, velocity is given as meters per second, and the centre position is given in meters. We were also provided with JSON files containing the characteristics of the vehicles and environment. Such as the dimensions of the vehicles, the environment and the vehicle type. All pose sequences follow the aforementioned structure, and an example of a pose sequence is shown below:

```json
{
  "time_s": {
    "0.0": {
      "vehicle0": {
        "normed_velocity_ms": 1.75,
        "steering_angle_rad": 0.0,
        "center_position_m": [
          35.0,
          20.0
        ],
        "heading_rad": 0.0
      }
    }
```

Code listing 3.1: An example of the pose sequence with the values of "vehicle0"

How we Parsed the JSON Files

To visualise the pose sequences with Blender, boxes were created and placed corresponding to the position and heading of the pose sequences. These boxes represent boats in the environment. Initially, the boats were created with the given dimensions and placed at the positions and heading corresponding to the initial timeframe. Then the positions and heading directions of the boats were updated for each timeframe. To generate a video sequence in Blender, a camera replaced the positions and headings of a boat. The ship equipped with the camera is denoted as the ego-ship in this thesis. Thus, the camera imitates a camera placed on a boat in real-life and can capture the movements of the other boats. Generating long video sequence sequences in Blender could be time-consuming and require sufficient computational power. Therefore, we created a script that divided the JSON file into an optional amount of uniform chunks. For instance, instead of generating a simulation with 10 000 timeframes, five simulations with 2000 timeframes can be generated. However, a computer with better computational power was acquired during the development process. Due to this, the pose sequences could be read in a single take. A simplistic maritime video sequence can be generated in Blender by utilising the provided pose sequences and scripting.

3.2.5 How to use Cameras for Pose Estimation in Blender

Blender has built-in camera objects which can be used to make animations. In our project, we decided to implement stereo cameras to generate a synthetic dataset containing stereo videos. This choice is further explained in section 3.3.2. In order to implement stereo cameras, two animations with cameras placed on the same boat with a given baseline (b) between them were created. Given the position of the boat (x, y) and the heading direction (θ), the position of Camera1 and Camera2 can be calculated. By setting the heading direction of the camera equal to the heading direction of the boat, the formula for calculating the camera positions is as follows:

$$\text{Camera1} = \begin{bmatrix} x + \frac{b}{2}\cos\theta \\ y + \frac{b}{2}\sin\theta \end{bmatrix} \text{ and Camera2} = \begin{bmatrix} x - \frac{b}{2}\cos\theta \\ y - \frac{b}{2}\sin\theta \end{bmatrix}$$

To make the camera capture more boats in the images, a focal length with a substantial field of view (FOV) was chosen for the cameras. We chose to have a focal length of 25 mm which gives us a FOV of 127 degrees. Focal length is defined as the distance from the optical centre to the lens and is described in Figure 3.1.

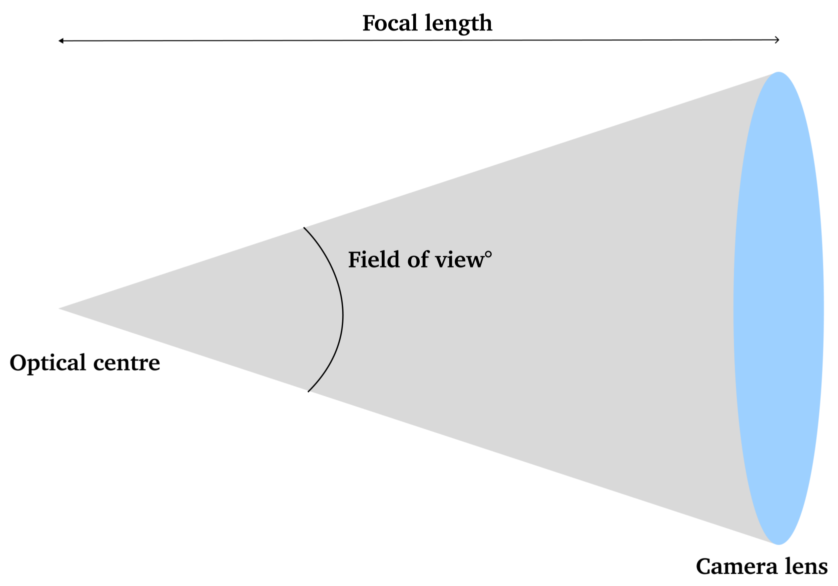


Figure 3.1: An illustration of a camera's FOV and focal length.

3.2.6 Creating the Visual Elements of the SDG

Once the Python code to import the pose sequences and instantiate objects for our desired vessels was implemented, we developed the SDG visual elements. First, we created 3D cubes in Blender to represent the vessels, with the corresponding width and height defined in the vehicle characteristics from the pose sequences. After the initial implementation, the synthetic data generator has undergone several iterations to replicate a busy port. The improvements are described in the following subsection.

Enclosing the Simulated Environment

After going through the video sequences, the cubes seemed to move around as expected, with many dangerous situations and near misses. The pose sequences are defined within a 100-meter x 100-meter cube, and the vehicles avoid crashing into the outer bounds of the cube. The vessels change directions to avoid these "invisible" walls; thus, we added walls in the simulator to make their movements more comprehensible. The roof was added to represent the sky in the real world and, combined with the walls, became a simulated port for our vessels.

Adding Realistic Texture to the Vessels

Moving cubes in an enclosed environment was not an adequate visual representation of our simulated port. Therefore, we decided to add texture to the boats to clarify the vehicle type of the objects in the simulator. To add texture to the boats, we utilised the Blender function:

```
bpy.ops.import_scene.obj
```

The function above renders 3D models as objects in Blender, allowing us to use free 3D models from the internet instead of making our own. Several websites have an abundance of 3D models for many different types of objects. We used the "Boat 3D model" from the website https://archive3d.net/, which can be seen in Figure 3.2, for each vessel in our simulator. When importing the 3D models to objects, the scale of the objects changes based on the size of the 3D model. Since the 3D model generated ships that were considerably larger than those specified in the pose sequences, the vessels' width and length had to be changed to align with the size definition in the pose sequences. Lastly, the pose sequences have no definition of height for the vessels. Therefore, the height was set to mimic an average cabin cruiser which is about 3 meters. The width and height in the JSON file for the pose sequences also correspond to an average cabin cruiser.

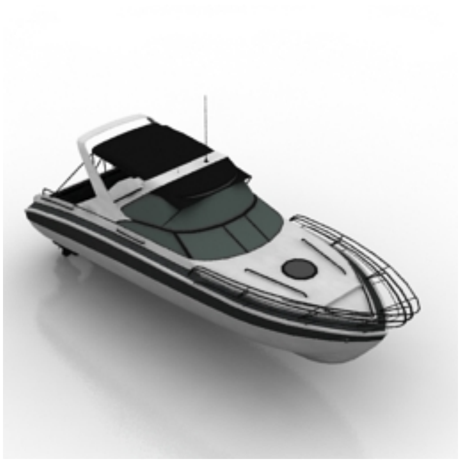


Figure 3.2: The 3D model used for the boat texture in the SDG.

Adding a Light Source to the Scene

A light source was added to the scene to emulate the sun in the real world. When applying the light to the synthetic data generator, the animations seemed more natural, and the light and shadows varied based on the locations of the objects in the scene as they would in the real world.

Importing Dynamic Water Material

After adding a light source to the scene, a visualisation of water was added to further improve the realism of the scene. As mentioned in subsection 3.2.3 Blender has implementations of water dynamics which can be used for simulations. However, due to the vehicle movements originating from pose sequences, the vehicles will not be affected by any motion created by the water dynamics. Therefore, the water added to the simulator is only a cosmetic addition to the scene. If the synthetic data generator were to include water dynamics, then it would rely on updating the pose sequences accordingly. As seen in Figure 3.3 the vessels look stranded on land without a visualisation of water.

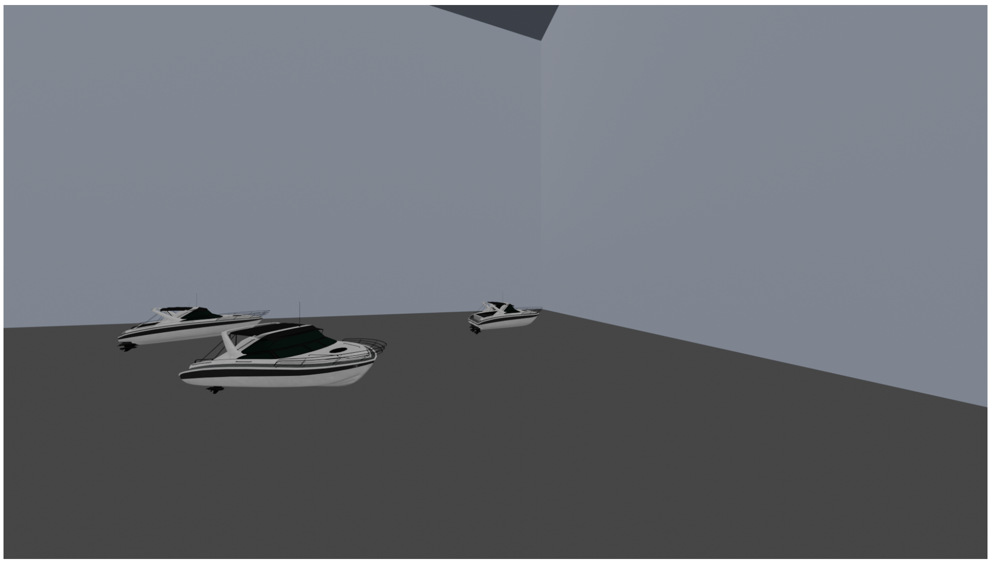


Figure 3.3: A scene generated by the SDG without water.

After adding the water texture from Blender, which can be seen in Figure 3.4, the boats appear submerged in the water, and the scene's appearance is much more impressive. Modifying some of the parameters in the water texture generated a semi-realistic ocean in terms of colour and choppiness of the waves.

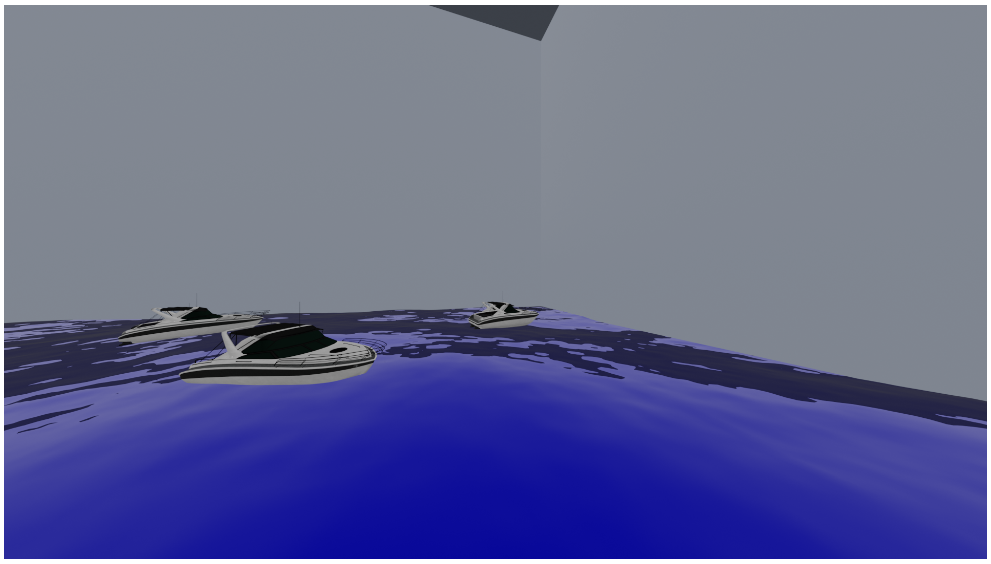


Figure 3.4: A scene generated in by the SDG with water.

Why we Decided Not to Add a Realistic Background

Adding a realistic image of, for example, a port with ships on the walls of our scene would have at least two advantages. Firstly, it increases the realism of the generated data's appearance. Secondly, it also enhances the difficulty for object detectors that use data generated from the generator. However, we are content with the degree of realism as can be seen in Figure 3.4 and do not believe that adding this detail improves the generated data by as much as the effort needed to add the background. Furthermore, even though adding a background would increase the robustness of an object detector trained on the synthetic data, the scope of the thesis is about creating a SA system and not about testing the performance of current state-of-the-art object detection methods utilising synthetic datasets.

3.3 Defining the Architecture of our ASV SA System

3.3.1 What Modules are Needed for the First-Level ASV SA System?

This section will present the architecture for our ASV SA system. As stated in subsection 2.3, an ASV can benefit from implementing a system which correlates to the first level of SA. To achieve a system capable of the first level of SA, the problem could be divided into sub-tasks. Firstly, the system must be able to sense its environment by utilizing some sensors. Then it must be able to detect and possibly classify objects around it. Furthermore, detected objects should be tracked over time, and their positions should be estimated in 3-DOF. The rest of the section is structured after the modules needed in the architecture to meet the criteria of a first-level SA system.

3.3.2 What Type of Sensor Data Should the SA System Use?

For our architecture, we chose to work solely with sensor data from visual sensors as defined in section 1.2. The stereo image frames are generated from the synthetic data generator described in section 3.2. Lastly, we decided to use stereo images as input instead of monocular images because stereo depth estimation is more accurate and easier to implement than monocular depth estimation, as stated in section 2.8.

3.3.3 How to Detect and Classify Surrounding Objects?

We wanted to utilize a state-of-the-art object detector for detecting objects in images, given that we are utilizing stereo images as the input for the ASV SA system. Therefore, we used the latest version of the YOLO object detection algorithm, namely YOLOv8.

3.3.4 How to Track Detected Objects?

After detecting objects, we wanted the system to keep track of the identities of the detected objects over a sequence of images. To solve this problem, we chose the state-of-the-art tracking algorithm ByteTrack due to its impressive performance and well-written documentation for implementation.

3.3.5 How to Estimate the Distance and Bearing of the Tracked Objects?

Then, stereo depth estimation is performed to predict the distance to tracked objects in the stereo images. And lastly, the bearing angles of the tracked objects are estimated.

3.3.6 The Final Architecture

The final architecture for our ASV SA system can be seen in Figure 3.5. In the following sections, detailed descriptions of the development and implementation process of the modules in the architecture are presented.

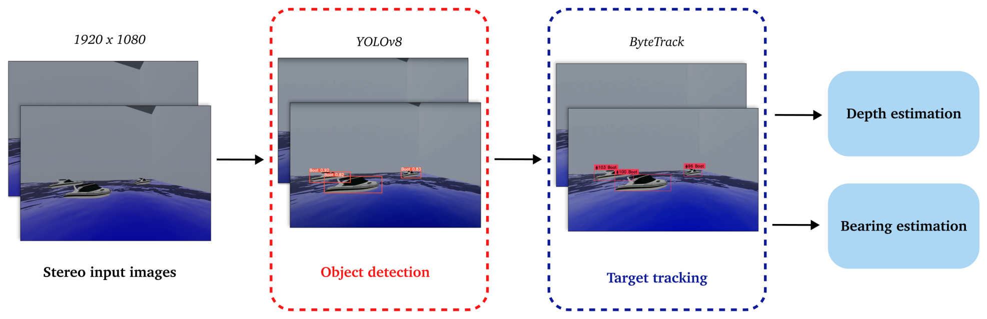


Figure 3.5: The final ASV SA architecture.

3.4 How we Used YOLOv8 for Object Detection

3.4.1 How Should we Generate Bounding Boxes?

For the SA system, we need AABBs to locate the position of boat detections in the images generated from the SDG. These AABBs are essential for the latter part of our thesis, where we want to create a dynamic representation from detections. From the provided pose files, the positional data of each ship in each timeframe are accessible. The aforementioned positional data could generate perfect AABBs without noise. However, this would limit the benefits of the results. In order to get more realistic AABB, one could generate AABB with artificial noise or use an actual object detection method on the generated data. Evaluating artificial noise is difficult and time-consuming while utilizing an actual object detection method would ensure realistic results. Therefore, we decided to use YOLO to make the system more applicable to real-world situations.

3.4.2 Initial Model of YOLOv8

We used the newest version of YOLO, which is YOLOv8. YOLOv8 is pre-trained on the MS COCO dataset and can detect various objects without any additional training. We installed a stable version of YOLOv8 and tried detecting the boats on some images from our simulated environment. However, the model could not identify any of the ships in the images. This might be due to a limited degree of realism in the texture of the boats in the generated data or limitations in the pre-trained model. In order to solve the problem, we decided to train a model on a dataset with images from the generated data.

3.4.3 Creating a Synthetic Maritime Dataset

To train a custom YOLOv8 model, it requires image data along with annotated AABBs following the MS COCO annotation format. Therefore we chose to create a dataset utilizing the computer vision framework Roboflow. Roboflow provides tools for the different steps of converting raw images into custom-trained computer vision models [Rob23]. The dataset consists of 41 images from the SDG. To improve the training of YOLOv8 model, we split the dataset into 28 training images, eight validation images and five test images [rec23]. Utilizing Roboflow's annotation tool, we manually drew the bounding boxes around the ships in the images. The dataset only has one class type, defined as "*Boat*". The number of ships, rotation and occlusion varies in the images. However, the dataset is quite limited due to the number of images. Increasing the number of images can be easily done, given the number of frames the simulation produces. However, manually annotating images is a time-consuming task.

3.4.4 Training a YOLOv8 Model on the Synthetic Maritime Dataset

The model was trained on the synthetic maritime dataset to achieve an adequate performance on the data from the SDG. The model was trained on the synthetic maritime dataset to achieve an adequate performance on the data from the SDG. To download the synthetic maritime dataset, the Roboflow API was utilized. The custom model was trained on the synthetic maritime dataset for 100 iterations. After validating the validation images from the synthetic maritime dataset using the best weights, the model proved to be very accurate. The precision of the AABB was 0.978 and had a recall of 0.958. The precision, recall and mAP scores for the YOLOv8 model trained on the synthetic maritime dataset can be seen in Figure 3.6. The x-axis denotes the training epoch, and the y-axis denotes a value between 0 and 1. The model achieved a mAP50 score of 0.992 and a mAP50-95 score of 0.795. The predicted AABBs for some of the validation images can be seen in Figure 3.7.

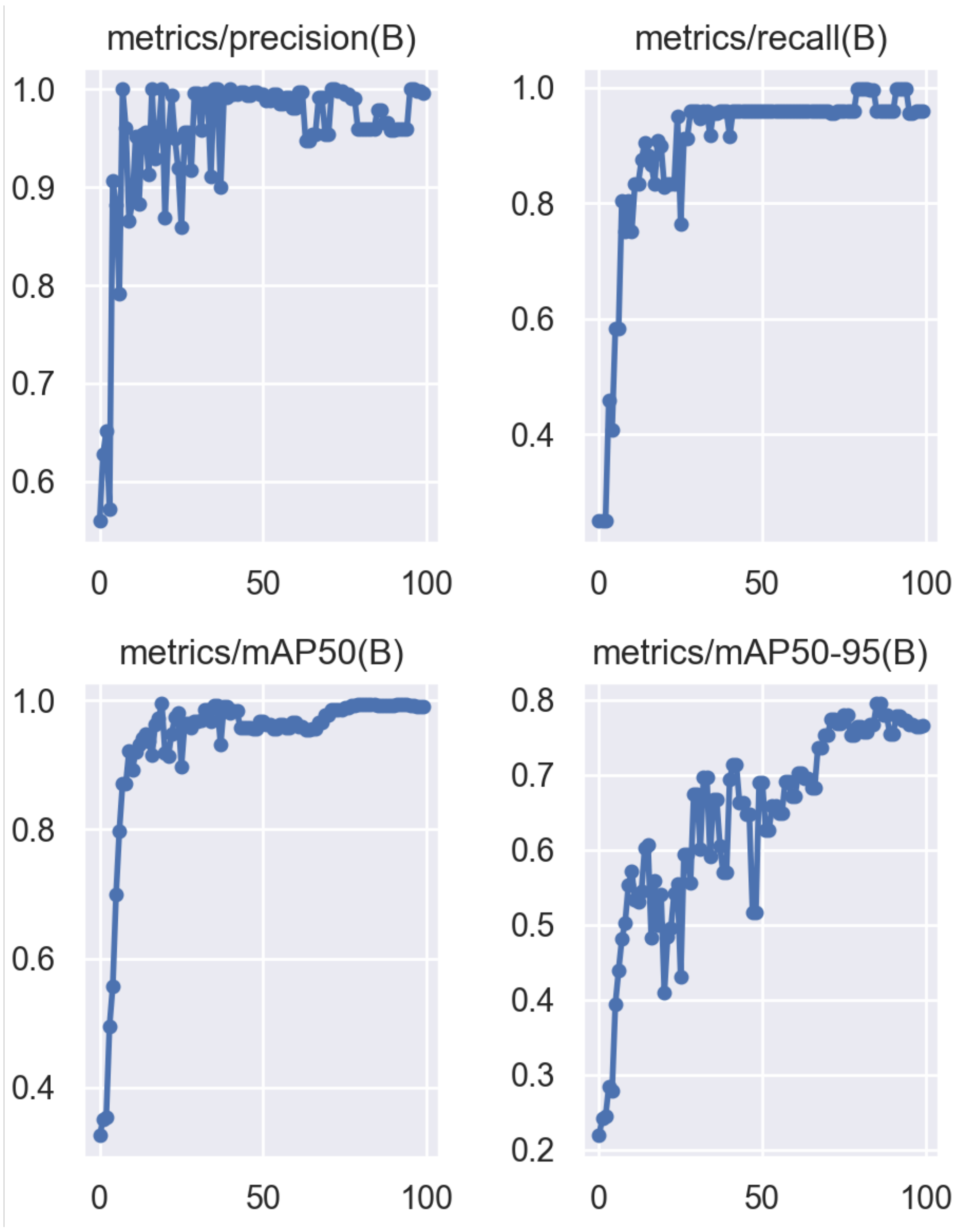


Figure 3.6: Precision, recall and mAP scores for the the YOLOv8 model.

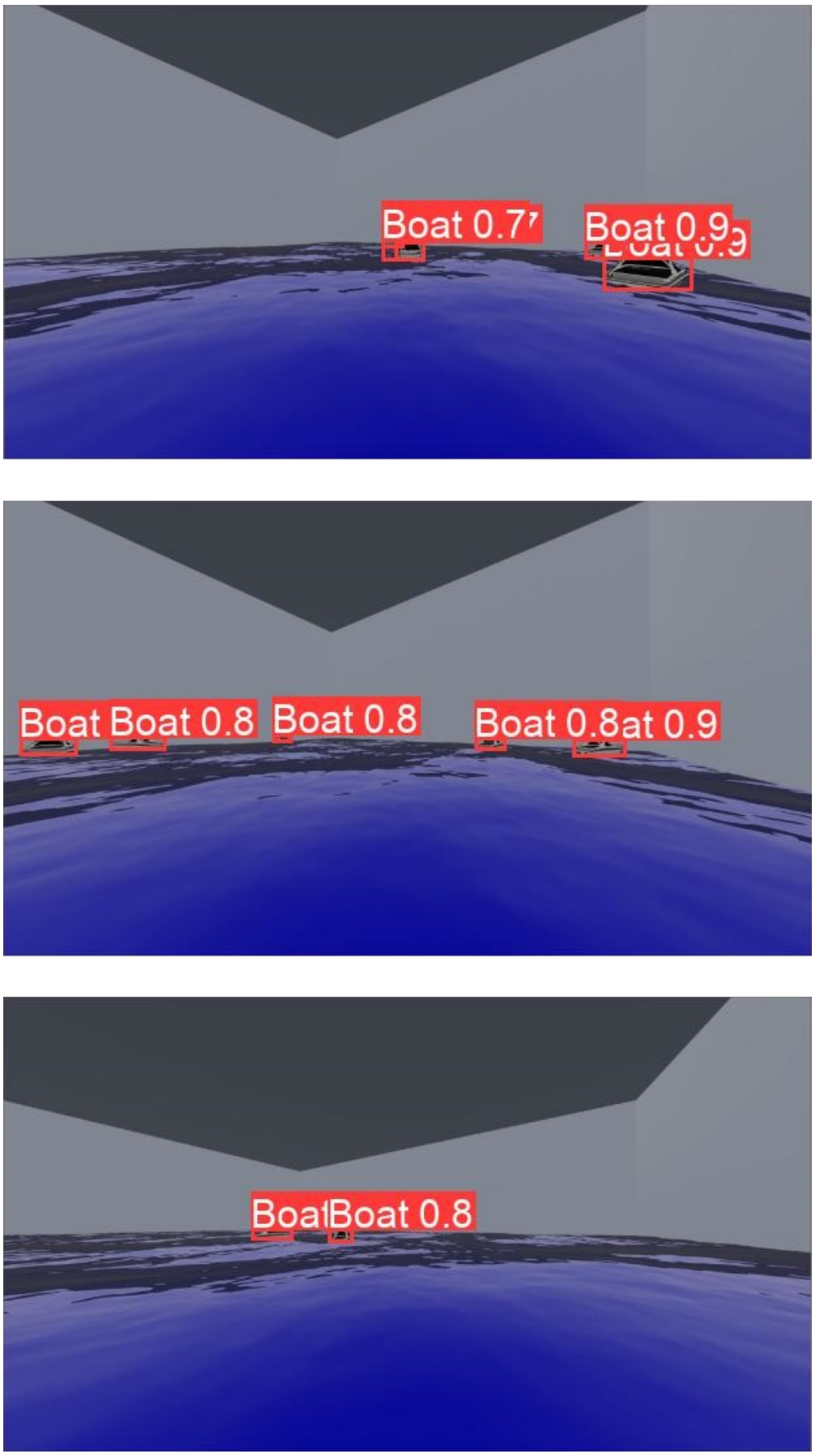


Figure 3.7: Predictions made on some of the validation images from the synthetic maritime dataset.

Given the limited size and variation of the synthetic maritime dataset, it is clear that the YOLO model is likely to overfit during training. Furthermore, we tested the best model on the test set, and it produced satisfactory results, detecting all the boats in the images with a high degree of certainty. However, by only testing on five images, a proper evaluation of the model can not be performed. Furthermore, the results could be deceiving and hide flaws in the robustness of the model. However, a qualitative evaluation was performed to ensure that the model performs accurately and reliably for the intended use. The model was given a video sequence consisting of 6000 unannotated images from the synthetic maritime dataset to generate AABBs for a qualitative evaluation of the model. As previously mentioned, the model should generate detections for most of the synthetic ships with a varying degree of certainty. When evaluating the quality of the model for the unannotated images, the model accurately predicted most of the boats with a high degree of certainty. Thus, the model is adequate for our intended use and the scope of the thesis. However, there are many aspects in which the model can be improved, which is further described in the following subsection.

3.4.5 How the YOLOv8 Model Could Be Improved

As previously stated, annotating more images and adding them to the dataset would definitely increase the robustness of our model. Furthermore, the simulated environment only has one type of boat with the same physical attributes for each instance, and it could be improved by varying the appearance and type of naval vessel in the environment. Furthermore, by changing the hyperparameters of the YOLOv8 model, the model could generate better results. However, as the scope of this thesis is not to evaluate the performance of object detectors such as YOLOv8 or create a high-quality synthetic dataset, we consider that the model is sufficient to generate AABB for the synthetic data generated from the SDG.

3.5 Using Bytetrack to Track Detections Over Multiple Frames

3.5.1 Implementing Tracking on YOLOv8 Generated AABBs

To fulfil the tracking module for our ASV SA system, the Bytetrack algorithm was used on the detections from the YOLOv8 model. After generating the AABBs for the video sequence mentioned in section 3.4.4, Bytetrack was used to track detections across frames. Since Bytetrack can track AABBs, utilizing it for tracking from the YOLOv8 model was pretty straightforward. To create visual AABBs similar to what the YOLOv8 model produces, the Roboflow supervision library was utilized. The Roboflow framework supervision library has built-in functions for visualizations of AABBs, video loading with meta-data generation, and generating videos with visualizations. We combine the input images with the tracked AABBs to create a video output to review. The output is now similar to the YOLOv8 model with additional information about the track identification of tracked detections.

An example of a Bytetrack output image can be seen in Figure 3.8.

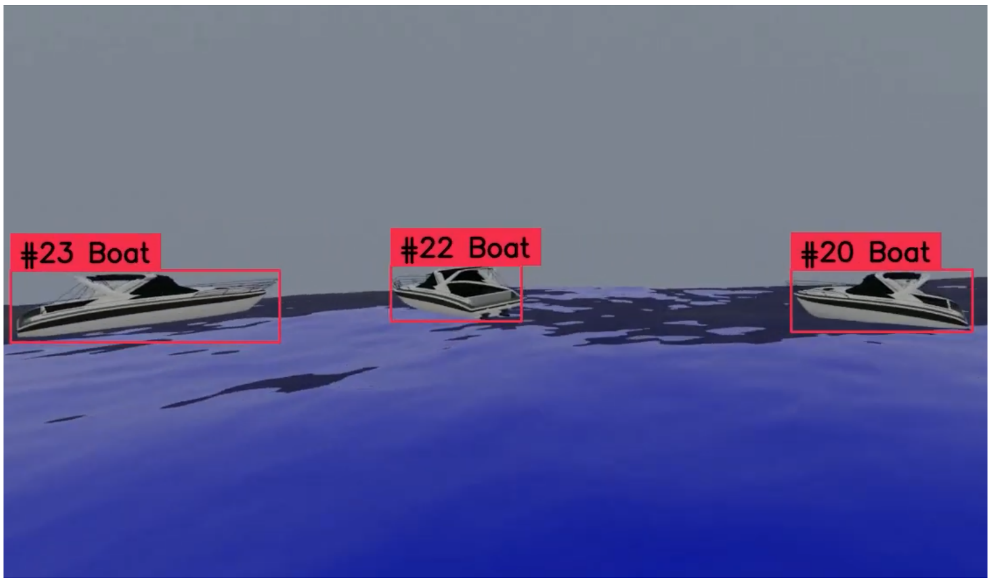


Figure 3.8: An output image from the Bytetrack algorithm.

3.5.2 Results of ByteTrack on Blender Simulator

After having a qualitative evaluation of the tracked sequence, the results seemed sufficient. When reviewing the output video, tracks are usually kept alive as long as the tracked object remains in the frame. However, occlusion is a common problem in object detection and tracking, and after evaluating the tracked sequences, it is clear that the Bytetrack model used is prone to occlusion. Nevertheless, there are in general very few identity shifts even when vehicles are crossing paths. Thus, given the scope of our thesis, the tracker works as intended and displays intelligent tracking behaviour and will not need any alteration from our initial implementation. Stereo videos generated from the SDG were used as input for the Bytetrack model, and the output videos can be seen at these links: https://www.youtube.com/watch?v=RnsbJGpP5wI, https://www.youtube.com/watch?v=ynenJbrcqTE.

3.5.3 Limitations of Tracking Input

ByteTrack utilizes the visual features of objects to compute the ReID scores [Zha+22]. Thus, by using a synthetic dataset where each boat has an identical appearance, we limit the capabilities of the tracking algorithm. Furthermore, tracking would most likely improve by having a different texture for each boat, as Bytetrack use feature distance and IoU for tracking. However, after having a qualitative evaluation of the tracked sequence, the results seemed sufficient. Due to the limited time of the project, we decided that bookkeeping of identities when a tracked object exits and reenters the scene is outside the scope of the project.

3.6 Generating Positional Ground Truth for the Pose Sequences

To evaluate the ASV SA system for all the ships in the environment, positional ground truth is necessary. However, the ground truth should be in relation to the ego-ship as the SA system uses the ego-ship as the origin point as illustrated in the right figure in Figure 3.9. This section describes the generation of positional ground truth relative to the ego-ship.

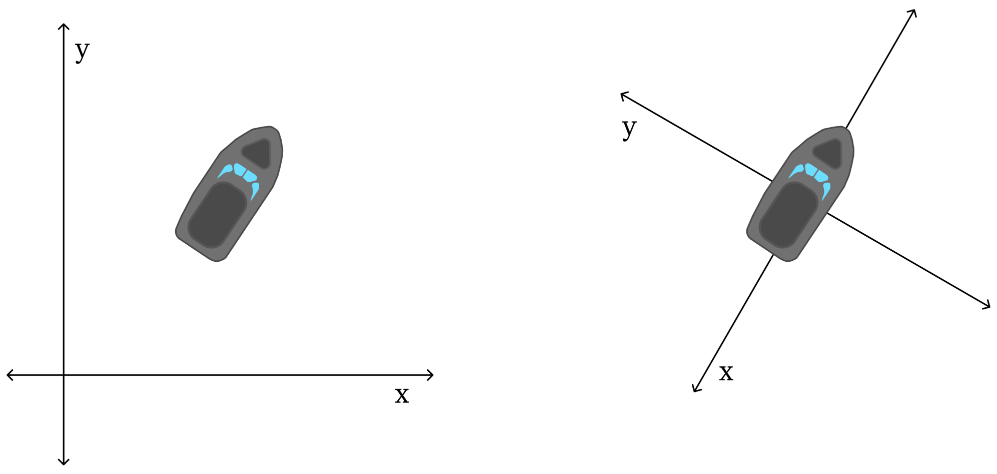


Figure 3.9: An illustration of the difference between a NED frame (left) and a frame relative to an ego-ship (right).

3.6.1 Converting the Pose Sequences From a NED Frame to an Ego-Ship Coordinate Frame

The following conversion process can be used to calculate the relative position of a ship to the ego-ship. If the coordinates of a vessel are given as (x_b, y_b), and the coordinates and heading direction of the ego-ship are given as (x_c, y_c) and θ_c. To convert a coordinate of the vessel in the NED frame to be relative to the ego-ship, the following formula is used:

$$x_{NEW} = (x_b - x_c) \cdot \cos(-\theta_c) - (y_b - y_c) \cdot \sin(-\theta_c)$$
$$y_{NEW} = (x_b - x_c) \cdot \sin(-\theta_c) + (y_b - y_c) \cdot \cos(-\theta_c)$$

Applying this formula to the pose sequences generates the positional ground truth relative to the ego-ship.

3.6.2 Conclusion

Based on these formulas, a script was made to generate ground truth values for all the ships in the pose sequences for each timestep. These ground truth values can then be used to evaluate the performance of the ASV SA system. This data is necessary for evaluating the ASV SA system but also adds value to the SDG of the thesis.

3.7 Association of AABB in a Stereo Image

In order to estimate a bearing and depth of an object in a stereo image, it is necessary to know which object in the left image corresponds to which object in the right image. In this section, the initial association method utilizing bearings is presented. Then a IoU based association method was tested, and the two methods were compared.

3.7.1 Using Bearings to Associate AABBs

The AABB with the lowest bearing in the left image was associated with the AABB with the lowest bearing in the right image. Furthermore, the AABB with the second lowest bearing in the left image is associated with the second lowest in the right, and so on for all the AABBs. When testing our depth and bearing estimation methods, we noticed significant drawbacks when associating AABBs this way. The association method could not handle an uneven number of objects between the images. In object detection, clutter and miss detections can be expected, which the association method does not take into account. Furthermore, in the case of stereo images, it is also possible for an object to be in the FOV of only one of the cameras. Thus, an uneven amount of objects between the images is not uncommon.

3.7.2 Using IoU to Associate AABBs

In stereo images, most objects have similar positions and sizes in both images. In order to utilize the similarities, we looked at the IoU to associate the AABB in the two images.

3.7.3 Comparing AABB Association Methods in a Stereo Image

To compare using IoU with the method using bearings, the performance of the bearing and depth estimation were examined. The results are shown in Table 3.1 and table 3.2.

The tables show the precision of the bearing and depth estimates within different thresholds for the two association methods. The two association methods perform quite similarly for bearing and depth estimations. However, the results of the bearing estimation do significantly better using the IoU method for a low threshold. This indicates that the objects are associated correctly between the two

	Threshold			
	0.1	**0.05**	**0.01**	**0.001**
Bearing	90.2%	68.7%	17%	1.7%
iou	90.2%	69.2%	18.8%	18.8%

Table 3.1: An overview of how bearing estimation did when associating images with corresponding bearing and IoU.

	Distances				
	0m-20m	**20m-40m**	**40m-60m**	**60m-80m**	**80m-**
Bearing	67.3%	80.6%	70.9%	57.5%	26.5%
IoU	62.6%	78.5%	66.8%	48.1%	19%

Table 3.2: An overview of how depth estimation did when associating images with corresponding bearing and IoU with a threshold of 4 meters.

images and with the correct ground truth. Thus, the IoU method was chosen to associate the AABBs in the stereo images for our SA system. The methods used for bearing and depth estimation are described in the following sections.

3.8 Different Methods for Bearing Estimation

Bearing estimation, in the context of this thesis, is defined as estimating the angle of an object relative to the heading direction of the ego-ship. The bearing (θ) is a number between $-\pi$ and π, and the sign of the number indicates which side of the ego-ship the object is on. To calculate the horizontal bearing of a pixel (p_x, p_y) relative to the center of a image (c_x, c_y) with a given width (w), the following methods were compared:

3.8.1 Method 1

The first method calculates the arctan value of the horizontal distance between the pixel and the centre of the image. Divided by the vertical value of the pixel:

$$\theta = \arctan \frac{c_x - p_x}{p_y}$$

3.8.2 Method 2

The second method explored was used to calculate the provided bearing in the MSFBD dataset[Hel+22]. First, the horizontal FOV is calculated using the width (w) of the image and the horizontal focal length (f_x) with the following formula:

$$FOV_x = 2 \arctan \frac{w}{2f_x}$$

Then, to calculate the bearing of a pixel relative to the centre, the following formula is used:

$$\theta = \frac{p_x - c_x}{w} FOV_x$$

3.8.3 Method 3

For the third method, we looked at how we could use the parameters provided by Blender to calculate the bearing. First, we looked at using the provided FOV (FOV_D) instead of calculating it with other parameters as in method 2. The FOV is provided in degrees and in order to use the FOV, we convert it to radians (FOV_R) by:

$$\text{FOV}_R = \frac{\text{FOV}_D \cdot \pi}{180}$$

In order to calculate the bearing of a given pixel where the pixel is relative to a principal point, which is the centre of the image. The formula for Method 3 can be seen below:

$$\theta = \frac{(p_x - c_x) \cdot \text{FOV}_R}{w_i}$$

3.8.4 Method 4

Blender also provides information about the width of the camera lens (w_c) and the focal length (f) in millimetres. If we convert the x-coordinate of the pixel to millimetres by using this formula::

$$x = \frac{p_x \cdot w_c}{w_i}$$

Then we can calculate the bearing can be calculated by using the following formula:

$$\theta = \arctan \frac{x}{f}$$

3.8.5 Comparing the Methods

At first, the centre of the AABB was used to calculate the bearing, but after testing and comparing the results, the bottom centre of the bounding box was used instead. A comparison of the results using the centre and bottom centre to calculate the bearing with Method 1 is in Table 3.3. The results show that using the bottom centre generates better bearing estimations for Method 1. The rest of the bearing estimation methods do not consider the height when estimating the bearing. When testing, the number of bearing estimates from the different methods within different thresholds was considered. To compute the accuracy of the bearing estimation methods, the estimates were associated with the closest ground truth.

	Threshold			
	0.1	**0.05**	**0.01**	**0.001**
Centre	53%	24.1%	4.8%	0.5%
Bottom centre	67.8%	30%	6.6%	0.5%

Table 3.3: Comparison of using the centre and bottom centre of the AABBs by looking at how many of the estimates are within different thresholds.

When comparing all methods, the bottom centre of the AABBs were used for the bearing estimations. The results for all the methods are shown in Table 3.4. The results indicate that Method 4 performs significantly better than the other bearing estimation methods.

	Threshold			
	0.1	**0.05**	**0.01**	**0.001**
Method 1	67.8%	30%	6.6%	0.5%
Method 2	67.4%	35.7%	7.6%	0.9%
Method 3	33.4%	13.9%	3%	0.3%
Method 4	90.5%	69.5%	18.8%	1.9%

Table 3.4: Comparison of how many estimates are within different thresholds for the different bearing estimation methods.

All the methods were also evaluated with the RMSE evaluation metric. The results are presented in Table 3.5. These results from Table 3.4 concur with the results from Table 3.5 that Method 4 outperforms the other bearing estimation methods. As a result, we chose to use Method 4 for our final architecture.

Method	**RMSE**
Method 1	0.1255
Method 2	0.1304
Method 3	0.1707
Method 4	0.0715

Table 3.5: Overview of the different RMSE values of the different bearing estimation methods.

3.9 Different Methods for Depth Estimation

As stated in subsection 3.3.2 we will perform stereo depth estimation to estimate the depth of detected objects. Many well-performing methods use a depth map for depth estimation. However, when creating a depth map, every pixel in the image from the left and the right camera is used. This includes mostly irrelevant information like water or sky. In order to avoid this overhead, we try to get all the necessary information from the AABBs predicted by the YOLOv8 model. We associate the AABBs in the images from the left camera with the AABBs in the right camera and calculate depth based on the centre of the AABBs in the two images. After calculating the depth, the depth estimations are compared to the ground truth. In this section, two methods used to estimate objects' depth are described and compared.

3.9.1 Method 1: Calculating the Depth Using the Disparity, Baseline, and Focal Length

The first depth estimation method utilizes the disparities of the pixels and the baseline and focal length of the camera for stereo depth estimation. The baseline (B) given as the distance between the cameras and the focal length (f) can be obtained in Blender. Whilst the disparity (B) can be calculated as the distance in pixels along the x-axis of a point in the left image and a point in the right image. An illustration of what the different values represent is shown in Figure 3.10. To calculate the depth (z) of an object, the following formula was used:

$$z = \frac{f \cdot B}{D}$$

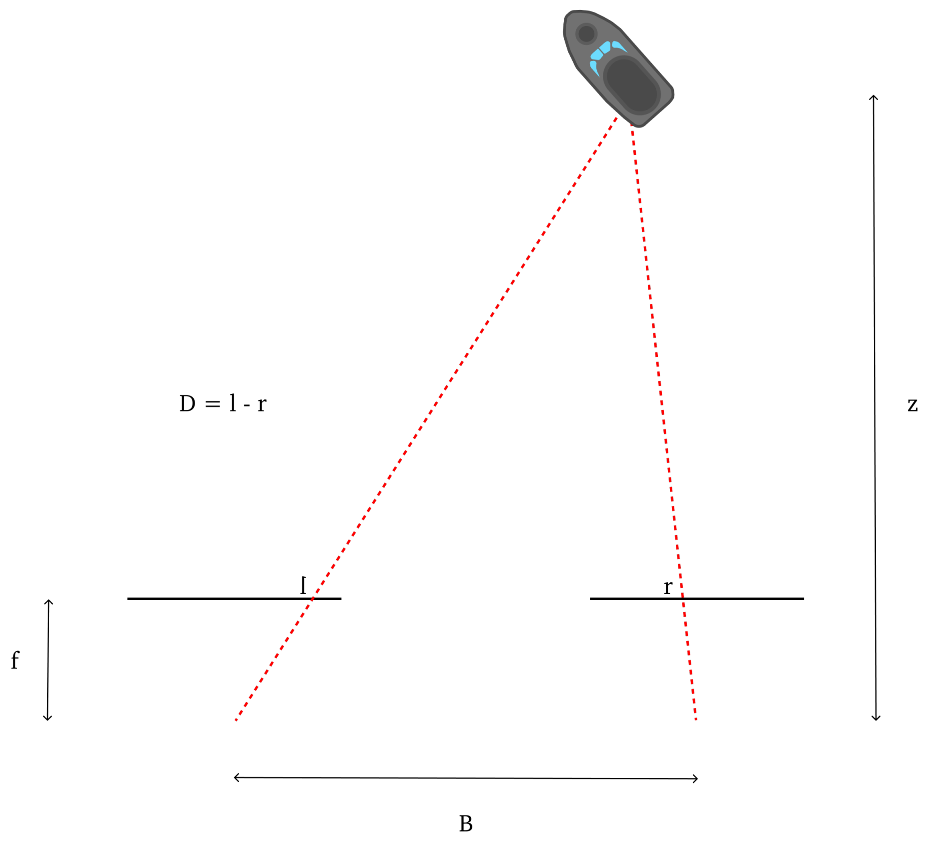


Figure 3.10: An overview of the different variables used to calculate the depth of a detection.

When testing Method 1, it seemed to estimate the depth of each object accurately relative to each other, but the depth estimates were inaccurate. We added a constant (c) to adjust the result to better fit the values of 5 test images. We used $c = 10$ in our system. The final formula is:

$$z = \frac{f \cdot B}{D} \cdot c$$

3.9.2 Method 2: Calculating the Depth Using the Baseline and the Distance and Angles Between the Cameras and the Object

Another depth estimation method is to calculate the depth based on the baseline and the distance and angles between the cameras and the detected object. If the bearing in the left (θ_l) and right (θ_r) camera can be calculated, then we can calculate the angle (θ_x) between the line from the centre of an object to the right camera and the line from the centre of the object to the right camera. An illustration of what the different values represent is shown in Figure 3.11.

$$\theta_x = \pi - \theta_l - \theta_r$$

Then we can combine the known baseline (B) with the angles to calculate the distances to the object from the cameras by using the law of sines:

$$A = \frac{B \cdot \sin\theta_l}{\sin\theta_x}$$

$$C = \frac{B \cdot \sin\theta_r}{\sin\theta_x}$$

In order to calculate the area of the triangle, we first calculate the semi-parameter (s).

$$s = \frac{A + B + C}{2}$$

Then we can use it to calculate the area of the triangle.

$$Area = \sqrt{s(s - A)(s - B)(s - C)}$$

We then use the area to calculate the height of the triangle, and in this case, the depth of the object:

$$Area = \frac{1}{2} B \cdot z \implies z = \frac{2 \cdot Area}{B}$$

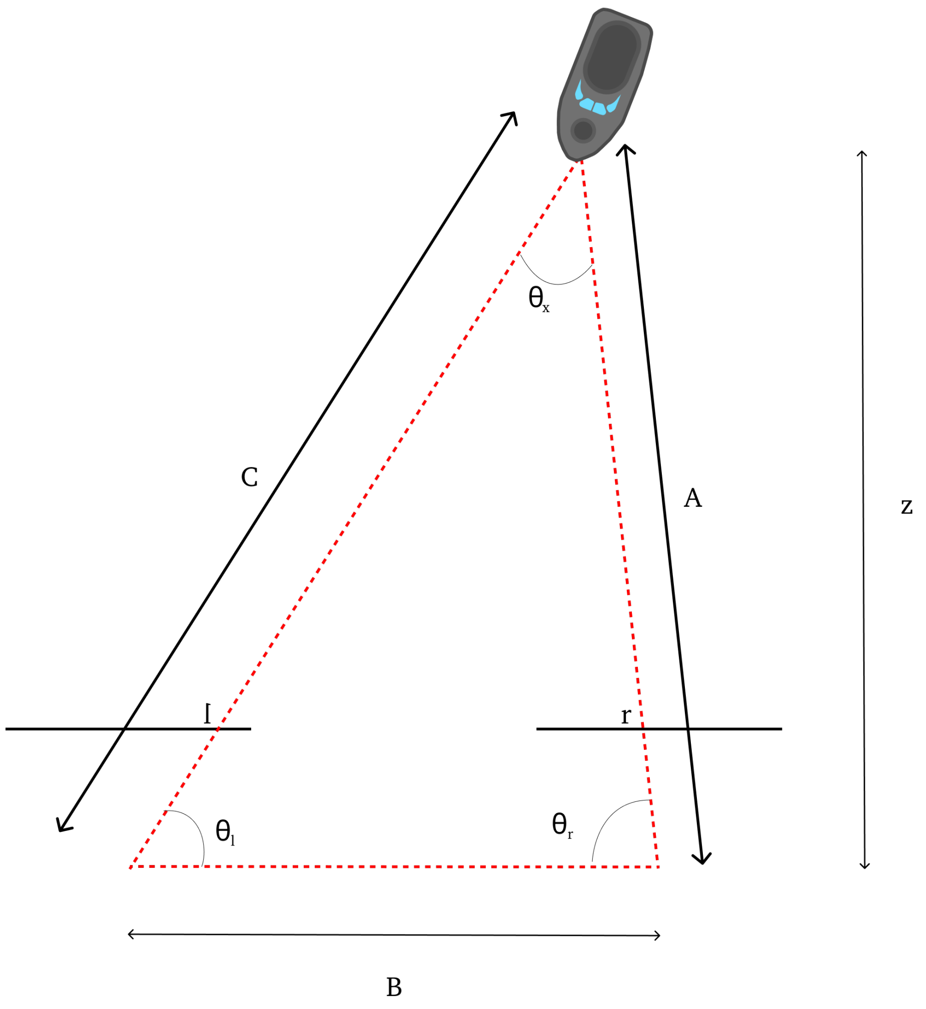


Figure 3.11: How the detection is represented as a triangle to calculate the depth.

3.9.3 Comparing the Methods

Initially, the SDG utilized a baseline of four meters when generating the videos. We compared the depth estimation results using the initial method described in subsection 3.9.1 using a baseline of four meters and 20 cm. The results are presented in Table 3.6. When analyzing the results, we see that using 20 cm gives a more accurate estimate of the depth of the object. Using a baseline of four meters seems to work well on objects within short distances. However, when we looked at the predictions, we saw that using a baseline of four meters resulted in way fewer predictions than using 20 cm. When using a greater baseline, the bounding boxes will often have a greater distance, and the intersection of the two camera views will be narrower. We use IoU to associate the AABBs from the left image with the right image. This makes the system using a greater baseline able to predict the depth of fewer objects. We decided to compare each estimate with the closest object.

	Baseline=4m			Baseline=20cm		
	6m	**4m**	**2m**	**6m**	**4m**	**2m**
0m-20m	93.1%	85.7%	45.4%	67.3%	62.6%	54.5%
20m-40m	46.4%	33.9%	14.5%	84.5%	78.5%	58.8%
40m-60m	9.4%	4.2%	4.2%	81.4%	66.8%	41.8%
60m-80m	2.2%	2.2%	0%	62.5%	48.1%	26.3%
>80m	4.8%	4.8%	2.4%	27.8%	19%	7.6%

Table 3.6: Precision of the depth estimations, with thresholds of 6, 4 and 2 meters, for different ranges and baselines.

When comparing Method 1 and Method 2, we see that Method 1 outperforms Method 2. However, if we find a more accurate way to estimate the bearing of the object, the method could perhaps compete with method 1. The results are presented in Table 3.7

	Method 1			Method 2		
	6m	**4m**	**2m**	**6m**	**4m**	**2m**
0m-20m	67.3%	62.6%	54.5%	61.2%	44.9%	21.7%
20m-40m	84.5%	78.5%	58.8%	1.6%	0.4%	0%
40m-60m	81.4%	66.8%	41.8%	25%	25%	0%
60m-80m	62.5%	48.1%	26.3%	3.2%	3.2%	1.6%
>80m	27.8%	19%	7.6%	2.7%	12.2%	1.7%

Table 3.7: Precision of the depth estimations, with thresholds of 6, 4 and 2 meters, for different ranges and methods.

3.10 From Depth and Bearing Estimation to Coordinate Prediction

A crucial aspect of the SA system is to estimate the location of detected objects accurately. To determine the position of a detection relative to the ego-ship, we can use the values of bearing (θ) and depth (z) to calculate the x- and y-coordinates. The relative detection coordinates (x, y) are calculated with this formula:

$$x = z$$

$$y = \tan\theta \cdot z$$

To evaluate the coordinate predictions, we calculated the distance to the ground truth values and assumed a prediction is associated with the closest ground truth object. The results are shown in the histogram in Figure 3.12.

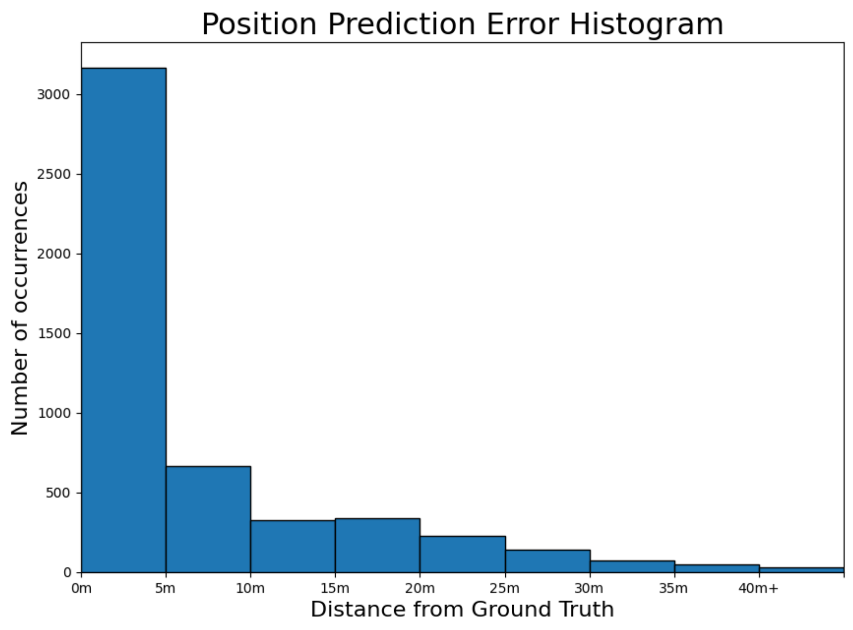


Figure 3.12: The errors of the predictions compared to the ground truth.

When looking at the result, we saw some outliers, but most predictions were within five meters of a target. We analysed the coordinate prediction errors with different bearings and distance thresholds to see if there were any correlations in the data. We started by looking at the bearing. We divided the field of view into three zones as described in Figure 3.13. Zone 1 corresponds to the FOV angles between -0.369 to 0.369. Zone 2 corresponds to angles between -0.738 to -0.369 and 0.369 to 0.738. Lastly, Zone 3 corresponds to the angles between -1.107 to -0.738 and 0.738 to 1.107.

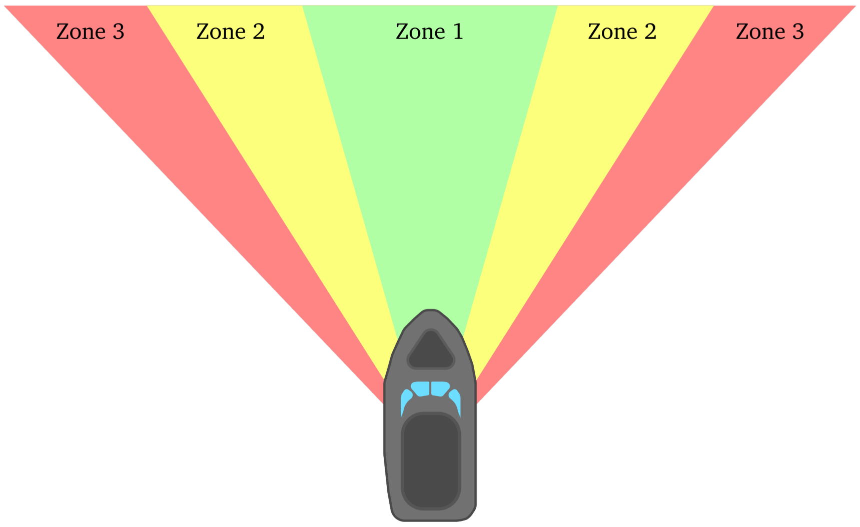


Figure 3.13: An illustration of how detections are divided into different bearing zones.

The precision of the predictions, with different thresholds, for each zone is presented in Table 3.8

	Threshold				
	2m	**4m**	**6m**	**8m**	**10m**
Zone 1	33.9%	69.2%	84.6%	90.6%	93.6%
Zone 2	40.9%	70.5%	81.5%	87.2%	90.1%
Zone 3	19.3%	40.4%	49.6%	53.7%	56.8%

Table 3.8: An overview of how many of the coordinate estimates were within different thresholds within the different bearing zones.

The results show a significant difference for the different bearing zones. The predictions associated with targets with low bearings seem to do considerably better than those with high bearings. When looking at some cases where the prediction is associated with a target in Zone 3, we often see that parts of the detected boat are outside the FOV. The ASV SA architecture does not separate between detections where only a part of the boat is present and will treat the resulting AABB interchangeably with a AABB covering an entire ship. This will naturally lead to some inaccurate predictions. Those predictions could be filtered out by checking if parts of the AABB are at the edge of the FOV. After looking at the predictions with different bearings, predictions of targets at different distances were analysed. The results are shown in Table 3.9.

	Threshold				
	2m	**4m**	**6m**	**8m**	**10m**
0m - 20m	35.6%	51.7%	53.6%	54.5%	56.7%
20m - 40m	37.1%	66%	72.8%	74.7%	77.1%
40m - 60m	25.7%	59.4%	76.3%	84.4%	87%
60m - 80m	10.9%	33.8%	52%	61.2%	67.3%
80m -	5.8%	18.3%	29.4%	38.8%	43.8%

Table 3.9: An overview of how many of the coordinate estimates were within different thresholds within the different ranges.

The results suggest that the targets closest to the ship give the most accurate predictions.

3.11 Visualising Predictions as a Bird's-Eye View

For the usage and development of autonomous vehicles visualising the environment in a bird's-eye view (BEV) can make it easier for humans to analyse the system's performance and explain how the system views the world. Therefore, creating a BEV visualisation increases the interpretability of the synthetic data input and the SA system output.

3.11.1 Generating a BEV Plot for a Given Timeframe

Firstly, the ego-ship's position was plotted as the origin point of the plot. The plot then corresponds to the ego-ship coordinate frame seen in the right figure in Figure 3.9. Then, for a given timeframe, the corresponding ground truth positions of the boats in the generated data in relation to the position of the ego-ship were added to the plot. Lastly, the predicted coordinates for the given timeframe, as described in section 3.10, are plotted. An example of a plot can be seen in Figure 3.14.

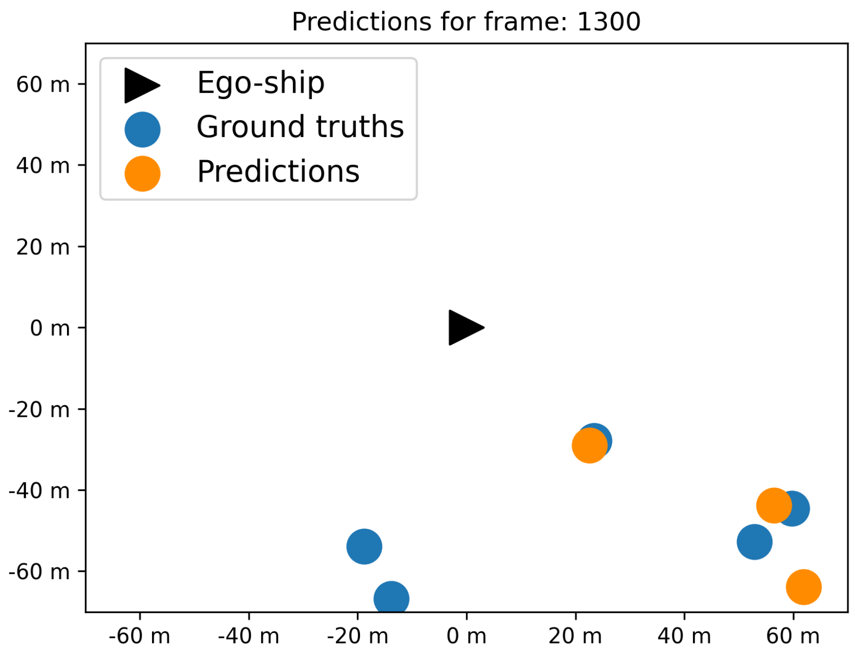


Figure 3.14: BEV plot for timeframe 1300.

3.11.2 Adding the Ego-Ship's FOV to the BEV Plot

As seen in Figure 3.14 there are no predictions for the ground truth positions behind the ego-ship. There are several factors that lead to missing detections. However, it is not possible for the ego-ship to detect or predict the positions of ships outside the FOV of the camera sensors on the ego-ship. Therefore, to increase the intuitiveness of the BEV visualisation, the FOV of the ego-ship is added to the plots. Figure 3.15 illustrates the plot with the added FOV, where the white segment of the plot is the ego-ship's FOV and the red segment describes the environment that the ego-ship cannot perceive.

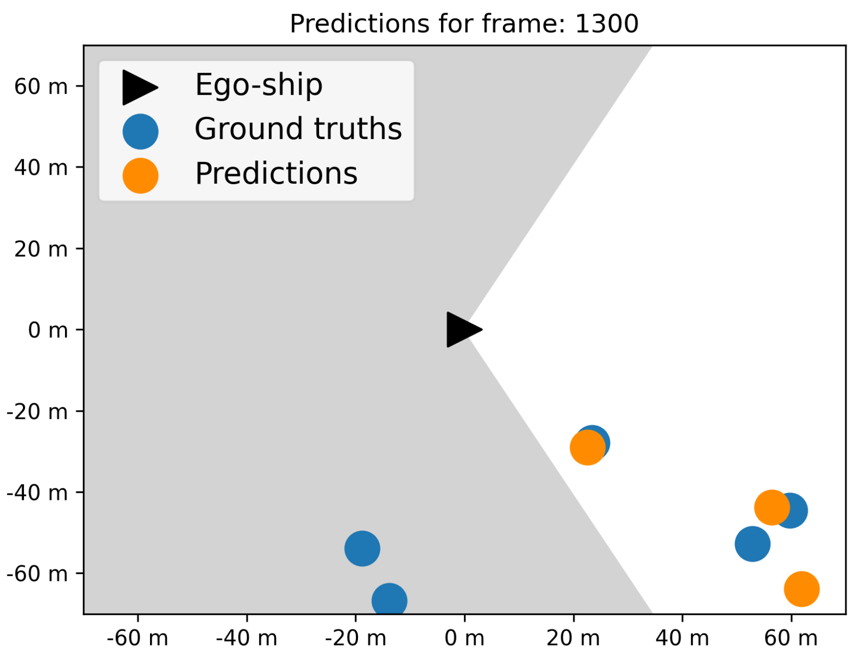


Figure 3.15: BEV plot for timeframe 1300 with visualized FOV.

3.11.3 Comparing the BEV Visualization with an Image Frame from the SDG

After creating the BEV visualisation, it is possible to showcase the ASV SA system from a holistic point of view. Figure 3.16 displays image frame 1300 from the right camera generated by the SDG. In the image from the SDG 3.16, three boats can be seen on the right side of the image. In the corresponding BEV visualisation 3.17 for the corresponding timeframe, there are three ground truth values with coordinates similar to the SDG image. Furthermore, three predictions are made with distances and bearings close to the ground truth values. Thus, when analysing the results, the BEV visualisation can be used to analyse several aspects of the system qualitatively.

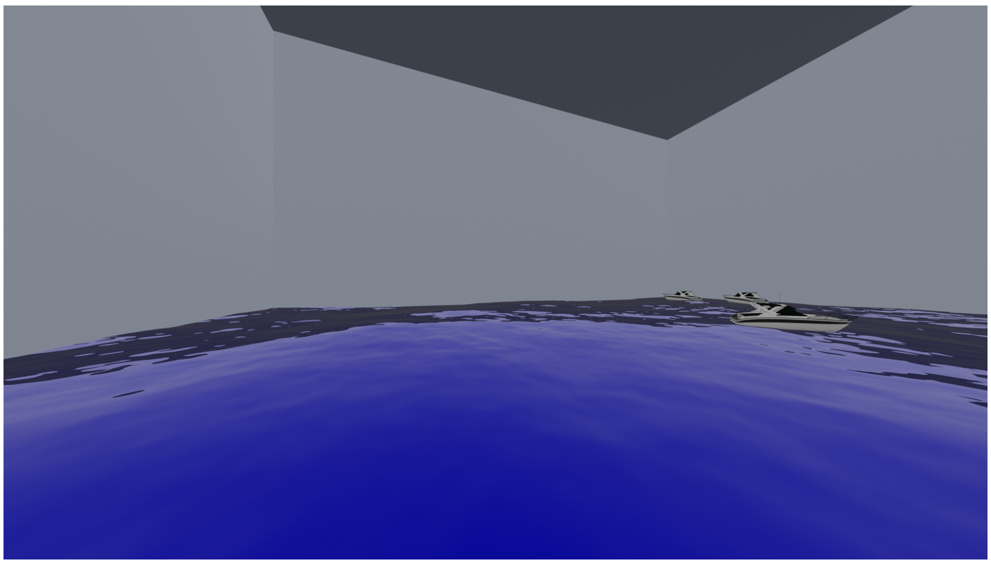


Figure 3.16: Image frame 1300 from the right camera in the SDG.

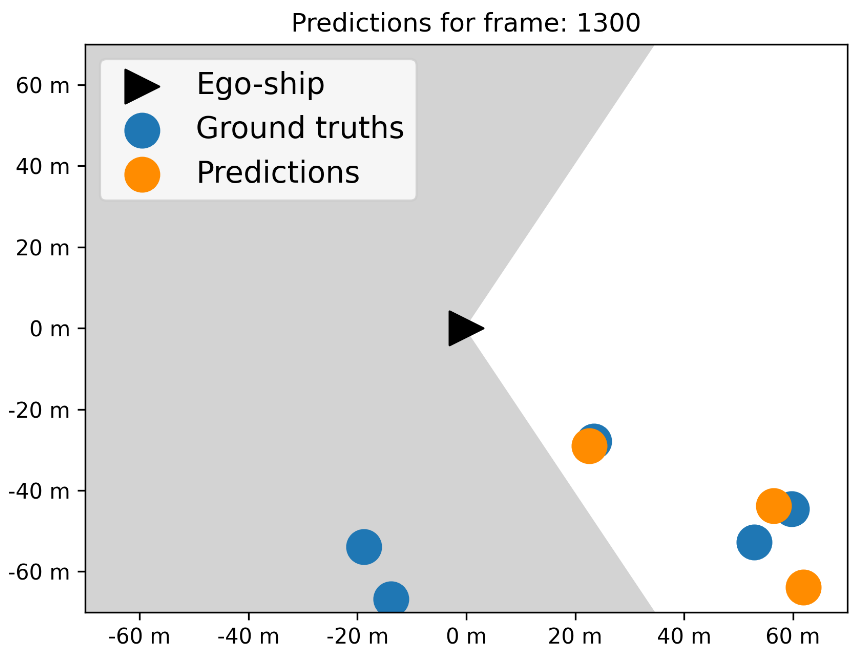


Figure 3.17: BEV plot for timeframe 1300.

3.12 Fusing Tracks with Coordinate Predictions

In order to gain information about the paths of different objects, tracking was performed on the synthetic data. In order to use the results from the ByteTrack algorithm, the coordinate predictions have to be fused with the tracking data. This section will discuss different methods for fusing the coordinate predictions with the tracking data and the results of the methods.

3.12.1 Initial Fusion with Tracks from a Single Camera

Due to the stereo camera setup in the SDG, tracks are generated for the videos taken from the left and right camera of the ego-ship. The tracks generated are unique for each camera, and there is no implementation in ByteTrack for fusing tracks in stereo images. Therefore, the tracks generated from the left camera were chosen as an initial fusion method. The tracks from the left camera are fused with the corresponding coordinate prediction for each timeframe where there exist tracks. Now the ASV SA system has information about the estimated positions of detected objects fused with their tracks. After fusing the tracks with the coordinate predictions, the bird's eye view BEV visualisation was updated to accurately reflect the tracked coordinate predictions. As a result of the aforementioned update, the BEV visualisation presents a more comprehensive and accurate depiction of the predicted coordinates, which can be seen in Figure 3.18. Furthermore, the ground truth was also annotated with tracks to make it easier to differentiate the ground truths over multiple frames.

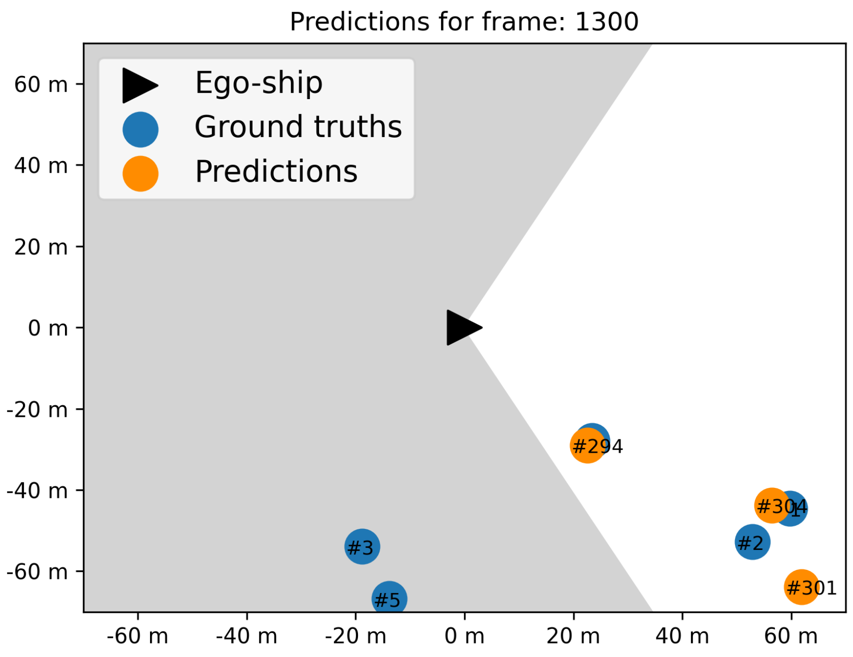


Figure 3.18: BEV plot for timeframe 1300 with tracks.

The following subsections describe and compare several methods for fusing the tracks from the SA system. Therefore an implementation with tracks originating only from the right camera was also made to properly evaluate the performance of tracks originating from a single camera in a stereo system.

3.12.2 Combining the Tracks from Both Cameras

When combining the tracks from the left and the right cameras, the performance can be improved in different ways. The level of confidence in a track increases if the tracking from both cameras indicates that the track is alive. However, if only a track from one of the cameras has to be alive, the results would give longer-lasting tracks. With these two approaches, there is a trade-off between confidence in a track and having longer-lasting tracks.

Method 1: Using Both Cameras to Increase Confidence in Tracks

If both cameras associate a detection with the same track, it is more likely to be the correct association. In order to combine the tracks from the left and the right camera, the track ids were concatenated. This means that if a detection has tracking id 6 in the left camera and 8 in the right, the final track id would be 68. Each step of the method is described below:

1. Check the id associated with the detection in the left (ID_l) and right (ID_r) cameras. If there is a track with $ID = ID_l + ID_r$, the detection is appended to the track.
2. If there is no track with the desired ID, a new track is initiated.

Method 2: Using Both Cameras to Increase Track Length

This method tries to keep tracks alive for as long as possible. If a track is initiated for an object in one of the cameras and then the track is removed, then the tracks for the other camera are inspected to see if any correspond to the lost track. For example, If no track from the left camera is associated with the detection, the method looks at the track from the right camera to see if it is associated with the detection. This will make tracks stay alive for a longer period of time but will also decrease the confidence of the tracks. The steps of the method can be described as follows:

1. Check if detection is still alive by looking at the tracking ID from the left camera.
2. Check if the ID associated with the track is still alive in the other camera.
3. If none of the steps above is right for the detection, it should be associated with a new track.

Method 3: Using Tracks from Both Cameras and Locations of Previous Tracks

Method 3 implements the same fusion as Method 2. However, instead of fusing every track that is alive in either camera, Method 3 utilises a more selective approach. A track should only be kept alive if the coordinate of a new detection associated with a track is feasible, given the location of the previous detection. By combining tracks from both cameras and selectively choosing which tracks to keep alive. Longer-lasting tracks can be generated while maintaining track confidence. The steps of Method 3 are described below:

1. Check if the detections are still alive by looking at the tracking ID from the left camera.
2. Check if the ID associated with the track is still alive in the other camera. If the detection is within a threshold distance of the detection from the previous frame, the track is kept alive.

Method 4: Generating Tracks Solely Using Coordinate Predictions

To analyse the performance of the different methods utilising the ByteTrack tracks, a tracking method only utilising the coordinate predictions without the tracks from Bytetrack was implemented. How far a boat can move in just a single frame is limited. With accurate, reliable coordinate estimates, it should be possible to have a well-functioning tracking algorithm based on the previous coordinates alone. In this method, detections are associated with the closest track if the distance is below a threshold. If two detections have the same track as the closest, only the closest of the two is assigned the track. When deciding what threshold to use, there is also a trade-off between having tracks that follow the target path accurately or having longer tracks with the risk of having parts of the track following different targets.

1. Check if any tracks with a detection from the previous frame are within the threshold distance and assign it to the closest.
2. Check if more than one detection is assigned to the same track and only keep the detection closest to the track.
3. If no track has a detection from the previous frame within the threshold distance, we create a new track.

3.12.3 Evaluation of the Different Methods

When evaluating the methods, each track prediction was initially associated with the closest ground truth path for a given time sequence. However, evaluating a track based on the average distance to a ground truth path poorly reflected how well the tracks were associated with the ground truth. As stated in section 3.10, it is known that boats partially outside the ego-ship's FOV could lead to significant prediction errors. As a result of those errors, the average distance from the track to the ground truth path might be high, even though the track might be accurate most of the time. Therefore, the number of coordinates in a track within a threshold distance to the ground truth path was evaluated. The results of the methods with this evaluation metric can be seen in Table 3.10.

	Threshold				
	2m	**4m**	**6m**	**8m**	**10m**
Left	28%	52.8%	63%	67.5%	70.5%
Right	28%	52.9%	63%	67.5%	70.4%
Method 1	28.5%	53.9%	64.3%	68.9%	71.9%
Method 2	27%	51%	60.9%	65.4%	68.3%
Method 3	27.8%	52.5%	62.7%	67.2%	70.2%
Method 4	26.6%	50%	60.8%	65.6%	69%

Table 3.10: An overview of how many detections in a track were within a threshold of the closest target path for all tracks.

In order to get a better understanding of how well the tracks were associated with different ground truths, the bearing angle between the different ground truths and the coordinate predictions of the tracks were evaluated. The deviations in the bearing estimates are not as significant as those in the coordinate prediction and depth estimation. Therefore, the RMSE could be used to get an understanding of how well the methods associated the tracks with the correct target. The results indicate that all methods except Method 2 do well in terms of RMSE. The RMSE when each detection is associated with the closest target is 0.072 as described in Table 3.5. The similar results between the methods presented in Table 3.11 and the result in Table 3.5 indicate that most coordinates in a track correspond to the same object path. However, the results for Method 2 indicate that there could be tracks that switch targets and subsequently perform worse than the other methods. Most of the methods have better RMSE values than associating each detection with the closest target because not all detections are associated with a track. This is due to the time it takes Bytetrack to establish tracks. When the target first enters the FOV it might be excluded from the track and lead to a lower RMSE value as the predictions closest to the edge of the FOV seem to be the least accurate.

Method	RMSE
Initial (left camera)	0.069
Initial (right camera)	0.066
Method 1	0.065
Method 2	0.113
Method 3	0.069
Method 4	0.074

Table 3.11: The RMSE for each method.

The tracks should be as accurate as possible and give as much information about a target path as possible. Furthermore, longer valid tracks give more valuable information about the targets. The results in Table 3.12 present the average track length and the number of tracks for each method.

Method	Average track length	Number of tracks
Initial (left camera)	61 frames	87
Initial (right camera)	62.3 frames	85
Method 1	38.6 frames	135
Method 2	69.4 frames	76
Method 3	61.4 frames	87
Method 4	6.26 frames	867

Table 3.12: An overview of the average track length and the total number of tracks for each method.

3.13 A Summary of the Final ASV SA System

In this chapter, several methods for different aspects of the ASV SA system have been presented and compared. This section summarises the data and methods utilized in the final ASV SA system.

3.13.1 Parameters Used in the SDG

The data used for the ASV SA system is generated by the SDG. Section 3.2 presents how the pose sequences were converted into the final synthetic data. Furthermore, important parameters used in the SDG are as follows. The stereo camera has a baseline of 20cm with a focal length of 25mm for each camera, and the sensor widths are 100mm. The stereo camera was placed "on" the vehicle with the identification "5" in the pose sequences. Lastly, the SDG generated synthetic data for 6000 timeframes in the interval [0, 5999] of the pose sequences, corresponding to a 10-minute long stereo video.

3.13.2 Overview of the Methods Implemented for the ASV SA System

- For the first module, the YOLOv8 model trained on the maritime synthetic dataset was used to detect objects in the stereo sequence.
- Then, the detections are associated with tracks utilizing the ByteTrack algorithm.
- The detections in the left camera are associated with the detections in the right camera using IoU.
- Bearing estimation is done on the associated detections with Method 4 described in section 3.8.
- Depth estimation is done using Method 1 described in section 3.9.
- The coordinates are calculated as described in section 3.10.
- The coordinates are fused with the tracks using the left camera as described in the initial method in section 3.12.

Chapter 4

Results

This chapter will present and discuss the results of the ASV SA system. The BEV visualization for the system output on all zones for 6000 frames of pose sequences can be viewed at this link: https://www.youtube.com/watch?v=49MAQCNLFuc. And the BEV visualization for the system output on Zone 1 and 2 can be viewed at this link: https://www.youtube.com/watch?v=958bvalmUYk.

4.1 Analysing the Paths of Tracks

This section presents the results from the ASV SA system. Evaluation metrics of the different modules of the system have been presented in chapter 3. To complement the results presented in chapter 3, this section will offer a qualitative analysis of the paths of tracks compared with the ground truths. The coordinate predictions for detections in zone 3 were removed to prevent outliers from generating visualisations that are hard to interpret. However, the effects of detections in zone 3 on the system are discussed in section 4.2. The ASV SA system can detect an object and accurately predict the location of the detection for its entire path for detections in zone 1 and 2. The accuracy of the SA system is satisfactory, considering the simplicity of the methods used in the architecture modules. Adding a motion model to the tracking could improve the coordinate predictions. However, adding a motion model was not prioritised as the pose sequences were made with the dynamics of a car. Therefore, adding it was deemed unnecessary for a thesis in the maritime domain. Figures 4.1, 4.2, 4.3, and 4.4 presents tracks compared with the corresponding ground truth. The figures only show single tracks and their corresponding ground truth. This was done since visualising the paths of multiple tracks and ground truths generated cluttered visualisations. However, the video linked at the beginning of this chapter showcases all tracks and ground truths for all data generated from the SDG.

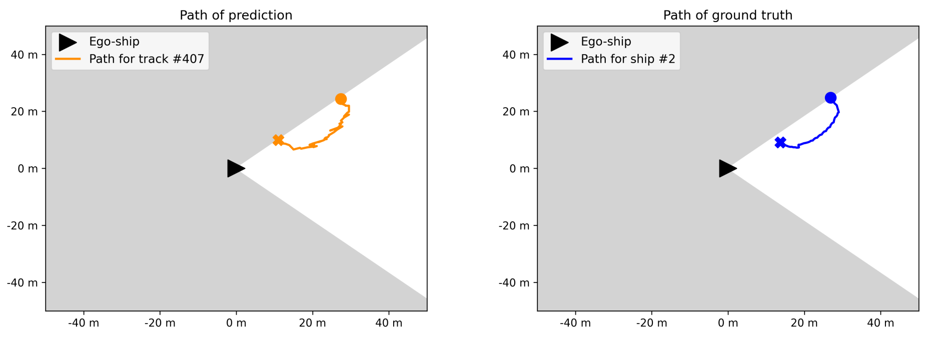


Figure 4.1: The path of track #407 (left image) and the path of ground truth ship #2 (right image).

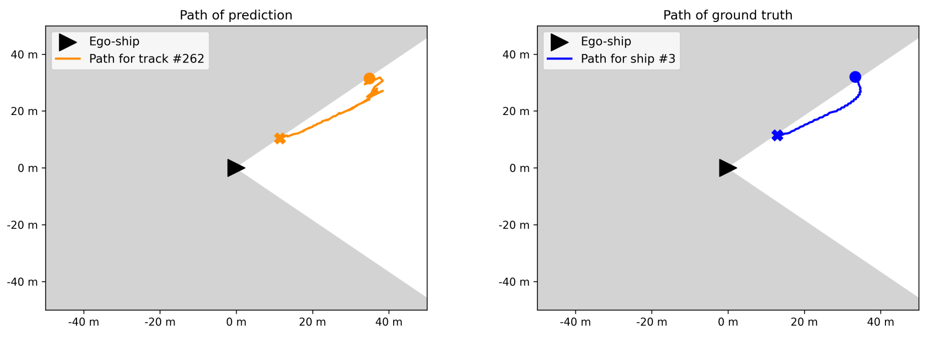


Figure 4.2: The path of track #262 (left image) and the path of ground truth ship #3 (right image).

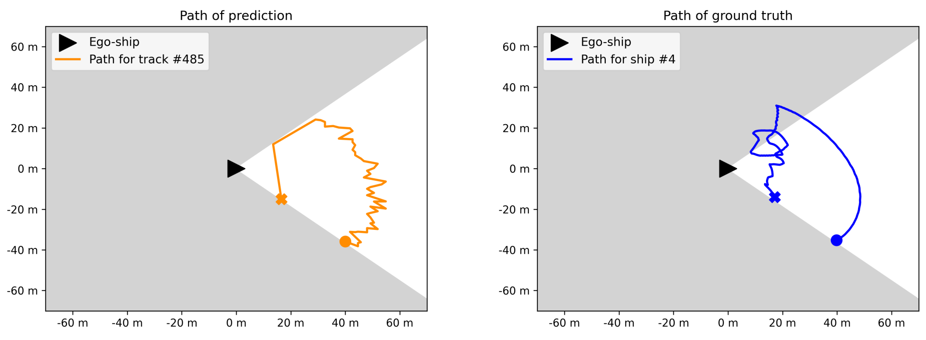


Figure 4.3: The path of track #485 (left image) and the path of ground truth ship #4 (right image).

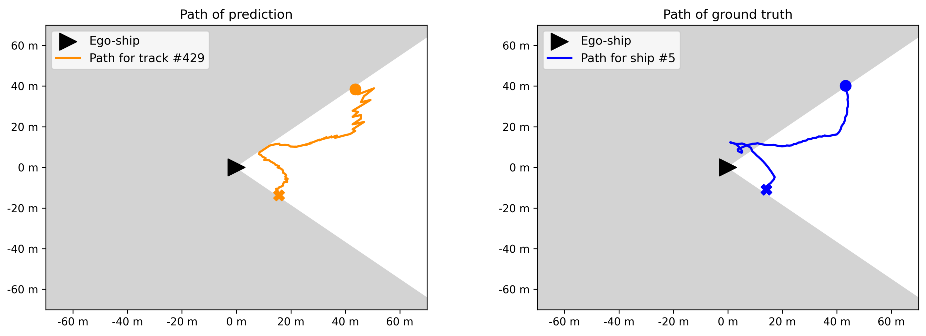


Figure 4.4: The path of track #429 (left image) and the path of ground truth ship #5 (right image).

4.2 Complex Situations for the ASV SA System

This section will analyse different types of complex situations regarding the SA of the ASV system. The analysis will cover situations such as occlusions and partially visible ships. In this analysis, the predictions from all zones are analysed.

4.2.1 The Dangers of Occluded Ships

Effects of Occlusion in the ASV SA System

When analysing the results, it is clear that when a ship is occluded by another ship, the system either can't detect the object or poorly predicts its distance and bearing angle. This can be seen in Figure 4.5 where ship *#4* is not detected as it is behind ship *#5* in the ego-ship's FOV. To interpret the performance of the ASV SA system, the input data can be viewed in Figure 4.6, where a ship is occluded by another ship. For the safety of an ASV, detecting all objects close to the ego-ship is crucial. Furthermore, the ASV SA system built in this thesis only predicts the distance and bearing of objects detected by the YOLOv8 model. Therefore, when an object is undetected, the ASV has no spatial information about the object either. Regarding the safety of the ego-ship, occluded objects in the distance don't pose an immediate threat. Still, if the occluded ships are in close vicinity to the ego-ship, it could have dire consequences.

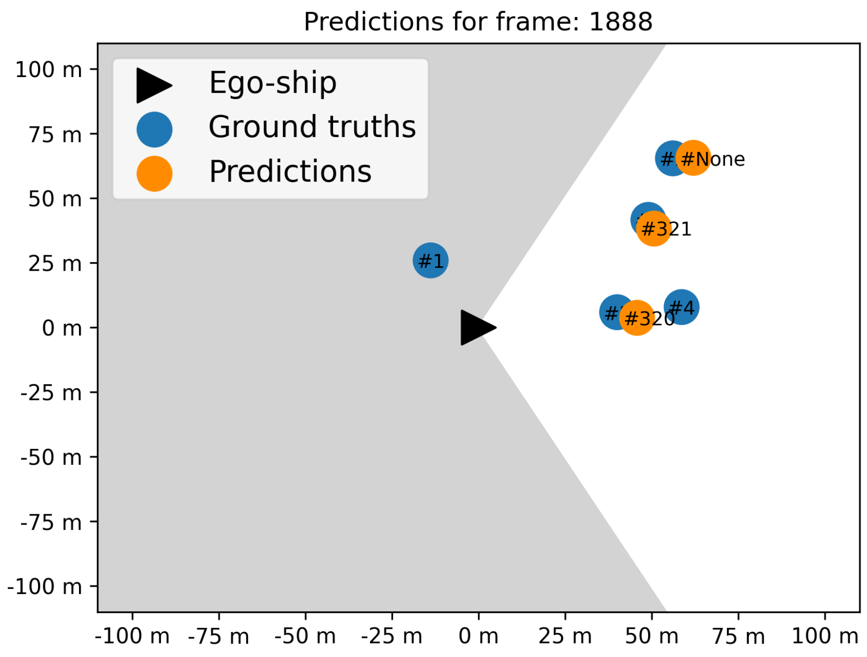


Figure 4.5: Ship #4 is occluded by ship #5 predictions viewed from BEV.

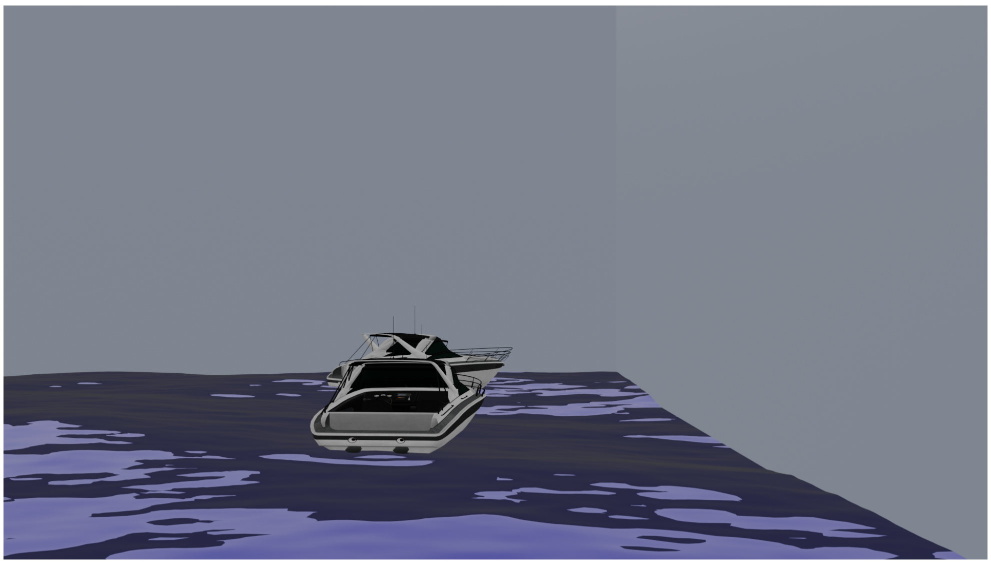


Figure 4.6: Ship #4 is occluded by ship #5 viewed in the SDG.

Occlusion Mitigation Strategies

Occlusion is a prominent problem in object detection, and there is research that suggests training object detection algorithms on labelled synthetic data of occluded objects in scenes could improve the ability to detect such objects. However, it also states that the current object detection algorithms struggle with occlusion and that there are few labelled datasets of occluded objects regardless of domain [SSV21]. Therefore, one solution to detect occluded ships could be to label a synthetic dataset of occluded vessels and train an object detection algorithm on the dataset. However, even with training on such a dataset, the model could still lack the robustness needed for a real-world application as precise detection is a necessary component of a safe ASV. Another solution could be implementing radar or LiDAR sensors and fusing the sensor input to create a more robust tracking system. The algorithm proposed by Shin et al. utilises high-level sensor fusion between LiDAR and camera sensors to create robust tracks even when the object is partially or entirely occluded [SAL17]. However, for a SA system that only utilises camera sensors improving the tracking algorithm for occluded objects could be a solution for tracking occluded objects based solely on camera sensors. One approach combines instance segmentation with the Bytetrack tracking algorithm to better predict the tracks of occluded objects. This approach yielded better results for ship tracking than other methods and specifically on ships that were occluded [Che+22].

Conclusion

To conclude, occlusion negatively impacts the accuracy of the detection and tracking algorithms used on the ASV SA system and the overall safety of the ASV. There are different approaches to improve the robustness of both object detection and tracking regarding occluded objects and should be investigated for any ASV to increase safety and SA, especially for real-world ASV applications.

4.2.2 Partially Visible Ships

Why are Partially Visible Ships a Problem?

Another limiting factor of the ASV SA system are scenarios when ships are partially visible in FOV of the ego-ship. When ships are entering or exiting the FOV, the left and right cameras view different portions of the ships as can be seen in Figure 4.7. In contrast to the occlusion problem discussed in subsection 4.2.1, where the ships in the scene are not detected or tracked, the partially visible ships are usually detected and tracked even if only a small part of the ship is visible. However, the problem is that the depth estimations are poor compared to other scenarios with tracked ships. This is due to the difference in the size and position of the AABBs that occur in the stereo camera setup when tracking partially visible ships. In Figure 4.8, a partially visible ship and a fully visible ship have been detected and tracked. The fully visible ship has been assigned track id #262 and has an accurate distance and bearing estimation. Furthermore, the partially visible ship has been assigned track id #266 and has an accurate bearing estimation but a poor distance estimation regarding the closest visible ground truth, ship #5. This example showcases a limitation with ASV SA system, namely the partially visible ships problem.

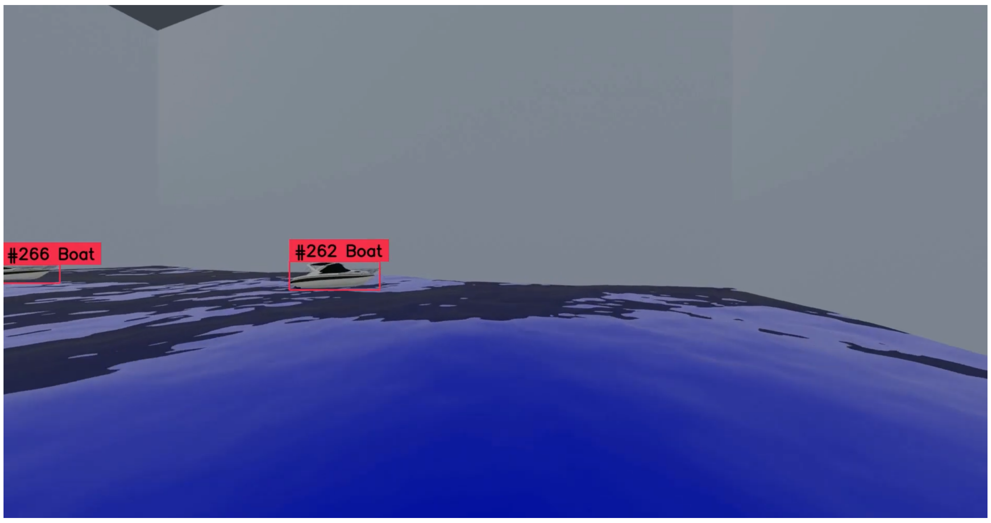


Figure 4.7: Partially visible ship (#266) viewed from left camera.

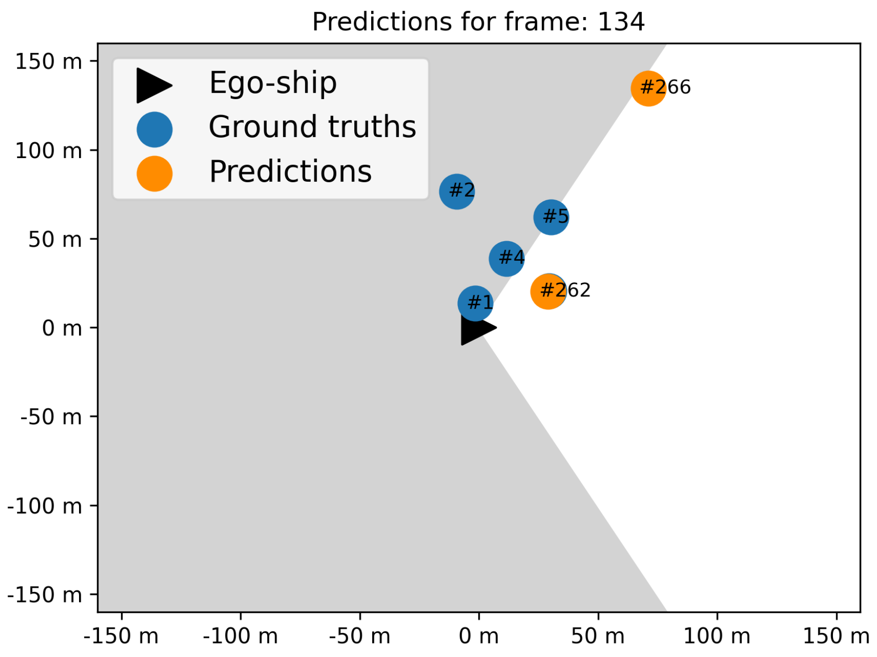


Figure 4.8: Partially visible ship (#266) viewed from BEV.

Improving Visibility by Increasing the Number of Cameras

As stated in section 3.10, the depth estimation predictions made in Zone 3, or the edges of the FOV of the stereo setup, were considerably worse than the depth estimation predictions made in Zone 1 and Zone 2. This restriction is due to the camera setup in the SDG. Thus a possible solution to the partially visible ships problem could be to increase the number of stereo cameras with overlapping FOVs on the ego-ship. By doing so, it could be possible to choose AABBs with a low bearing angle in Zone 1 and Zone 2 and avoid predictions for detections in Zone 3. Such a multi-stereo camera setup can be seen in Figure 4.10 where in contrast to the single stereo camera setup seen in Figure 4.9, there is no zone 3 due to overlapping zone 2's in the multi-stereo camera setup. By having overlapping FOVs, we hypothesise that it's possible to predict the location of objects in either zone 1 or zone 2 of the ego-ship. Excluding the need to predict the location of objects in Zone 3, increasing the accuracy of the ASV SA system as a whole. Furthermore, increasing the number of cameras on the ego-ship and creating a 360 degrees FOV could detect more ships, not only those in front of it. However, such an architecture could still have to predict detections in Zone 3 in the real world due to sensor failures, but it would likely increase the robustness of the system nonetheless.

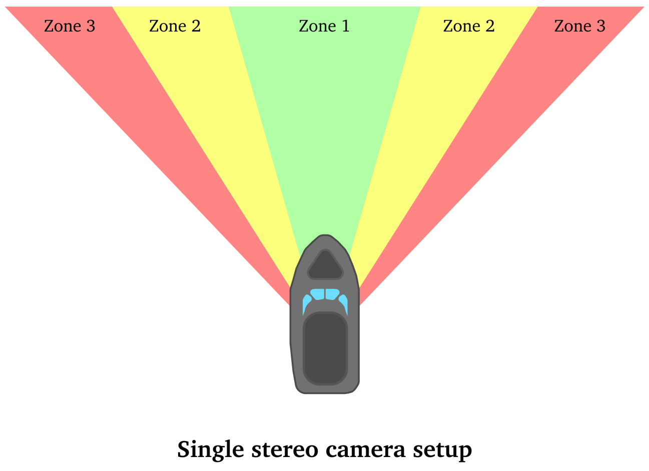


Figure 4.9: Bearing zones of a single stereo camera setup.

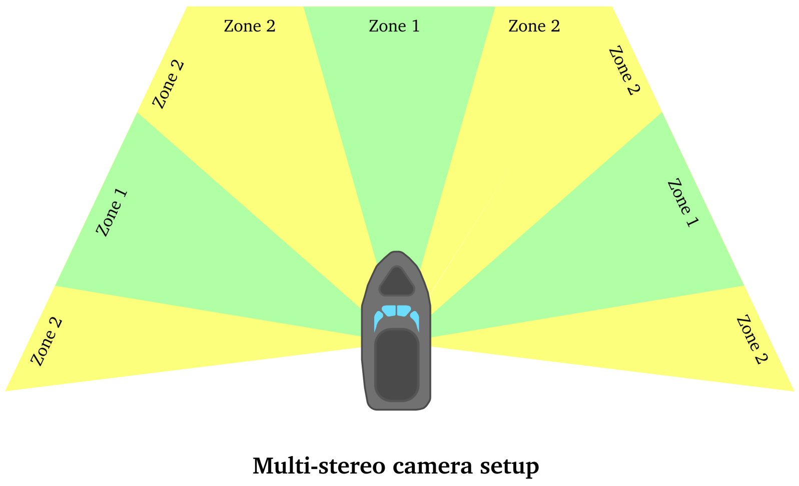


Figure 4.10: Bearing zones of a multi-stereo camera setup.

4.2.3 Conclusion

Partially visible ships are an issue for a single stereo camera setup which negatively impacts the depth estimations of tracked ships in the edges of the ego-ship's FOV. A possible solution could be to increase the number of cameras with overlapping FOV's. However, such a solution is prone to real-world complications such as sensor failure and noise. Therefore, other solutions which improve the SA of the ego-ship with a single stereo camera setup should be investigated as it would also improve the robustness of the multi-stereo camera setup if a sensor should fail.

Chapter 5

Future Work

This chapter presents several possible improvements for the ASV SA system and synthetic data for future research. The ASV SA system contains multiple modules, and each module could have been further developed by focusing solely on that module of the system. Furthermore, the synthetic data needed for the system could also have been improved in several aspects. In order to complete the project within the given period of time, simplifying modules and accepting novel solutions were imperative in order to complete the system as a whole. By spending more time on a specific module of the system and creating better and more accurate methods, the credibility and performance of the system could be improved.

5.1 Improving the Synthetic Data

There are several simplifications regarding the synthetic data, as described earlier in the thesis. The vehicle dynamics are not designed for a boat at sea. To improve the degree of realism, the dynamics of a boat could be added with waves and ocean currents. Furthermore, the pose files could be improved by adding reinforcement learning. Currently, the files are generated using reactive agents and do not move very intelligently. As stated in Section 3.2.6, we did not add any background. The object detection task would be more difficult and realistic with a background. It would be important to add a more realistic background for an end-to-end system with a focus on all the parts of the system. Additional vessel types, such as kayaks, tanker ships and passenger ships, should be added to further increase the realism of the synthetic data. Having different vessels with different dimensions, textures, and dynamics would generate more interesting and realistic environments. Furthermore, other sensor types such as radar and LiDAR could be added to the SDG to improve the versatility of the ASV SA architecture by supporting other ASV tasks such as sensor fusion.

5.2 Improving Robustness by Incorporating Real and Synthetic data

To gain more trust in the methods and implementations, the methods tuned and tested on synthetic data could be tested on data from a real experiment. This would let the system utilize the different advantages that synthetic and real data provide. For example, the system could be tested in numerous dangerous situations and realistic settings with authentic data. It is also easier to measure the performance of the methods with a reference point. If the methods were tested on a more established dataset or benchmark, it would be easier to compare to other methods.

5.3 Object Detection

Creating accurate AABBs is a crucial part of the SA system. Training on a more extensive and varied dataset would give a better detector. It was also considered outside of the scope to create a detector that could differentiate between different types of vessels, such as a motor boat and a kayak. This should be added for a real-life application as it reveals valuable information about the surrounding objects.

5.4 Tracking

By using the same boat texture for all the vehicles in the animation, we lose some of the advantages of ByteTrack. If different boat textures were used for each vehicle, more interesting results could be acquired in terms of ReID and the overall tracking performance. The results could also be improved by adding a motion model. For instance, a Kalman filter could be applied to improve the tracking.

5.5 Depth Estimation

The strengths of the method used for depth estimation are the simplicity and absence of overhead. There are methods that perform better in terms of accuracy, for instance, a pseudo-LiDAR method designed by Wang et al. [Wan+20]. This method does not calculate depth based on a single point in the object but on every pixel in the image instead. This would generate more accurate estimations while probably eliminating the problem of partially visible ships described in Section 4.2.2. By improving the depth estimation method, the accuracy of the SA system would improve significantly.

5.6 3D Object Detection

Currently, the ASV SA represents the objects in the environment as 2D points which compress vital information about the objects. The simplified representation is an adequate representation of the object in the system for the scope of this thesis. However, as stated in section **??**, objects can be represented as 3D bounding boxes with information about the detected object's dimensions, location and orientation. For a SA system, such information is valuable for an accurate representation of the environment. With this information, meta-information such as velocity, direction and future states of the environment are easier to predict. With the SDG, the necessary ground truth data for 3D object detection can be extracted for a model to train on. There exist several state-of-the-art 3D object detectors utilizing image data. However, as stated above, the SDG can implement a LiDAR sensor which allows for LiDAR and sensor fusion 3D object detection models.

Chapter 6

Conclusion

This paper proposes a novel ASV SA architecture which represents the surrounding environment of an ASV as a dynamic representation solely utilizing camera sensors. To accomplish the implementation of the architecture, semi-realistic synthetic data was created for ASV tasks which can be further developed to shorten the gap of publicly available high-quality synthetic data between autonomous vehicle domains. The implemented ASV SA system can accurately represent its environment in terms of tracked detections and their locations and is a building block towards achieving SA in an ASV. For all tracks, the system can predict an object's location during the entire track with an accuracy of 63% with a margin of error of 6 meters. However, the location predictions are less accurate when vessels are occluded or only partially visible. Finally, comparing the ASV SA system with other existing systems is difficult since it has not been tested on real data or evaluated on a benchmark dataset.

Bibliography

Bai, X., Li, B., Xu, X., & Xiao, Y. (2022). A Review of Current Research and Advances in Unmanned Surface Vehicles. *Journal of Marine Science and Application*, *21*(2), 47–58. https://doi.org/10.1007/s11804-022-00276-9

designboom matthew burgos, m. b. |. (2023). Electric self-driving ferry in stockholm ready for public transport use in the summer of 2023. Retrieved April 27, 2023, from https://www.designboom.com/technology/electric-self-driving-ferry-stockholm-zeam-zeabuz-torghatten-04-25-2023/

Agency, E. M. S. (2022). ANNUAL OVERVIEW OF MARINE CASUALTIES AND INCIDENTS 2022. https://safety4sea.com/wp-content/uploads/2022/11/EMSA-Annual-Overview-of-Marine-Casualties-and-Incidents-2022-2022_11.pdf

Wardziński, A. (2006). The Role of Situation Awareness in Assuring Safety of Autonomous Vehicles. In J. Górski (Ed.), *Computer Safety, Reliability, and Security* (pp. 205–218). Springer. https://doi.org/10.1007/11875567_16

He, G., Wang, W., Shi, B., Liu, S., Xiang, H., & Wang, X. (2022). An Improved YOLO v4 Algorithm-based Object Detection Method for Maritime Vessels. *International Journal of Science and Engineering Applications*, *11*(04), 50–55. https://doi.org/10.7753/IJSEA1104.1001

Endsley, M. R. (1995). Toward a theory of situation awareness in dynamic systems [Publisher: SAGE Publications Sage CA: Los Angeles, CA]. *Human factors*, *37*(1), 32–64.

Helgesen, Ø. K., Vasstein, K., Brekke, E. F., & Stahl, A. (2022). Heterogeneous multi-sensor tracking for an autonomous surface vehicle in a littoral environment. *Ocean Engineering*, *252*, 111168. https://doi.org/10.1016/j.oceaneng.2022.111168

Rasshofer, R. H., & Gresser, K. (2005). Automotive Radar and Lidar Systems for Next Generation Driver Assistance Functions [Conference Name: Kleinheubacher Berichte 2004 - Publisher: Copernicus GmbH]. *Advances in Radio Science*, *3*(B.4), 205–209. https://doi.org/10.5194/ars-3-205-2005

Campbell, S., O'Mahony, N., Krpalcova, L., Riordan, D., Walsh, J., Murphy, A., & Ryan, C. (2018). Sensor Technology in Autonomous Vehicles : A review. *2018 29th Irish Signals and Systems Conference (ISSC)*, 1–4. https://doi.org/10.1109/ISSC.2018.8585340

Ahrnbom, M. (2022). Computer Vision for Automated Traffic Safety Assessment: A Machine Learning Approach.

Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2021). A survey of convolutional neural networks: Analysis, applications, and prospects [Publisher: IEEE]. *IEEE transactions on neural networks and learning systems*.

Terven, J., & Cordova-Esparza, D. (2023). A Comprehensive Review of YOLO: From YOLOv1 and Beyond [arXiv:2304.00501 [cs]]. Retrieved May 24, 2023, from http://arxiv.org/abs/2304.00501

Tripathi, A., Gupta, M. K., Srivastava, C., Dixit, P., & Pandey, S. K. (2022). Object Detection using YOLO: A Survey. *2022 5th International Conference on Contemporary Computing and Informatics (IC3I)*, 747–752. https://doi.org/10.1109/IC3I56241.2022.10073281

Ghasemieh, A., & Kashef, R. (2022). 3D object detection for autonomous driving: Methods, models, sensors, data, and challenges [Publisher: Elsevier]. *Transportation Engineering*, *8*, 100115.

Welch, G., Bishop, G., et al. (1995). An introduction to the Kalman filter [Publisher: Chapel Hill, NC, USA].

Zhang, Y., Sun, P., Jiang, Y., Yu, D., Weng, F., Yuan, Z., Luo, P., Liu, W., & Wang, X. (2022). Bytetrack: Multi-object tracking by associating every detection box. *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXII*, 1–21.

Ming, Y., Meng, X., Fan, C., & Yu, H. (2021). Deep learning for monocular depth estimation: A review. *Neurocomputing*, *438*, 14–33. https://doi.org/10.1016/j.neucom.2020.12.089

Jain, R., Kasturi, R., & Schunck, B. G. (1995). *Machine vision*. McGraw-Hill.

Smolyanskiy, N., Kamenev, A., & Birchfield, S. (2018). On the Importance of Stereo for Accurate Depth Estimation: An Efficient Semi-Supervised Deep Neural Network Approach. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 1120–11208. https://doi.org/10.1109/CVPRW.2018.00147

Geiger, A., Lenz, P., Stiller, C., & Urtasun, R. (2013). Vision meets robotics: The KITTI dataset [Publisher: SAGE Publications Ltd STM]. *The International Journal of Robotics Research*, *32*(11), 1231–1237. https://doi.org/10.1177/0278364913491297

Nikolenko, S. I. (2021). *Synthetic Data for Deep Learning* (Vol. 174). Springer.

Dilmegani, C. (2022). Synthetic Data vs Real Data: Benefits, Challenges in 2023. Retrieved April 24, 2023, from https://research.aimultiple.com/synthetic-data-vs-real-data/

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., & Koltun, V. (2017). CARLA: An Open Urban Driving Simulator [ISSN: 2640-3498]. *Proceedings of the 1st Annual Conference on Robot Learning*, 1–16. Retrieved April 24, 2023, from https://proceedings.mlr.press/v78/dosovitskiy17a.html

Talwar, D., Guruswamy, S., Ravipati, N., & Eirinaki, M. (2020). Evaluating Validity of Synthetic Data in Perception Tasks for Autonomous Vehicles. *2020 IEEE*

International Conference On Artificial Intelligence Testing (AITest), 73–80. https://doi.org/10.1109/AITEST49225.2020.00018

Seib, V., Lange, B., & Wirtz, S. (2020). Mixing Real and Synthetic Data to Enhance Neural Network Training – A Review of Current Approaches [arXiv:2007.08781 [cs]]. Retrieved April 25, 2023, from http://arxiv.org/abs/2007.08781

Foundation, B. (n.d.). About. Retrieved April 27, 2023, from https://www.blender.org/about/

Roboflow. (2023). About. Retrieved May 29, 2023, from https://docs.roboflow.com/

recognition, B. (2023). Boat detection Dataset [Publication Title: Roboflow Universe Type: Open Source Dataset Published: https://universe.roboflow.com/boat-recognition/boat-detection-1vhc2]. https://universe.roboflow.com/boat-recognition/boat-detection-1vhc2

Saleh, K., Szénási, S., & Vámossy, Z. (2021). Occlusion Handling in Generic Object Detection: A Review [arXiv:2101.08845 [cs]]. *2021 IEEE 19th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, 000477–000484. https://doi.org/10.1109/SAMI50585.2021.9378657

Shin, S.-G., Ahn, D.-R., & Lee, H.-K. (2017). Occlusion handling and track management method of high-level sensor fusion for robust pedestrian tracking. *2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 233–238. https://doi.org/10.1109/MFI.2017.8170434

Chen, X., Chen, W., Yang, Y., Li, C., Han, B., & Yao, H. (2022). High-fidelity ship imaging trajectory extraction via an instance segmentation model. *2022 International Symposium on Sensing and Instrumentation in 5G and IoT Era (ISSI)*, 165–169. https://doi.org/10.1109/ISSI55442.2022.9963190

Wang, Y., Chao, W.-L., Garg, D., Hariharan, B., Campbell, M., & Weinberger, K. Q. (2020). Pseudo-LiDAR from Visual Depth Estimation: Bridging the Gap in 3D Object Detection for Autonomous Driving [arXiv:1812.07179 [cs]]. Retrieved April 13, 2023, from http://arxiv.org/abs/1812.07179