Master's thesis

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

Andrea Stette Jessen
Solveig Jørgensen Mohr

Object Detector Simulation for Certification of Autonomous Surface Vessels

Master's thesis in Computer Science
Supervisor: Rudolf Mester
Co-supervisor: Grunde Løvoll
June 2023

NTNU
Norwegian University of
Science and Technology

Andrea Stette Jessen
Solveig Jørgensen Mohr

Object Detector Simulation for Certification of Autonomous Surface Vessels

Master's thesis in Computer Science
Supervisor: Rudolf Mester
Co-supervisor: Grunde Løvoll
June 2023

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

NTNU
Norwegian University of
Science and Technology

Abstract

The potential of Autonomous Surface Vessels (ASV) has captured the attention of the maritime industry. For ASVs to operate safely, it is crucial to certify them. Certification involves testing each component of the autonomous system individually and as a complete system. One such component is the external Situational Awareness (SITAW) system, particularly the tracker within this system. A tracker monitors the environment by connecting detections of objects over time into tracks. Detections are obtained through sensors and object detection algorithms.

Conducting full-scale testing of trackers presents significant challenges due to the need for extensive test data. Obtaining real-life test data is costly and labour-intensive, and covering all significant operational scenarios would be difficult, if not impossible. An alternative approach for obtaining test data is through simulations. Simulations can replicate the system with high accuracy or prioritise computational efficiency. This thesis proposes a novel approach for efficiently simulating camera-based object detector output in maritime environments to enable tracker testing. The designed and developed system, Marine Object Detector Simulator (MODSIM), replicates the output behaviour of the detector and incorporates detector errors through statistics and statistical distributions. This approach eliminates the need for complex simulations of the underlying error causes. The viability of the system is assessed by performing three different simulations using statistical parameters. These parameters are based on a real camera-based object detector trained on a synthetic environment under various detection conditions.

Sammendrag

Interessen for autonome fartøy i den maritime industrien har økt betydelig. For å sikre sikker drift av disse fartøyene, er sertifisering avgjørende. Dette innebærer omfattende testing av hver enkelt komponent i det autonome systemet, og systemet som helhet. En viktig komponent i et autonomt system er det eksterne situasjonsforståelse systemet, og spesielt målfølgingssystemet innenfor dette systemet. Målfølgingssystemet overvåker omgivelsene ved å koble sammen deteksjoner av objekter over tid og danne spor. Deteksjoner gjøres ved hjelp av sensorer og deteksjonsalgoritmer.

Fullskalatesting av målfølgingssystemene er en betydelig utfordring, grunnet behovet for omfattende mengder testdata. Det er dyrt og resurskrevende å skaffe ekte testdata, og det vil være vanskelig, om ikke umulig, å dekke alle viktige testsituasjoner. En alternativ tilnærming er å skaffe testdata gjennom simuleringer i syntetiske omgivelser. Simuleringer kan prioritere å gjenskape systemet på en nøyaktig måte eller prioritere effektivitet.

Denne oppgaven introduserer en ny og effektiv måte å simulere kamerabaserte objektdeteksjonssystemer i maritime omgivelser for å tilrettelegge for testing av målfølgingssystemer. Det utviklede systemet, Marine Object Detector Simulator (MODSIM), etterligner oppførselen til deteksjonssystemet og introduserer feil i deteksjoner ved hjelp av statistikk og statistiske fordelinger. Denne metoden eliminerer behovet for komplekse simuleringer av underliggende feilkilder. Systemet demonstreres ved å gjennomføre tre ulike simuleringer basert på statistiske parametere. Parameterne er basert på et ekte kamerabasert objektdeteksjonssystem som er trent på syntetisk data under forskjellige vær- og lysforhold.

Preface

This document presents a master's thesis in Computer Science at the Norwegian University of Science and Technology (NTNU), conducted in collaboration with Det Norske Veritas (DNV). The primary research for this thesis was performed during the spring semester of 2023, building upon a specialisation project carried out in the fall semester of 2022.

First and foremost, our sincere gratitude goes to the individuals at DNV for providing the problem description and granting access to data. We would like to extend special thanks to our main contacts at DNV, Grunde Løvoll and Kristian Bertheussen Karolius, for their valuable guidance throughout the project.

We would like to express our deep appreciation to our supervisor, Rudolf Mester, for dedicating his time and effort to this thesis. His profound insights into the underlying concepts and valuable input on problem-solving have been instrumental in improving the quality of this work.

Additionally, we extend our gratitude to our fellow student, Henrik Fjellheim, for providing pose sequences from a synthetic environment, enabling us to conduct more complex simulations.

Finally, we would like to thank our family and friends for their continuous support. Thanks to our classmates at Gamle Fysikk for making the long study days much more enjoyable. A special acknowledgement goes to our classmate, Erlend Øien, for being an excellent sparring partner whenever challenges arose.

Andrea Stette Jessen & Solveig Jørgensen Mohr

Trondheim, June 9, 2023

Contents

Abstract i

Sammendrag iii

Preface v

List of Figures xi

List of Tables xiii

Abbreviations xv

1. Introduction 1
- 1.1. Motivation and Problem Description . . . 1
- 1.2. Objective and Scope . . . 3
- 1.3. System Requirements . . . 3
- 1.4. Related Work . . . 4
- 1.5. Contributions . . . 5
- 1.6. Thesis Outline . . . 5

2. Marine Situational Awareness Fundamentals 7
- 2.1. Sensors in External Marine Situational Awareness Systems . . . 7
- 2.2. Object Detection . . . 8
  - 2.2.1. Visual Object Detection . . . 9
- 2.3. Tracking of Marine Objects . . . 10
  - 2.3.1. Evaluating the Performance of Tracker Systems . . . 11

3. Coordinate Systems 13
- 3.1. Definition of Coordinate Systems . . . 13
  - 3.1.1. World Coordinate Frame (WCF) . . . 13
  - 3.1.2. Vehicle Coordinate Frame (VCF) . . . 13
  - 3.1.3. Nominal Vehicle Coordinate Frame (nVCF) . . . 14
  - 3.1.4. Camera Coordinate Frame (CCF) . . . 14
- 3.2. Transformation Between Coordinate Systems . . . 14
  - 3.2.1. Notation . . . 14
  - 3.2.2. Rotation and Translation Between Coordinate Systems . . . 15
  - 3.2.3. Rotation Using Euler Angles . . . 15

4. Camera Fundamentals 17
- 4.1. Pinhole Camera . . . 17
- 4.2. A More Comprehensive Camera Model . . . 19
  - 4.2.1. The Intrinsic Camera Parameters . . . 19
  - 4.2.2. The Extrinsic Camera Parameters . . . 21
  - 4.2.3. Projection of Real-World Coordinates to the Image Plane . . . 21

5. Characterising a Visual Object Detector 23
5.1. Characterising a Binary Decision-Making System . . . 23
5.2. Empirical Characterisation of Binary Decision-Making Systems . . . 23
5.2.1. Metrics for Performance Measuring of Binary Decision-Making Systems . 24
5.2.2. Imbalanced Dataset . . . 26
5.3. Characterisation of a Binary Decision-Making System with Probabilistic Concepts 26
5.4. Characterisation of Decision Systems with Multiple Labels . . . 28
5.5. Characterisation of Probabilistic Classifications . . . 29
5.5.1. Cross-Entropy Loss . . . 29
5.6. Characterisation of Systems that Produce Axis-Aligned Bounding Boxes . . . 30
5.6.1. Identifying "Correct" and "Incorrect" Detections . . . 30
5.6.2. Confusion Matrix of Systems with Multiple Labels that Produce Real-Valued Vector Outputs . . . 31
5.6.3. Characterising the Positional and Dimensional Uncertainty in Bounding Box Detection . . . 32
5.7. Tuning a Detector . . . 36

6. Characterising Object Detector Performance under Varying Conditions 39
6.1. Temporal Model of Detector Performance . . . 39
6.2. Calculating the Transition Matrix from Observation . . . 42
6.3. Transition Matrix for Object Detection in Dynamic Environments . . . 42
6.4. Influence of Weather and Light Conditions on Visual Detector Performance . . . 44

7. A Proposed System for Object Detection Simulation for Autonomous Surface Vessel Certification 47
7.1. Overview of the Marine Object Detector Simulator (MODSIM) . . . 47
7.2. Dynamic Scene Generator . . . 48
7.2.1. Information Represented in the Dynamic Scene Generator . . . 48
7.2.2. Coordinate Transformation from nVCF to WCF . . . 50
7.2.3. Generate Pose Sequences Using a Circular Motion Model . . . 51
7.2.4. Read Pose Sequences from a Synthetic Environment Created by Henrik Fjellheim . . . 53
7.3. Camera Setup in MODSIM . . . 54
7.3.1. Describing the Camera Pose in the MODSIM System . . . 54
7.3.2. Extrinsic Camera Parameters in the MODSIM System . . . 56
7.3.3. Intrinsic Camera Parameters in the MODSIM System . . . 59
7.3.4. Incorporating a Camera Setup in MODSIM . . . 59
7.4. Wave-Induced Motion . . . 61
7.4.1. Ship Dynamics . . . 62
7.4.2. Effect of Wave Motion on Image Mapping for a Camera Setup Mounted on a Vessel . . . 63
7.4.3. Simulating Wave Motion on a Vessel . . . 64
7.4.4. Adjusting Camera Pose for Wave Motion . . . 65
7.4.5. A Simplified Approach to Modelling the Impact of Waves for Simulation Purposes . . . 66
7.5. Annotation Generator . . . 66
7.5.1. Annotation Modes . . . 67
7.6. Error Generator . . . 68
7.6.1. Types of Errors . . . 68
7.6.2. Incorporating Errors . . . 69

7.6.3. Temporal Model . . . 69
7.6.4. User Input Requirements in the Error Generator . . . 70
7.7. Output Structure and Visualisation . . . 71

8. Learning the Statistical Parameters from a Real Detector 73
8.1. Available Datasets . . . 73
8.1.1. Hurtigruten Dataset . . . 73
8.1.2. Synthetic Dataset under Various Weather Conditions . . . 74
8.2. Experimental Model . . . 76
8.3. Experimental Setup . . . 76
8.3.1. Creating Training, Validation, and Test Datasets . . . 76
8.3.2. Training of Models . . . 76
8.4. Experimental Plan . . . 77
8.4.1. Experiment 1: Real-World Dataset . . . 77
8.4.2. Experiment 2: Synthetic Dataset with Good Conditions . . . 77
8.4.3. Experiment 3: Synthetic Dataset with All Conditions . . . 77
8.5. Experimental Results . . . 77
8.5.1. Experiment 1 . . . 77
8.5.2. Experiment 2 . . . 78
8.5.3. Experiment 3 . . . 79
8.6. Discussion of Experimental Results . . . 80

9. Showcasing the MODSIM System via Simulations 85
9.1. Simulation Setup . . . 85
9.1.1. Classification . . . 85
9.1.2. Annotation Mode . . . 85
9.1.3. Error Generator and Temporal Model . . . 85
9.1.4. Simulation 1 . . . 88
9.1.5. Simulation 2 . . . 89
9.1.6. Simulation 3 . . . 90
9.2. Simulation Results . . . 92
9.2.1. Simulation 1 . . . 92
9.2.2. Simulation 2 . . . 94
9.2.3. Simulation 3 . . . 96
9.3. Discussion of the Developed Simulator . . . 98

10. Conclusion and Further Work 101
10.1. Conclusion . . . 101
10.2. Further Work . . . 102
10.2.1. Evaluation of the Simulator . . . 102
10.2.2. Temporal Model . . . 102
10.2.3. Acquiring the Required Error Statistics . . . 103
10.2.4. A More In-depth Analysis of Detector Performance . . . 103
10.2.5. Dynamic Scene Generator . . . 104
10.2.6. Wave Motion . . . 105
10.2.7. Including Other Sensors . . . 105
10.2.8. Output Format of the Detector . . . 105

Bibliography 106

Appendices 111

A. Results of Experimental Bounding Box Errors **113**

B. Calculation of Extrinsic Parameters for the Camera Setup in Simulation 1 **117**

List of Figures

1.1. A simplified schematic overview of detector pipelines streamed into a tracker. . . 2
1.2. A simplified schematic overview of simulated detectors streamed into a tracker. . 2

2.1. The autonomous ferry milliAmpere. 8
2.2. Example of an object detector output. 9

3.1. World Coordinate Frame (WCF). 13
3.2. Vehicle Coordinate Frame (VCF). 14
3.3. Camera Coordinate Frame (CCF). 14

4.1. Pinhole camera model geometry 18
4.2. Central projection by similar triangles 18
4.3. Visualisation of the relationship between the WCF, CCF and image plane. 19

5.1. Visualisation of Intersection over Union (IoU). 30
5.2. A theoretical receiver operating characteristic (ROC) curve. 36
5.3. A theoretical precision-recall curve. 37

6.1. A numeric example of a stochastic automaton. 43
6.2. A stochastic automaton state sequence using dependent state transitions. 43
6.3. A stochastic automaton state sequence using independent state transitions. . . . 44

7.1. A block diagram illustrating the structure of the MODSIM system. 48
7.2. Required structure of the JSON file containing vessel characteristics. 49
7.3. Required structure of the JSON file containing pose sequences. 50
7.4. The relation between the orientation of the nVCF and the WCF 51
7.5. Illustration of the circular motion model. 52
7.6. Illustration of the circular motion of a ship 53
7.7. A plot of a pose sequence from the agents made by Henrik Fjellheim. 54
7.8. Camera base pose in MODSIM. 55
7.9. Roll, pitch and yaw in the Camera Coordinate Frame (CCF). 55
7.10. Illustration of the fixed yaw angle for a dynamic camera mounted on a vessel . . 58
7.11. Ship motion with 6 degrees of freedom, from Kornev (2012). 62
7.12. Vessel rotation on water causing pitch, roll and yaw motion. 62
7.13. Plots of first-order and second-order autoregressive processes. 64
7.14. An illustration of annotation mode 0. 67
7.15. An illustration of annotation mode 1. 67
7.16. An illustration of annotation mode 2. 68
7.17. The structure of the output generated by MODSIM. 72

8.1. Examples of annotated images in the Hurtigruten dataset. 74
8.2. Examples of annotated images in different conditions in the synthetic dataset. . . 75
8.3. Examples of the varying annotation quality. 75
8.4. Example of a false positive detection in Experiment 2. 83

8.5. Detections on the same image for Experiment 2 and Experiment 3. 83
8.6. Examples of the varying annotation quality from the synthetic training and test sets. 84

9.1. QR code to access the visualisations of Simulation 1. 92
9.2. **Simulation 1**: Vessel and camera positions in the dynamic scene at timestep 255. 93
9.3. **Simulation 1**: Projected points at timestep 255. 93
9.4. **Simulation 1**: Detections at timestep 255. 94
9.5. QR code to access the visualisations of Simulation 2. 94
9.6. **Simulation 2**: Vessel and camera positions in the dynamic scene at timestep 3899. 95
9.7. **Simulation 2**: Projected points at timestep 3899. 95
9.8. **Simulation 2**: Detections at timestep 3899. 96
9.9. QR code to access the visualisations of Simulation 3. 96
9.10. **Simulation 3**: Vessel and camera positions in the dynamic scene at timestep 3899. 97
9.11. **Simulation 3**: Projected points at timestep 2071. 97
9.12. **Simulation 3**: Detections at timestep 2071. 98

B.1. Camera placement for a static simple legacy camera in Simulation 1. 117
B.2. Illustration of the pitch angle for a static simple legacy camera in Simulation 1. . 118

List of Tables

5.1. Confusion matrix layout. 24
5.2. Probabilistic confusion matrix. 27
5.3. Probabilistic confusion matrix estimated from empirical case numbers. 27
5.4. An example of a confusion matrix of systems with multiple labels that produce AABBs. 32
5.5. Probabilistic confusion matrix estimated from empirical case numbers of systems with multiple labels that produce AABBs. 32

7.1. Required information for the vessel characteristics in the dynamic scene generator. 49
7.2. Required information for the pose sequences in the dynamic scene generator. . . . 49
7.3. Rotation steps from WCF to CCF. 56
7.4. Rotation steps from WCF to CCF for a dynamic camera at time step t. 59
7.5. Required user input for defining a static camera in the MODSIM system. 60
7.6. Required user input for defining a dynamic camera in the MODSIM system. . . . 61
7.7. The necessary information that the error generator requires from the user. 71

8.1. **Experiment 1**: Precision, recall, F_1-score, FNR and FDR. 78
8.2. **Experiment 1**: Bounding box errors in pixels (px). 78
8.3. **Experiment 1**: Covariance coefficients for the bounding box error vector value pairs. 78
8.4. **Experiment 2**: Precision, recall, F_1-score, FNR and FDR. 78
8.5. **Experiment 2**: Bounding box errors in pixels (px). 79
8.6. **Experiment 2**: Covariance coefficients for the bounding box error vector value pairs. 79
8.7. **Experiment 3**: Precision, recall, F_1-score, FNR and FDR. 79
8.8. **Experiment 3**: Bounding Box Errors in pixels (px). 80
8.9. **Experiment 3**: Covariance coefficients for the bounding box error vector value pairs. 80

9.1. Detection conditions separated into performance states. 86
9.2. The TNR, FNR, and FDR used in the temporal error model for the simulations. 86
9.3. The mean vector and covariance matrix of the bounding box errors used in the simulations. 87
9.4. Parameters for the circular motion model used in Simulation 1. 88
9.5. Camera setup in simulation 1. 88
9.6. Parameters for dynamic scene generation in Simulation 2. 89
9.7. Camera setup in simulation 2. 89
9.8. Wave motion parameters for Simulation 2. 90
9.9. Parameters for dynamic scene generation in Simulation 2. 90
9.10. Camera setup in Simulation 3 for *Camera 1*. 91
9.11. Camera setup in Simulation 3 for *Camera 2*. 91
9.12. Wave motion parameters for simulation 3. 91

A.1. **Experiment 1**: Mean vector and covariance matrix of the bounding box error vectors for real-world Conditions 113
A.2. **Experiment 2**: Mean vector and covariance matrix of the bounding box error vectors for "Noon clear" condition. 113
A.3. **Experiment 3**: Mean vector and covariance matrix of the bounding box error vectors 114
A.4. **Experiment 3**: Mean vector and covariance matrix of the bounding box error vectors 115

Abbreviations

AABB Axis-Aligned Bounding Box.

AI Artificial Intelligence.

AIS Automatic Identification Systems.

ASV Autonomous Surface Vessels.

CCF Camera Coordinate Frame.

CNN Convolutional Neural Network.

COLREG International Regulations for Preventing Collisions at Sea.

DNV Det Norske Veritas.

FDR False Discovery Rate.

FN False Negative.

FNR False Negative Rate.

FP False Positive.

GNSS Global Navigation Satellite Systems.

IMU Internal Measurement Unit.

IoU Intersection over Union.

LIDAR Light Detection and Ranging.

ML Machine Learning.

MODSIM Marine Object Detector Simulator.

NTNU Norwegian University of Science and Technology.

nVCF Nominal Vehicle Coordinate Frame.

QR Quick-Response.

rad Radians.

RADAR Radio Detection and Ranging.

SITAW Situational Awareness.

TN True Negative.

TNR True Negative Rate.

TP True Positive.

VCF Vehicle Coordinate Frame.

WCF World Coordinate Frame.

1. Introduction

1.1. Motivation and Problem Description

In recent years, the potential of Autonomous Surface Vessels (ASV) has captured the attention of the maritime industry. The development of this technology has the potential to significantly impact the industry, offering a multitude of possible applications that could make several human-based operations unnecessary. The benefits of ASVs are substantial and diverse. By removing the need for a human crew to operate vessels and handle port operations, companies can benefit from significant cost reductions. Expenses such as crew accommodation, safety equipment, air conditioning, sanitary systems, or bridges are no longer needed. Additionally, operational costs are reduced as offshore crews are no longer necessary. Furthermore, ASVs can greatly enhance marine transport safety, as most marine incidents are caused by human error. Several companies have initiated experimentation and development of ASVs, such as Yara Birkeland (Kongsberg Maritime (2023)), ReVolt (DNV GL (2015)), Zeabuz (Zeabuz (2023)). These autonomous vessels are intended for use in both cargo and human transportation, indicating progress towards achieving a fully autonomous marine environment.

Ensuring safety in ASVs is crucial to the success of this technology. The autonomous system should be designed to handle unforeseen events and failures effectively. The performance of the autonomous control systems should be at least as good as, if not better than, traditional vessels with a human captain at the command. For ASVs to function safely without continuous human control, it is essential to certify the entire autonomous system. Certification involves testing each component of the autonomous system individually and as a complete system. One such component is the external Situational Awareness (SITAW), particularly the tracker within this system. A tracker monitors the location of surrounding vessels and other relevant objects in the waterway. Figure 1.1 gives a simplified schematic overview of detector pipelines streamed into a tracker. The sensors listed in the figure are examples of sensors used for object detection on ASVs, namely cameras (CAM), Radio Detection and Ranging (RADAR), Light Detection and Ranging (LIDAR) and sound. The sensors provide measurements from the environment, and these measurements are streamed into detection algorithms that make detections of relevant objects. The detection algorithms are often data-driven models utilising Machine Learning (ML) and Artificial Intelligence (AI). It should be noted that in Figure 1.1, "AI" is used in the broadest sense to mean any algorithm or model used for inference from raw sensor data and is thus not limited to models using neural networks or other machine learning techniques. Additionally, other sensors than the ones listed in the figure, combinations of sensors through sensor fusion and processing of sensors based on available navigation information can be utilised for obstacle detection and tracking.

Full-scale testing of a tracker is challenging as it requires large amounts of test data. Obtaining real-life test data is costly and labour-intensive, and covering all significant operational scenarios would be difficult, if not impossible. One possible strategy to obtain test data is to use simulators and digital twins for testing in a synthetic environment. Simulations can be done by either reproducing the total system to a high degree of exactness or by focusing on producing a computationally inexpensive, low-level representation of the overall system.

By simulating the output behaviour of the detector rather than simulating each step in the detection pipeline, the computational cost of the simulator can be reduced. This approach eliminates the need to simulate sensor outputs with a high degree of exactness, followed by applying a real detection algorithm to the simulated data. Thereby offering the advantage of simulating errors through statistics and statistical distributions, avoiding the complexity of simulating the underlying causes of errors. The statistical distributions can be derived from empirical measurements based on observed data rather than attempting to recreate an exhaustive simulation of the entire real-life system. Figure 1.2 provides a schematic representation of simulated detections streamed into a tracker, where a simulation of the detector output replaces each sensor and detection algorithm combination in Figure 1.1.

This thesis is conducted in collaboration with Det Norske Veritas (DNV), an international company focusing on quality assurance and risk management. The company has over 150 years of experience shaping safety standards and regulations in the maritime industry. As the maritime industry moves towards increased autonomy, DNV's role in ensuring the safe adoption and integration of ASVs will be crucial in establishing trust and confidence in this emerging technology.

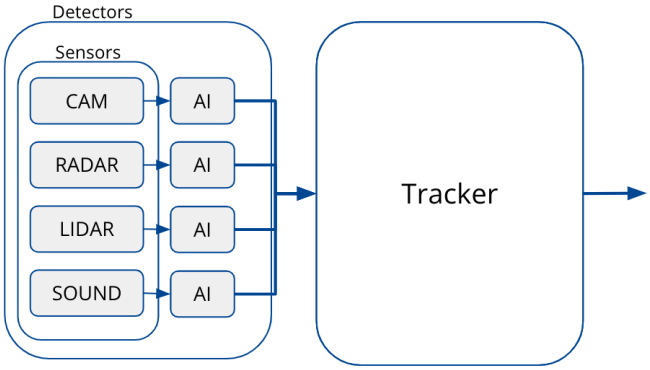


Figure 1.1.: A simplified schematic overview of detector pipelines streamed into a tracker. The term "AI" is used broadly to mean any algorithm or model used for inference from raw sensor data and is thus not limited to neural networks or other machine learning models trained on data.

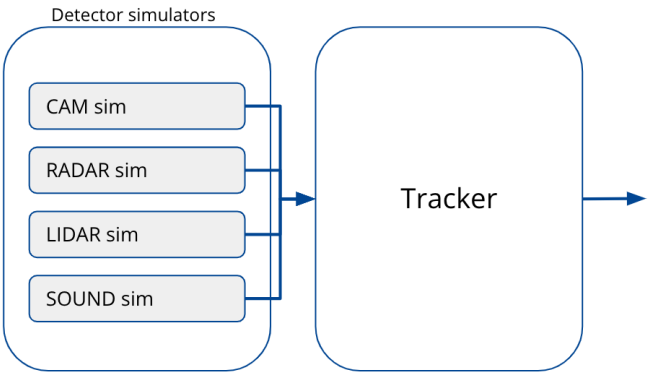


Figure 1.2.: A simplified schematic overview of simulated detectors streamed into a tracker.

1.2. Objective and Scope

The objective of this master's thesis is to design and implement an efficient system for simulating camera-based object detector output data, which can be used to assess the tracker within the external SITAW system of an Autonomous Surface Vessels (ASV). The system should replicate the characteristics of real camera-based object detectors by introducing detector errors through statistical analysis and distributions. This approach eliminates the need for complex simulations of underlying error causes. Instead, it focuses on replicating the output behaviour of the detector, enhancing simulation simplicity and efficiency. The complete camera-based object detector pipeline consists of a camera sensor and a detector algorithm, as depicted in Figure 1.1. The system developed in this thesis will correspond to the CAM sim component shown in Figure 1.2, where a simulation of the camera-based detector output replaces both the camera sensor and detection algorithm.

The scope of the thesis encompasses designing and implementing the complete system for simulating camera-based object detector output. This involves an analysis of how to characterise an object detector and understanding the factors that influence the performance of an object detector. With this knowledge, a method for replicating the errors of real camera-based object detectors through statistics and statistical distributions should be designed. The viability of the system will be assessed by performing simulations using statistical parameters. These parameters are based on a real camera-based object detector trained in a synthetic environment under various detection conditions. In-depth investigations of each component are beyond the scope of the project. Therefore, it is crucial to implement the simulator using a modular design that facilitates easy enhancement and future improvements for each component.

1.3. System Requirements

For a system to successfully and efficiently generate camera-based object detector output data for evaluating a tracker within the external SITAW system of an Autonomous Surface Vessels (ASV), it must meet certain requirements. The system requirements can be divided into the following five components:

1. **Dynamic scene generator** that contains a representation of time, a 2D map of the marine scene, and moving vessels represented as 3D shoe boxes that can be partially or fully hidden for periods. The extent of the vessels must be known, as well as the pose data, which is the position and orientation of the vessel for each time step. Pose data of the vessels can be generated directly using a motion model or by replaying previously recorded pose sequences from the real world or a synthetic environment.

2. **Camera setup** that includes the position and orientation of one or several virtual cameras. The virtual camera(s) can be stationary or mounted on one of the moving vessels. In the last case, the vessel where the camera(s) is located is referred to as the ownship. The virtual camera(s) can be moved around, observing the same scene from different perspectives. Each camera projects the eight 3D corner points of the vessels seen from the camera's perspective onto a 2D virtual image plane.

3. **Annotation generator** that generates perfect Axis-Aligned Bounding Boxes (AABBs) and classifications. A perfect AABB is a rectangle aligned with the image coordinate axes that tightly encloses the vessel without including any unnecessary surrounding areas or excluding any part of the vessel. In this thesis, unless otherwise specified, a bounding box refers to an AABB. The correct classification should be provided through the dynamic

scene as a label on the vessel object. The annotations serve as the ground truth, meaning the true bounding boxes and classifications, for a detector.

4. **Error generator** that converts the annotations to distorted bounding boxes and classifications. The distortions include missed detections, false detections, and bounding box height, width, and centre point variations. Further, the error generator should occasionally assign the incorrect classification and vary the confidence score of the classification. The extent of the distortions depends on the detector's performance, which may vary over time. This variability must be reflected in the error generator. The set of distorted detections is the final output of the camera-based object detector simulator.
5. **Visualisation** of the dynamic scene, camera setup, and simulated detections compared to the generated annotations moving and changing over time. The main objective of this system component is to enhance user-friendliness by providing a visual interface. It should enable users to observe the degree of error in the detections, and identify potential errors during code implementation.

1.4. Related Work

Traditionally, the assessment of ship control and navigation systems has relied on real-world testing with human pilots. With the emergence of autonomous ships, the complexity of such systems has increased due to the integration of probabilistic models and AI techniques. The operation of autonomous agents often demands measurements from a diverse range of sensors beyond the conventional Automated Identification System (AIS) and Inertial Measurement Unit (IMU), including cameras, LIDAR, and RADAR. Obtaining sufficient real-world data for such sensors can be both costly and difficult. As a result, there is a rising demand for simulations that can provide accurate and abundant sensor data. To overcome these challenges, researchers have turned to video game technologies to create simulators in the automotive industry.

For the automotive industry, simulators like Carla (Dosovitskiy et al. (2017)), AirSim (Shah et al. (2017)) (built with the Unreal Engine), and LGSVL (Rong et al. (2020)) (built with Unity) have been developed. In the maritime sector, the Offshore Simulation Centre (OSC) in Ålesund has been employed for education, training, and research purposes, including the study of human behaviour in offshore operations, crane control design in challenging environments, and the testing of multi-sensor systems to enhance SITAW (Vasstein et al. (2020)). It should be noted that the OSC simulator is not an open-source platform.

To address the need for an open-source marine simulation platform, Autoferry Gemini has been developed (Vasstein et al. (2020)). This simulator is based on the Unity game engine and includes sensor models for IR (infrared) camera, RADAR, VL (visible light) camera, and LIDAR. Additionally, Vasstein (2021) implemented a high-fidelity digital twin framework for testing the exteroceptive perception of autonomous vessels by improving Autoferry Gemini's LIDAR models and connecting it with other platforms to generate sensor data from scenario inputs.

Further, MathWorks (2017) has developed a generator that generates camera-based detections from a camera sensor mounted on a vehicle, namely "visionDetectionGenerator". The detection generator outputs object detections as point measurements that can be used for testing trackers. The generator provided by MathWorks was developed as part of the "Automated Driving Toolbox", which provides tools for designing, simulating, and testing autonomous car systems. However, utilising this service requires payment.

1.5. Contributions

This master's thesis aims to contribute to the research on verification of external SITAW systems on Autonomous Surface Vessels. The primary contribution is designing and developing the Marine Object Detector Simulator (MODSIM) system, which offers a novel approach to efficiently simulate camera-based detector output in marine environments. The system is designed with an emphasis on modularity providing a framework for further development.

Additionally, the thesis contributes a simple method for replicating the errors of real camera-based object detectors through statistics and statistical distributions.

Finally, the thesis offers an open-source implementation of the MODSIM system. This implementation is publicly available through a Git repository at Jessen and Mohr (2023a).

1.6. Thesis Outline

The thesis is divided into several chapters.

- **Chapter 2: Marine Situational Awareness Fundamentals**: Introduces the concepts of marine situational awareness, including sensors, object detection and tracking.
- **Chapter 3: Coordinate Systems**: Describes the coordinate systems used in this thesis and introduces methods for transformation between different coordinate systems.
- **Chapter 4: Camera Fundamentals**: Describes the fundamentals of camera technology and principles of image formation.
- **Chapter 5: Characterising a Visual Object Detector**: Introduces methods for characterising a visual object detector and presents the statistics and statistical distributions used to replicate detector error in the MODSIM system. The chapter describes the characterisation of a system outputting binary values, probabilistic values, and Axis-Aligned Bounding Boxes (AABBs).
- **Chapter 6: Characterising Object Detector Performance Under Varying Conditions**: First introduces the theoretical concepts used to model temporal variations in detector performance. Then reviews relevant literature on the influence of weather and light conditions on visual detector performance.
- **Chapter 7: A Proposed Architecture for Simulating a Marine Object Detector for Autonomous Vessels**: Presents the proposed architecture of the MODSIM system with a detailed description of each component in the system. Further, the chapter provides instructions on how to utilise the system and specifies user-required parameters.
- **Chapter 8: Learning the Statistical Parameters from a Real Detector**: Describes and discusses the conducted experiments using a real detector to obtain the required statistical parameters in the error generator.
- **Chapter 9: Showcasing the MODSIM**: Describes and discusses the produced simulations to showcase the capabilities and demonstrates how to use the MODSIM system.
- **Chapter 10: Conclusion and further work**: Summarises the conclusions of this work and suggests potential improvements and further directions for the system.

2. Marine Situational Awareness Fundamentals

Situational Awareness (SITAW) can be described as the ability to perceive and understand the various elements present within a given environment within a specific time frame and space. This involves sensing these elements, comprehending their significance, and predicting their potential future states. The external SITAW system is responsible for the surrounding environment of the autonomous vessel. In contrast, the internal SITAW system is tasked with monitoring and tracking the autonomous vessel's internal operations, state, and capabilities. This master's thesis focuses on the external SITAW system. This chapter presents the key components of the external marine SITAW system, namely sensors, object detection and tracking. The theory discussed in this chapter is inspired by Endsley (1995), Wilthil (2019), Ahrnbom (2022), and Schöller (2022).

The process of an autonomous vessel can be described as a dynamic decision-making process. Actions are performed based on decisions derived from information and goals within a given environment. Such an environment is constantly subject to change due to prior actions taken by the decision-maker or unforeseen circumstances and events beyond its control. A SITAW system is responsible for obtaining relevant information to enable the decision-making process and tracking. For external SITAW systems in Autonomous Surface Vessels (ASV), the environment is perceived by gathering and processing information from different sensors. In addition to the sensors, navigation information, such as sea charts, can be utilised. The sensors can be fused in different combinations and/or processed based on the available navigation information to improve the perceived model of the environment. The perceived information is comprehended by object detection algorithms that detect relevant objects in the surrounding environment and a tracking component which connects the detections made at different points in time into tracks. The tracks and navigation information provides a comprehensive understanding of the surrounding environment, enabling predictions on the future evolution to be made and future actions to be decided upon.

2.1. Sensors in External Marine Situational Awareness Systems

In external marine SITAW systems, the choice of sensors utilised is contingent upon various factors such as operational objectives, vessel specifications, environmental constraints, and limitations. The sensors provide a measurement model describing the relation between the target state and the measurements. The measurement model varies for each sensor and can include anything from the measurement position to the temperature. Sensors are categorised as either active or passive, with active sensors transmitting signals into the environment and measuring the resulting effects, such as in the case of RADAR and LIDAR. On the other hand, passive sensors operate by detecting changes in the environment through natural energy sources like ambient light, heat, and radiation. Examples of passive sensors include infrared (IR) and RGB camera sensors.

In conventional ships, the use of RADAR, Automatic Identification Systems (AIS) messages, and Global Navigation Satellite Systems (GNSS) is customary. For ASV, RADAR and AIS provide

useful information on nearby objects. However, only some vessels are obliged to send out AIS messages, and several smaller vessels do not have RADAR reflectors. Electro-optical sensors such as IR and RGB cameras are often utilised on autonomous vessels to supplement the traditional sensors. This is exemplified by the autonomous ferry milliAmpere where the sensors used include GNSS, LIDAR, RADAR, cameras, IMU, and ultrasonic distance sensors (Brekke et al. (2022)). milliAmpere is developed as part of the Autoferry project at NTNU, which involves the research of an autonomous all-electric passenger ferry in an urban environment (NTNU (2023)), the ferry can be viewed in Figure 2.1.

Figure 2.1.: The autonomous ferry milliAmpere developed as part of the Autoferry project at NTNU. In courtesy of Kai Dragland (NTNU (2023)).

2.2. Object Detection

The primary objective of object detection is to identify objects' positions in the environment by processing information received through various sensors. By accurately detecting objects in the vicinity, ASVs can make informed decisions to avoid collisions and navigate safely. The previous section introduced several commonly used sensors in object detection for ASVs. The data from these sensors are processed using Artificial Intelligence (AI) and Machine Learning (ML) techniques, allowing the autonomous system to understand and interpret the objects' positions and characteristics.

Additionally, object detection often includes the task of classifying detected objects into predefined categories. If the categories are chosen carefully, this knowledge can assist the ASV in adhering to the International Regulations for Preventing Collisions at Sea (COLREG), which consists of a set of rules and regulations designed to govern the behaviour of vessels to prevent collisions and ensure the safety of navigation. These regulations consider the current situation and behaviour of the vessels in addition to the type of vessel. For instance, different regulations apply to fishing vessels depending on whether they are actively engaged in fishing. Similarly, power-driven vessels have distinct guidelines when they are anchored versus underway. Most visual object detectors classify vessels solely based on the vessel type without providing insights into the vessel's intentions or ongoing actions, rendering the categories inadequate for assisting the ASV in complying with COLREGs.

2.2.1. Visual Object Detection

A visual object detector operates on images and can be modelled as a collection of multiple individual detectors, each focusing on a small sub-region of the complete image area. The primary function of each detector is to perform a binary detection, i.e., to determine whether an object is present in the examined image cell.

In addition to binary information, some detectors can provide further details about the detected object. These may include classifying the object, finding its spatial extent, and finding its speed and direction vector represented in image coordinates. Other characteristics may also be possible. The extent of objects in images and videos is typically represented as either Axis-Aligned Bounding Box (AABB) or an object mask, represented by binary object/no-object labels on the pixel grid. Figure 2.2 displays an example of detections made by a visual object detector. Each detection includes an AABB, the predicted object class, and a confidence score. A confidence score is a numeric value that denotes the degree of certainty or confidence exhibited by the model in its prediction for a specific instance.

Figure 2.2.: Example of an object detector output, taken from (Schöller, 2022, 18). Each detection comprises an Axis-Aligned Bounding Box (AABB) indicating the spatial extent of the detected object, an assigned object class label, and a confidence score representing the detector's confidence level in associating the object with that specific class.

Visual object detection and classification are complex problems commonly solved using Machine Learning (ML). The advantage of ML is that it provides a framework that empowers computers to automatically learn the rules required to solve these problems, unlike traditional algorithms that require manually defining the necessary rules. To model the problem and discover the rules, ML needs access to a dataset. A dataset comprises a collection of input data, such as images, along with the ground truth of the input. The ground truth refers to the desired output of the model given the input. For example, in the context of marine vessel detection, if the model is presented with an image of a marine vessel, the ground truth would include the location of the vessel, possibly in the form of an AABB, as well as the vessel's classification. The quality of the ground truth is crucial as it serves as the standard against which the detector's performance is evaluated. Accurate and representative ground truth data ensures the detector learns meaningful patterns and behaviours, leading to reliable and robust outputs when deployed in practical applications.

Visual object detectors can broadly be categorised into two types: two-stage and one-stage detectors. A two-stage detector divides the network into two separate stages. First, a region proposal

network generates a set of AABBs that are likely to contain objects in the given image. The region proposal network is followed by a classifier that classifies and refines the proposals provided by the first stage. A known example of a two-stage visual object detector is R-CNN (Girshick et al. (2013)), and approaches that are based on R-CNN, namely Faster R-CNN (Ren et al. (2015)) and Mask R-CNN (He et al. (2017)). One-stage visual object detectors construct a single network performing both the localisation and classification tasks simultaneously, resulting in only a single pass over the input image. Examples of implementations of one-stage detectors are SSD (Liu et al. (2015)) and YOLO (Redmon et al. (2015)) and its multiple versions. Two-stage detectors are typically characterised by higher accuracy but slower processing speeds, whereas one-stage detectors prioritise speed and may sacrifice some accuracy. This makes one-stage detectors particularly relevant for real-time applications.

According to (Ahrnbom, 2022, 25), Convolutional Neural Networks (CNNs) are the state-of-the-art tool for solving visual object detection. They are widely used as the underlying feature extractor in two-stage and one-stage detectors. CNNs have shown to be very effective in learning to recognise patterns in data. A CNN comprises three main building blocks: convolutional layer, pooling, and activation. The convolutional layer uses learnable filters to perform convolutions over the input. The operation of convolutional layers is based on the assumption that significant image features are often local. The filter also called a kernel, incorporates various hyperparameters, including kernel size, stride, and padding. Multiple kernels are typically present in a convolutional layer. Once the kernel is convolved over the input image, the resulting output is known as a feature map or activation map. The number of channels in the feature map corresponds to the number of kernels in the layer. A pooling layer is used to decrease the size of the feature map outputted by the convolutional layer. Convolution is a linear operation, which is why activation functions are frequently incorporated after a convolutional layer. This allows for the modelling of nonlinear inputs. Common activation functions used are Sigmoid and Rectified Linear Unit (ReLU).

Evaluating the performance of object detection systems is crucial to assess their accuracy and reliability. Several metrics are commonly used to measure the performance of these systems. These evaluation metrics facilitate in comparing and establishing benchmarks among different object detection algorithms and enable assessment of their performance. Evaluation metrics for visual object detectors are elaborated on in Chapter 5.

2.3. Tracking of Marine Objects

Tracking is the task of connecting detections at distinct points in time to form tracks, whereby each object should be connected to only one track. In the context of maritime situations, this involves identifying the tracks of vessels moving or staying still on the sea. The tracks provide an overview of marine vessels' current and past positions and can facilitate the estimation of their future position. In object tracking, the range of potential issues that may arise is considerably broader than those encountered in single-image tasks. These issues may include mixing or intertwining tracks of objects, assigning multiple tracks to an object, and improperly splitting or combining tracks.

In the case of external marine situational awareness, the task of tracking can be characterised as multi-object tracking. This involves associating multiple detections to distinct tracks, while also creating and removing tracks as objects move through a scene. In contrast, single-object tracking typically involves tracking a single object with a known initial position. In single-object tracking, tracks are calculated without considering the possibility that the detections originate from other objects in the surrounding area.

Various sensors can produce different inputs for the tracker. Certain sensors generate a single point in 3D space as a detection outcome, whereas others provide a more detailed representation of the detected object, including its size and shape. The former approach is referred to as point tracking, whereas the latter is called extended object tracking. Detection with camera sensors provides information about objects' positions and dimensions in an image. Therefore, tracking with camera sensors is an example of extended object tracking.

A common approach to solving the complex task of multi-object tracking is tracking-by-detection. This approach involves data association of detections in subsequent images. Data association is the task of calculating association probabilities that reflect the likelihood of a detection belonging to a given track. As stated in (Wilthil, 2019, 13), data association represents the primary challenge in object tracking. The sensor measurements lack labels indicating the object of origin. Consequently, data association is crucial for identifying the measurements as clutter (i.e. noise and unwanted measurements), an existing target, or a new target. There are several different approaches to performing data association in the context of tracking. Some tracking algorithms, known as class-agnostic, use motion and appearance characteristics for data association and do not rely on object class information. Conversely, other tracking algorithms consider the object class, employing class-specific models and assumptions for data association. Traditionally, trackers have been class-agnostic, treating objects generically and focusing on spatial and temporal information. However, advancements in computer vision and deep learning have led to the emergence of class-aware trackers, which leverage object-specific knowledge for improved tracking performance.

2.3.1. Evaluating the Performance of Tracker Systems

Accurate and reliable tracking is an important aspect of an autonomous vessel's situational awareness. Therefore, quantifying and measuring tracker performance is essential. Helgesen et al. (2022) presents various performance measures for tracking based on data association between state estimates and the ground truth. This section presents the tracker performance metrics described. Further details can be found in Helgesen et al. (2022).

Root Mean Square Error (RMSE)

One fundamental measure is the Root Mean Square Error (RMSE), which indicates the average error of individual tracks. The position RMSE compares tracking estimates to ground truth positions. Defining $\bar{\boldsymbol{e}_i}$ as the error, the RMSE can be calculated as follows,

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} |\bar{\boldsymbol{e}_i}|^2} \tag{2.1}$$

Average Normalised Estimation Error Squared (ANEES)

The Normalised Estimation Error Squared (NEES) evaluates statistical consistency by assessing the relationship between state errors and the estimated covariance matrix. If the state errors are accurately described, the tracking output is regarded as statistically consistent. NEES is defined as follows,

$$NEES_k = \bar{\boldsymbol{x}}_i \boldsymbol{P}_i^{-1} \bar{\boldsymbol{x}}_i \tag{2.2}$$

where $\bar{\boldsymbol{x}}_i$ is the estimated error and P_i^{-1} is the error covariance matrix.

The ANEES (average NEES) is chi-squared distributed and is found by taking the average of the NEES across all time steps,

$$ANEES = \sum_{k=1}^{n} NEES_k \tag{2.3}$$

Track Establishment

The mean track establishment time measures the tracker's ability to establish tracks quickly. The tracker's ability to establish a track rapidly is desirable, especially when the tracker is operating in an environment where targets can appear close to the vessel. Track establishment occurs once a target is first associated with a track.

Track Breaks

Track breaks, both in terms of number and length, indicate the tracking system's performance in maintaining valid tracks. Track breaks can occur due to factors like obscuration by larger vessels.

False Tracks

The total number and length of false tracks serve as measures of tracker performance. False tracks are those that do not originate from valid targets.

The General Optimal Subpattern Assignment (GOSPA)

The General Optimal Subpattern Assignment (GOSPA) is a metric that considers localisation errors, false tracks, and missed targets. Let the set of estimated tracks at time k be defined as $\boldsymbol{X}_k = [\boldsymbol{x}_k^1, \dots \boldsymbol{x}_k^m]$ and the set of ground truth tracks $\boldsymbol{Y}_k = [\boldsymbol{y}_k^1, \dots \boldsymbol{y}_k^n]$, then the GOSPA metric is defined as follows,

$$GOSPA = \left(\min_{\pi \in \prod_n} \sum_{i=1}^{m} d^{(c)}(\boldsymbol{x}_k^i, \boldsymbol{y}_k^{\pi(i)})^p + \frac{c^p}{\alpha}(n-m) \right)^{\frac{1}{p}} \tag{2.4}$$

Where $\prod_n$ is the set of all permutations of $\{1, \dots, n\}$. $d(x, y)$ is a metric for track-truth distance and $d^{(c)}(x, y) = \min(d(x, y), c)$ the distance cut-off given by the parameter c.

3. Coordinate Systems

This chapter describes the coordinate systems used in the thesis and introduces the transformations between such coordinate systems.

3.1. Definition of Coordinate Systems

3.1.1. World Coordinate Frame (WCF)

The World Coordinate Frame (WCF) describes the orientation and position of the marine vessels in the dynamic scene. Figure 3.1 describes the world coordinate frame. The order of the coordinate frame unit vectors is $(\vec{w}_1, \vec{w}_2, \vec{w}_3)$. The w_3-axis is always perpendicular to the 2D marine surface of the dynamic scene, i.e. pointing towards the sky. The w_1-axis and w_2-axis are parallel to the marine surface plane. All the unit vectors point in the positive direction and are orthogonal to each other. Further, the WCF is right-handed. The unit of length in the WCF is metre (m).

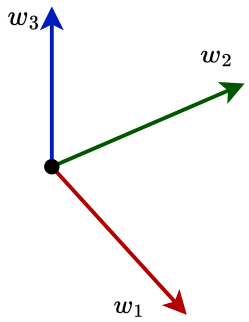


Figure 3.1.: World Coordinate Frame (WCF).

3.1.2. Vehicle Coordinate Frame (VCF)

The Vehicle Coordinate Frame (VCF) is always fixed to the vehicle, as illustrated in Figure 3.2. The origo of the VCF is defined at the centre of the bottom plane of the vehicle. The order of the coordinate frame unit vectors is $(\vec{v}_1, \vec{v}_2, \vec{v}_3)$. The VCF is right-handed, and all unit vectors are orthogonal. The v_1-axis is always pointing in the forward direction of the vehicle. When looking in the direction of the first unit vector, the third unit vector $\vec{v}_3$ points upward of the vehicle, i.e. orthogonal to the bottom plane of the vehicle. The second unit vector $\vec{v}_2$ points 90 degrees leftward of the vehicle. All the unit vectors point in the positive direction, and the unit of length in the VCF is metre (m).

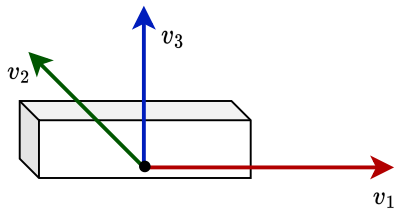


Figure 3.2.: Vehicle Coordinate Frame (VCF).

3.1.3. Nominal Vehicle Coordinate Frame (nVCF)

The VCF is always fixed to the vehicle. This is not the case for the Nominal Vehicle Coordinate Frame. The nVCF follows the 2D moving direction of the vehicle, i.e. the heading direction on the resulting plane of $\vec{w}_1$ and $\vec{w}_2$. The nVCF is right-handed, and the coordinate frame unit vector order is $(\vec{v}_1^n, \vec{v}_2^n, \vec{v}_3^n)$. The third unit vector of the nVCF $\vec{v}_3^n$ is always parallel to the third unit vector of the WCF $\vec{w}_3$. All the unit vectors point in the positive direction and are orthogonal to each other, and the unit of length is metre (m). VCF is equal to nVCF (VCF = nVCF) when there is no rotation about the first or second unit vector of the VCF ($\vec{v}_1$ and $\vec{v}_2$).

3.1.4. Camera Coordinate Frame (CCF)

The Camera Coordinate Frame (CCF) is attached to the camera and represents the camera's viewport. The frame is represented by an origin and three orthogonal unit vectors in the following order, $(\vec{c}_1, \vec{c}_2, \vec{c}_3)$. The origin is the optical centre, and the c_3-axis is the optical axis. The third unit vector $\vec{c}_3$ points in the direction of the view. When looking in the direction of $\vec{c}_3$, the second unit vector $\vec{c}_2$ points downwards, and the first unit vector $\vec{c}_1$ points to the right. All the unit vectors point in the positive direction, and the CCF is right-handed. The unit of length is metre (m).

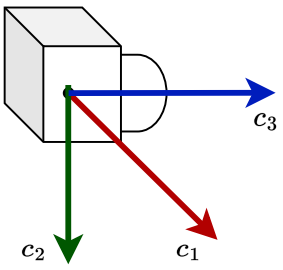


Figure 3.3.: Camera Coordinate Frame (CCF).

3.2. Transformation Between Coordinate Systems

3.2.1. Notation

In this master's thesis, a point is denoted using lowercase, for instance, p. A point in a specific coordinate frame K is denoted using an upper script, i.e. p^K.

3.2.2. Rotation and Translation Between Coordinate Systems

In the context of expressing coordinates in one coordinate frame when they are defined in another coordinate frame, it is necessary to perform rotation and translation. The right-hand rule applies universally to all rotations about an axis. If the thumb of the right hand follows the axis of rotation, the other fingers show the direction of the rotation. The rotation from coordinate system A to B can be expressed using a 3x3 rotation matrix R_B^A. Similarly, the translation from coordinate system A to B can be expressed using a 3x1 translation vector $\vec{t}_B^A$. Considering an arbitrary point p in 3D space, p^B represents the coordinates of point p in the coordinate system B, and p^A represents the same point p in the coordinate system A. To perform the translation and rotation of coordinates in the coordinate system A to coordinate system B, the following equation can be used:

$$p^B = R_B^A \cdot p^A + \vec{t}_B^A \tag{3.1}$$

The Cartesian coordinates can be expressed as homogeneous coordinates to represent the rotation and translation as matrix multiplication. To do so, the 4x4 matrix $\mathbf{M}_B^A$ is defined as the composition of the rotation and translation,

$$\mathbf{M}_B^A = \begin{bmatrix} \mathbf{R}_B^A & \vec{t}_B^A \\ \vec{0}^T & 1 \end{bmatrix}, \tag{3.2}$$

where $\vec{0}^T = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$. The rotation and translation of the homogeneous coordinates are given as,

$$y^B = \mathbf{M}_B^A \cdot y^A, \tag{3.3}$$

where y is a 4×1 homogeneous coordinate.

3.2.3. Rotation Using Euler Angles

A rotation about an arbitrary rotation axis can be described using Euler angles. This is done by a sequence of partial rotations about the three axes of a defined coordinate system. As rotations are not commutative, the order of the rotations has to be defined clearly, and it is important always to do the rotations in that consecutive order. In the 3D world, there are three coordinate frame axes that can be rotated about. After rotating about one of these axes, a frame that is not aligned with the initial coordinate frame is obtained. For the next rotation operation, it is therefore essential to state whether the rotation is about the *initial frame* or the new frame, namely the *moving frame*. While both options are viable, it is not logical to switch between them in the sequence of three rotations.

Define an arbitrary coordinate system U, with $(\vec{u}_1, \vec{u}_2, \vec{u}_3)$ as the first, second, and third unit vectors, respectively, as an illustrative example. Further, counterclockwise is defined as the positive orientation direction. Let K be the resulting frame after rotating about the three unit vectors of the initial frame U. A point p expressed in the initial frame U as p^U is related to the point expressed in K as p^K by the following equation:

$$p^K = \mathbf{R}_M^U \cdot p^U \tag{3.4}$$

Where $\mathbf{R}_K^U$ is the rotation matrix from the initial frame U to the rotated frame K. This matrix is the product of three rotation matrices, denoted $\mathbf{R}_{\vec{u}_1}$, $\mathbf{R}_{\vec{u}_2}$, and $\mathbf{R}_{\vec{u}_3}$ each corresponding to a rotation about the first, second and third unit vector respectively.

The rotation matrix for rotation of ϕ about the 1st unit vector $\vec{u}_1$ is defined as,

$$\mathbf{R}_{\vec{u}_1}(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}. \tag{3.5}$$

The rotation matrix for rotation of ψ about the 2nd unit vector $\vec{u}_2$ is defined as,

$$\mathbf{R}_{\vec{u}_2}(\psi) = \begin{bmatrix} \cos\psi & 0 & \sin\psi \\ 0 & 1 & 0 \\ -\sin\psi & 0 & \cos\psi \end{bmatrix}. \tag{3.6}$$

The rotation matrix for rotation of θ about the 3rd unit vector $\vec{u}_3$ is defined as,

$$\mathbf{R}_{\vec{u}_3}(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.7}$$

The order of the matrix multiplication corresponds to the order in which the rotations are performed and should be clearly stated. When rotating with respect to the moving frame, the rotation order is in the post-multiplication order. However, when rotating with respect to the initial frame, subsequent rotations should be pre-multiplied.

4. Camera Fundamentals

The following sections presents the fundamentals of camera technology and the principles of image formation. The basic pinhole camera model is introduced first, followed by a more comprehensive camera model that encompasses CCD (Charge-coupled device) sensors found in digital cameras. This comprehensive model is utilised for image formation in the MODSIM system. This section is largely based on the work of Hartley and Zisserman (2003).

4.1. Pinhole Camera

The pinhole camera model is a fundamental concept used to represent how a camera captures an image. This model assumes that light from a scene passes through a single point, the camera's aperture or pinhole, and creates a reversed, upside-down image on a plane, namely the image plane.

The camera centre, or optical centre, is positioned at the origin of a Euclidean coordinate system, namely the camera coordinate frame (CCF). The image plane is located at a fixed distance f in front of the camera, along the third unit vector of CCF, c_3. This distance f is called the focal length. The line through the camera centre perpendicular to the image plane is called the principal axis and meets the image plane at the principal point. The geometry of the pinhole camera is illustrated in Figure 4.1.

When the camera captures an image, each point in the 3D world corresponds to a point on the 2D image plane. To determine where a particular point, p^C, in CCF will appear on the image plane, a line is drawn from the optical centre through that 3D point. The point where the line intersects the image plane, p^I, is the projected image of the 3D point. This process is known as central projection and is illustrated in Figure 4.1 and Figure 4.2. By similar triangles, it is easy to compute that,

$$[p_1^C, p_2^C, p_3^C]^T \mapsto [f\frac{p_1^C}{p_3^C}, f\frac{p_2^C}{p_3^C}, f]^T. \tag{4.1}$$

Ignoring the final image coordinate, which is a constant, the central projection is described by,

$$[p_1^C, p_2^C, p_3^C]^T \mapsto [f\frac{p_1^C}{p_3^C}, f\frac{p_2^C}{p_3^C}]^T. \tag{4.2}$$

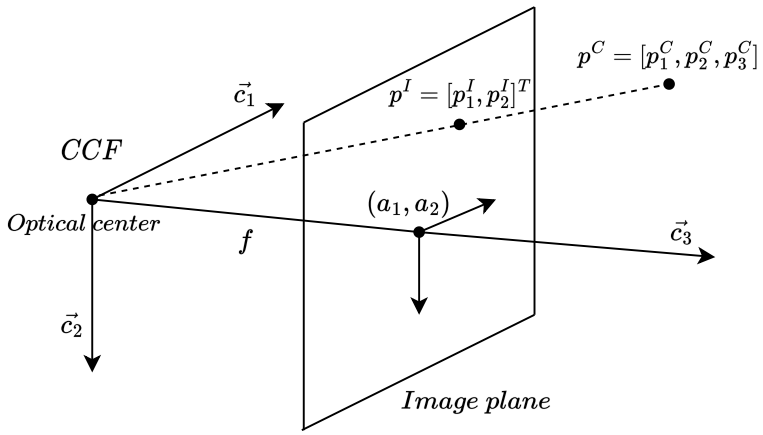


Figure 4.1.: The pinhole camera model geometry. Point (a_1, a_2) is the principal point, p^C is a point in CCF, and p^I is the mapped point on the image plane. The figure is inspired by Hartley and Zisserman (2003).

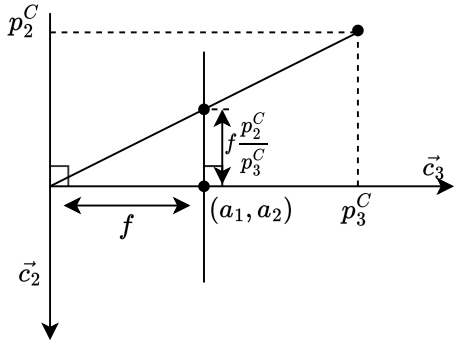


Figure 4.2.: Central projection by similar triangles. The figure is inspired by Hartley and Zisserman (2003).

Using homogeneous coordinates for the 3D point, the projection can be represented by matrix multiplication,

$$\begin{bmatrix} p_1^C \\ p_2^C \\ p_3^C \\ 1 \end{bmatrix} \mapsto \begin{bmatrix} f\frac{p_1^C}{p_3^C} \\ f\frac{p_2^C}{p_3^C} \\ 1 \end{bmatrix} = \begin{bmatrix} fp_1^C \\ fp_2^C \\ p_3^C \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} p_1^C \\ p_2^C \\ p_3^C \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} [\mathbf{I}|\vec{0}] \begin{bmatrix} p_1^C \\ p_2^C \\ p_3^C \\ 1 \end{bmatrix}. \tag{4.3}$$

By defining the projection matrix as,

$$\mathbf{P} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} [\mathbf{I}|\vec{0}], \tag{4.4}$$

the central projection can be written more compactly,

$$m^I = \mathbf{P} m^C \tag{4.5}$$

where m^I is the homogeneous coordinates of the 2D image point p^I, and m^C is the homogeneous coordinates of the 3D point p^C.

4.2. A More Comprehensive Camera Model

Although straightforward in its computational implementation, the pinhole camera model proves insufficient for real-world scenario projections. To obtain more accurate projections, it is essential to consider additional factors, specifically image digitalisation and the 6 degrees of freedom rigid motion between the camera coordinate frame (CCF) and the world coordinate frame (WCF), in which points in space are expressed. A more precise projection of 3D coordinates to 2D points in an image plane is accomplished using the extrinsic and intrinsic camera parameters. The intrinsic camera parameters model the camera's characteristics and transform the 3D coordinates in CCF to image coordinates in pixel units on the image plane. The extrinsic parameters convert the coordinates from the WCF to the CCF. The relationship between the WCF, CCF and the image plane is illustrated in Figure 4.3.

It is important to note that real lenses introduce distortions with reference to the plain pinhole camera model. However, for the sake of simplicity and the scope of this project, the lens distortions will be disregarded.

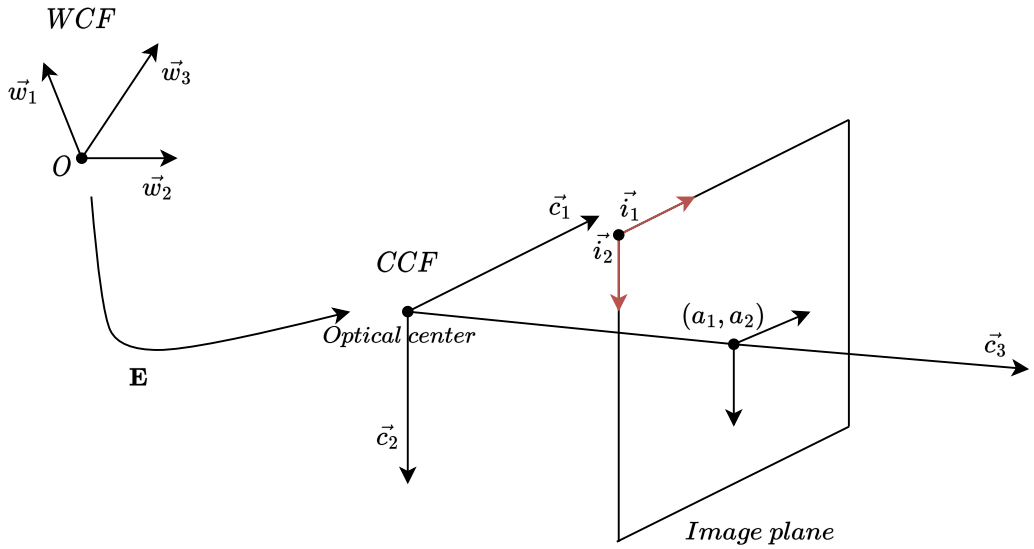


Figure 4.3.: Visualisation of the relationship between the WCF, CCF and image plane, where **E** is the extrinsic matrix and (a_1, a_2) is the principal point.

4.2.1. The Intrinsic Camera Parameters

The intrinsic camera parameters model the characteristics of the camera. In the basic pinhole model, it is assumed that the origin of the image coordinates is at the principle point (a_1, a_2), which is not valid for most digital cameras. Therefore, in general, there exists a mapping for a point in CCF, p^C, to a point on the image plane,

$$[p_1^C, p_2^C, p_3^C]^T \mapsto [f\frac{p_1^C}{p_3^C} + a_1, f\frac{p_2^C}{p_3^C} + a_2]^T. \tag{4.6}$$

Moreover, it is necessary to account for the quantisation associated with the horizontal pixel size (Δp_1) and vertical pixel size (Δp_2). In the general pinhole model, it is assumed that image coordinates are Euclidean coordinates having equal scales in both axial directions. If image coordinates are measured in pixels, non-square pixels introduce unequal scale factors in each direction. Accounting for the pixel size gives the following mapping,

$$[p_1^C, p_2^C, p_3^C]^T \mapsto [f_1 \frac{p_1^C}{p_3^C} + a_1, f_2 \frac{p_2^C}{p_3^C} + a_2]^T, \tag{4.7}$$

where

$$f_1 = \frac{1}{\Delta p_1} f, \tag{4.8}$$

$$f_2 = \frac{1}{\Delta p_2} f. \tag{4.9}$$

This equation may be expressed conveniently in homogeneous coordinates as

$$\begin{bmatrix} p_1^C \\ p_2^C \\ p_3^C \\ 1 \end{bmatrix} \mapsto \begin{bmatrix} f_1 \frac{p_1^C}{p_3^C} + a_1 \\ f_2 \frac{p_2^C}{p_3^C} + a_2 \\ 1 \end{bmatrix} = \begin{bmatrix} f_1 & 0 & a_1 & 0 \\ 0 & f_2 & a_2 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} p_1^C \\ p_2^C \\ p_3^C \\ 1 \end{bmatrix} \tag{4.10}$$

Defining the intrinsic camera matrix $\mathbf{K}$ as,

$$\mathbf{K} = \begin{bmatrix} f_1 & 0 & a_1 \\ 0 & f_1 & a_2 \\ 0 & 0 & 1 \end{bmatrix}, \tag{4.11}$$

Equation (4.10) has the concise form,

$$m^I = \mathbf{K}[\mathbf{I}|\vec{0}] \cdot m^C \tag{4.12}$$

where m^I is the homogeneous coordinates of the 2D image point p^I, and m^C is the homogeneous coordinates of the 3D point p^C.

To generalise further, one can include a fifth parameter skew (s) to account for non-orthogonality between the axis of the image plane. This leads to the following intrinsic camera matrix,

$$\mathbf{K} = \begin{bmatrix} f_1 & s & a_1 \\ 0 & f_1 & a_2 \\ 0 & 0 & 1 \end{bmatrix}, \tag{4.13}$$

For most cameras, it is assumed orthogonality between the axis of the image plane, giving zero skew ($s = 0$).

4.2.2. The Extrinsic Camera Parameters

In the presented pinhole camera model, the projected point p^C is defined in the CCF. However, in practical applications, 3D coordinates are defined in WCF external to the CCF. The WCF and CCF are related by rotation and translation. Therefore, to obtain the corresponding point in the CCF, p^C, for a given point in the WCF, p^W, the following equation is employed:

$$p^C = \mathbf{R}_C^W p^W + \vec{t}_C^W \tag{4.14}$$

where $\mathbf{R}_C^W$ is a 3x3 rotation matrix, and $\vec{t}_C^W$ is the translation vector. The rotation matrix and translation vector can be summarised in an extrinsic matrix,

$$\mathbf{E} = \begin{bmatrix} \mathbf{R}_C^W & \vec{t}_C^W \end{bmatrix}. \tag{4.15}$$

So that a transformation from a point p^W in WCF to the corresponding point in CCF, p^C, is expressed as,

$$p^C = \mathbf{E} p^W. \tag{4.16}$$

4.2.3. Projection of Real-World Coordinates to the Image Plane

Projection of real-world coordinates is achieved by combining the intrinsic and extrinsic parameters in a 3x4 projection matrix,

$$\mathbf{P} = \mathbf{K} \cdot \begin{bmatrix} \mathbf{R}_C^W & \vec{t}_C^W \end{bmatrix}. \tag{4.17}$$

Assuming affine projections of 3D points in the WCF to the image plane, the projection is calculated as follows,

$$k \cdot m^I = \mathbf{P} \cdot m^W, \tag{4.18}$$

where $m^W = \begin{bmatrix} p_1^W & p_2^W & p_3^W & 1 \end{bmatrix}^T$ is the homogeneous coordinates of a 3D point in WCF, $m^I = \begin{bmatrix} p_1^I & p_2^I & 1 \end{bmatrix}^T$ is homogeneous coordinates of the resulting 2D image point expressed in pixel units, and k is the ambiguous proportionality factor due to calculations with homogeneous coordinates.

If the resulting image coordinates after the projection of a 3D point are outside the image bounds (i.e. the width and height of the image) or behind the camera, the point does not lie on the image plane. These 3D points are outside the field of view of the camera and should not be included in the resulting image.

5. Characterising a Visual Object Detector

Characterising the performance of a detector is a critical aspect of evaluating its effectiveness in detecting and classifying objects accurately. The primary goal of characterising the performance of a detector is to provide statistical characterisations that quantify the correctness of the values generated by the detector. This chapter elaborates on how to characterise visual object detectors.

5.1. Characterising a Binary Decision-Making System

A detector that outputs a binary value is a binary decision-making system. A binary decision-making system is a two-classification problem, such as whether a target is present or not, or whether a disease is present or not. It is important to note that in this context, it is assumed that the detector does not possess a memory, meaning it does not consider previous input data or decisions.

When characterising the performance of such a system, it is necessary to differentiate between characterisations based on probabilistic concepts and those based on empirical data and concepts. Statistical characterisations based on probabilistic concepts typically involve using probability theory to model the system's behaviour. On the other hand, characterisations based on empirical data and concepts rely on analysing data obtained from experiments or observations.

5.2. Empirical Characterisation of Binary Decision-Making Systems

For empirical characterisation of binary decision-making systems, the following case numbers can be defined:

1. The true number of positive cases in the dataset: P
2. The true number of negative cases in the dataset: N
3. True positives, the correctly classified positive cases: TP
4. True negatives, the correctly classified negative cases: TN
5. False positives, the negative cases that were classified as positive cases: FP
6. False negatives, the positive cases that were classified as negative cases: FN
7. Estimated positives, the total number of instances classified as positives: EP
8. Estimated negatives, the total number of instances classified as negatives: EN

where positives mean target present and negative means no target present. The case numbers have the following relationships:

1. $P = TP + FN$

2. $N = TN + FP$
3. $EP = TP + FP$
4. $EN = TN + FN$
5. $EN + EP = P + N$,

and can be summarised in a confusion matrix. The layout of such a matrix is shown in Table 5.1.

Decision result (d)	Actual values (t)		
		Positive	**Negative**
	Positive	TP	FP
	Negative	FN	TN

Table 5.1.: Confusion matrix layout.

5.2.1. Metrics for Performance Measuring of Binary Decision-Making Systems

The empirical case numbers increase as the number of trials increases. More stable metrics are obtained by creating relative case numbers, meaning normalising the case numbers against the total of positives and/or negatives. The relative case numbers are less impacted by changes in the number of trials. As the number of trials increases, the relative case numbers will converge towards probabilities. There are a number of different metrics for performance measuring, which can be derived by creating relative case numbers from these four fundamental elements of the confusion matrix.

Precision

Precision describes how many positive classifications, meaning the instances classified as 'target present', that was correct. Precision is defined as follows,

$$\textbf{Precision} = \frac{TP}{TP + FP} = \frac{TP}{EP}. \tag{5.1}$$

Recall

Recall, or true positive rate (TPR), describes how many of the actual positives were classified correctly. The recall is defined as follows,

$$\textbf{Recall (TPR)} = \frac{TP}{TP + FN} = \frac{TP}{P}. \tag{5.2}$$

Accuracy

Accuracy is the proportion of classifications that the system classified correctly and is defined as follows,

$$\textbf{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN} = \frac{TP + TN}{P + N}. \tag{5.3}$$

Specificity

Specificity, or true negative rate (TNR), measures the proportion of actual negatives that were classified as negatives. Specificity is defined as follows,

$$\textbf{Specificity} \quad \textbf{(TNR)} = \frac{TN}{FN + TN} = \frac{TN}{EN}. \tag{5.4}$$

Fall-out

Fall-out, or false positive rate (FPR), measures how often an actual negative instance is classified as a positive instance. The fall-out is the inverse of specificity and is defined as follows,

$$\textbf{Fall-out (FPR)} = 1 - \textbf{Specificity} = \frac{FP}{FP + TN}. \tag{5.5}$$

False discovery rate

False discovery rate (FDR) measures how many of the positive classifications were incorrect. It is defined as follows,

$$\textbf{False discovery rate (FDR)} = \frac{FP}{FP + TP}. \tag{5.6}$$

False negative rate

False negative rate (FNR), also called miss rate, measures how often an actual positive instance is classified as a negative instance. The false negative rate is the inverse of recall and is defined as follows,

$$\textbf{False negative rate (FNR)} = \frac{FN}{TP + FN} = \frac{FN}{P} = 1 - \textbf{Recall}. \tag{5.7}$$

F_1-score

F_1-score is the harmonic mean of precision and recall and measures the preciseness and robustness of a system. The harmonic mean of the variables $x_1, ..., x_n$ is the reciprocal of the arithmetic mean of the reciprocals of the variables and is defined as follows,

$$H = \left(\frac{\sum_{i=1}^{n} x_i^{-1}}{n} \right)^{-1} = \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}. \tag{5.8}$$

The definition of the F_1-score is, therefore

$$F_1 = \frac{2}{\frac{1}{\textbf{Precision}} + \frac{1}{\textbf{Recall}}} = 2 \cdot \frac{\textbf{Precision} \cdot \textbf{Recall}}{\textbf{Precision} + \textbf{Recall}}. \tag{5.9}$$

F_β-score

A more general approach to the F_1-score is the F_β-score, where β is a balancing factor for weighting the harmonic mean of precision and recall. When $\beta = 1$, the F_β-score equals the F_1-score and precision and recall are weighted equally. Increasing β increases the weight for recall, allowing for higher tolerance of false positives. Conversely, decreasing β places more weight on precision, resulting in a higher tolerance for false negatives.

The weighted harmonic mean of variables $x_1, ..., x_n$ is defined as,

$$H_{weighted} = \frac{\sum_{i=1}^{n} w_i}{\sum_{i=1}^{n} \frac{w_i}{x_i}}, \tag{5.10}$$

where w_i is the weight for variable x_i.

To weigh recall β times more than precision, recall can be assigned weights of $\frac{\beta^2}{\beta^2+1}$ while precision can be assigned weights of $\frac{1}{\beta^2+1}$ [1]. By using the weighted harmonic mean Equation (5.10) the F_β-score is defined as follows,

$$F_\beta = \frac{\frac{1}{\beta^2+1} + \frac{\beta^2}{\beta^2+1}}{\frac{1}{\beta^2+1}\frac{1}{\mathbf{Precision}} + \frac{\beta^2}{\beta^2+1}\frac{1}{\mathbf{Recall}}} = (1+\beta^2)\frac{\mathbf{Precision} \cdot \mathbf{Recall}}{\beta^2\mathbf{Precision} + \mathbf{Recall}}. \tag{5.11}$$

5.2.2. Imbalanced Dataset

Accuracy depends on the ratio of positive to negative instances and should only be utilised when the dataset is balanced, meaning positive and negative counts are close. For an imbalanced dataset where the vast majority of examples belong to a single class, the accuracy metric can give a misleading performance measure. Assume the majority of instances in the dataset belong to the negative class, and the system always classifies an instance as negative. Then the achieved accuracy will be high, but the system will never detect any positive instances. For imbalanced datasets, it is necessary to use performance metrics that consider how good the detector is at finding all ground-truth objects ($FN = 0 \equiv$ high recall) while identifying only relevant objects ($FP = 0 \equiv$ high precision). Therefore, $F1$-score and F_β-score, which includes both the precision and recall metrics, are often used performance metrics for imbalanced datasets.

5.3. Characterisation of a Binary Decision-Making System with Probabilistic Concepts

The detector, i.e. binary decision-making system, can be characterised by four conditional probabilities. Let $t \in \{T, B\}$ denote the correct decision result, and $d \in \{T, B\}$ represent the actual output of the detector. T indicates that the target object is present, while B indicates that it is absent. The following four conditional probabilities can then describe the performance of the decision-making system:

1. Conditional probability of true positives: $\mathsf{Pr}\,(d = T \mid t = T)$
2. Conditional probability of false positives: $\mathsf{Pr}\,(d = T \mid t = B)$
3. Conditional probability of true negatives: $\mathsf{Pr}\,(d = B \mid t = B)$
4. Conditional probability of false negatives: $\mathsf{Pr}\,(d = B \mid t = T)$

Using Bayes theorem, the conditional probabilities can be estimated from the empirical case numbers, where the estimation is represented by $\widehat{\mathsf{Pr}\,(...)}$:

1.
$$\mathsf{Pr}\,(d = T \mid t = T) = \frac{\mathsf{Pr}\,(t = T \mid d = T)\,\mathsf{Pr}\,(d = T)}{\mathsf{Pr}\,(t = T)}$$

[1] To read more about the weights and the use of β^2 see Rijsbergen (1979) and Sasaki (2007)

$$\widehat{\mathsf{Pr}\,(TP)} = \frac{\frac{TP}{FP+TP}\frac{FP+TP}{P+N}}{\frac{P}{P+N}} = \frac{TP}{P} \tag{5.12}$$

2.

$$\mathsf{Pr}\,(d = T \mid t = B) = \frac{\mathsf{Pr}\,(t = B \mid d = T)\,\mathsf{Pr}\,(d = T)}{\mathsf{Pr}\,(t = B)}$$

$$\widehat{\mathsf{Pr}\,(FP)} = \frac{\frac{FP}{FP+TP}\frac{FP+TP}{P+N}}{\frac{N}{P+N}} = \frac{FP}{N} \tag{5.13}$$

3.

$$\mathsf{Pr}\,(d = B \mid t = B) = \frac{\mathsf{Pr}\,(t = B \mid d = B)\,\mathsf{Pr}\,(d = B)}{\mathsf{Pr}\,(t = B)}$$

$$\widehat{\mathsf{Pr}\,(TN)} = \frac{\frac{TN}{TN+FN}\frac{TN+FN}{P+N}}{\frac{N}{P+N}} = \frac{TN}{N} \tag{5.14}$$

4.

$$\mathsf{Pr}\,(d = B \mid t = T) = \frac{\mathsf{Pr}\,(t = T \mid d = B)\,\mathsf{Pr}\,(d = B)}{\mathsf{Pr}\,(t = T)}$$

$$\widehat{\mathsf{Pr}\,(FN)} = \frac{\frac{FN}{TN+FN}\frac{TN+FN}{P+N}}{\frac{P}{P+N}} = \frac{FN}{P} \tag{5.15}$$

These values can be summarised in a probabilistic confusion matrix. The layout of such a matrix is shown in Table 5.2. The estimated conditional probabilities can be summarised in an estimated probabilistic confusion matrix displayed in Table 5.3.

Decision result (d)	Actual values (t)	
	Positive	**Negative**
Positive	$\mathsf{Pr}\,(d = T \mid t = T)$	$\mathsf{Pr}\,(d = T \mid t = B)$
Negative	$\mathsf{Pr}\,(d = B \mid t = T)$	$\mathsf{Pr}\,(d = B \mid t = B)$

Table 5.2.: Probabilistic confusion matrix.

Decision result (d)	Actual values (t)	
	Positive	**Negative**
Positive	$\frac{TP}{P}$	$\frac{FP}{N}$
Negative	$\frac{FN}{P}$	$\frac{TN}{N}$

Table 5.3.: Probabilistic confusion matrix estimated from empirical case numbers.

All entries in the probabilistic confusion matrix are probabilities and are, therefore, within the range of 0 and 1. By the law of total probability, which states that if events $A_1, ..., A_n$ constitute a partition of the sample space S where $\Pr(A_i) \neq 0$ for $i = 1, .., n$, then for any event C of S,

$$\Pr(C) = \sum_{i=1}^{n} \Pr(C \cap A_i) = \sum_{i=1}^{n} \Pr(A_i \cap C), \tag{5.16}$$

and by the conditional probability formula, the following holds,

$$\sum_{i=1}^{n} \Pr(A_i \mid C) = \sum_{i=1}^{n} \frac{\Pr(A_i \cap C)}{\Pr(C)} = \frac{1}{\Pr(C)} \sum_{i=1}^{n} \Pr(A_i \cap C) = \frac{1}{\Pr(C)} \Pr(C) = 1. \tag{5.17}$$

Hence each column in the confusion matrix sums to one,

$$\sum_{i=1}^{2} \Pr(d = A_i \mid t = C) = 1. \tag{5.18}$$

So by collecting one entry in each column in the confusion matrix, the two remaining elements can be derived.

5.4. Characterisation of Decision Systems with Multiple Labels

In the case of decision systems with multiple labels, the detector must identify the type of object present. This leads to a $k \times k$ confusion matrix if k is the number of classes. The row number represents the predicted value, and the column number represents the true value. Element c_{ja} in the confusion matrix is the number of instances belonging to class a but falsely classified as class j. Element c_{aa} in the confusion matrix is the number of true positives for class a (TP_a). The false positive is the sum of all cases that do not belong to class a classified as class a,

$$FP_a = \sum_{i \neq a} c_{ai}, \ i \in 1, ..., k. \tag{5.19}$$

The false negative of class a is the sum of all instances of class a not classified as a,

$$FN_a = \sum_{j \neq a} c_{ja}, \ j \in 1, ..., k. \tag{5.20}$$

The true negatives are the remaining instances,

$$TN_a = P + N - (TP_a + FP_a + FN_a). \tag{5.21}$$

Precision (5.1), Recall (5.2), Accuracy (5.3), Specificity (5.4), Fall-out (5.5), False discovery rate (5.6), False negative rate (5.7), F_1-score (5.9), and F_β-score (5.11) can be calculated for each class separately. To obtain a single score for the system, one can calculate the arithmetic average or weighted average where the weights are the number of true instances in each class. Examplewise, the arithmetic average of precision is,

$$\overline{\mathbf{Precision}} = \frac{\sum_{i=1}^{k} \mathbf{Precision}_i}{k}, \tag{5.22}$$

and the weighted average precision is,

$$\overline{\mathbf{Precision}_{weighted}} = \frac{\sum_{i=1}^{k} \mathbf{Precision}_i \cdot n_i}{\sum_{i=1}^{k} n_i}, \tag{5.23}$$

where n_i is the number of true instances of class i.

A probabilistic confusion matrix with multiple labels is obtained by replacing the number of instances in each cell of the confusion matrix with the probability,

$$\Pr(c_{ij}) = \Pr(d = i \mid t = j), \tag{5.24}$$

where t is the correct classification result and d is the actual output of the detector. These probabilities can be estimated from empirical case numbers by,

$$\Pr(c_{ij}) = \Pr(d = i \mid t = j) \approx \frac{c_{ij}}{\sum_i c_{ij},\ i \in 1, ..., k} \tag{5.25}$$

5.5. Characterisation of Probabilistic Classifications

Probabilistic classifiers output predicted probabilities of an instance belonging to each class instead of a class label. The predicted probabilities for an instance are all between 0 and 1 and sum to one. The probabilities are converted into a single class label to assign a final classification label to an instance. For classification with multiple labels, the instance is typically classified as the class label with the highest probability. For binary classification problems, the predicted probability of a target present (p_T) is compared to a threshold t and defined as positive if $p_T \geq t$. Intuitively the threshold for binary classification is 0.5, but there is no restriction on adjusting the threshold.

Cross-entropy loss can be used to evaluate the predicted probabilities outputted by the system. When the probabilities are converted to class labels, the described metrics in Section 5.2.1 can be used to evaluate the system's performance.

5.5.1. Cross-Entropy Loss

Probabilistic classifiers are often evaluated using the cross-entropy loss, which measures the divergence between the predicted probabilities and the actual class labels. This loss function increases as the predicted probability deviates further from the true label. The cross-entropy loss for one instance i is calculated as,

$$\textbf{Cross-entropy loss}_i = -\sum_{c=1}^{k} y_{i,c} \cdot \log p_{i,c}.$$

Where k is the number of classes, $y_{i,c}$ is a binary indicator (0 or 1) if instance i is of class c, and $p_{i,c}$ is the predicted probability that instance i is of class c.

For binary classification problems, the cross-entropy loss for instance i is,

$$\textbf{Binary cross-entropy loss}_i = -(y \log p_T + (1 - y) \log(1 - p_T),$$

where $y = 1$ if target is present and 0 otherwise, and p_T is the predicted probability of target present.

The total cross-entropy loss for a system is the sum of the cross-entropy loss for all instances,

$$\textbf{Cross-entropy loss}_{tot} = \sum_{\{i\}} \textbf{Cross-entropy loss}_i.$$

5.6. Characterisation of Systems that Produce Axis-Aligned Bounding Boxes

Most visual object detectors produce real-valued vector outputs representing the size and location of a detected target. Typically, this is done by generating an Axis-Aligned Bounding Box (AABB) or an object mask represented by labels on the pixel grid. This section will focus on the characterisation of systems that produce real-valued vector outputs as AABBs and excludes the characterisation of detectors that output object masks.

The previously described concepts, True Positive (TP), False Positive (FP), and False Negative (FN), are used in common metrics that measure the performance of a detector that generates real-valued vector outputs. However, since there are an infinite number of bounding boxes in an image that should not be detected, it is not possible to obtain the number of True Negative (TN). As a result, metrics such as Accuracy (5.3), Fall-out (5.5), and Specificity (5.4) can not be applied. Object detection assessment methods mainly rely on Precision (5.1) and Recall (5.2), and commonly used metrics include F_1-score (5.9) and F_β-score (5.11).

5.6.1. Identifying "Correct" and "Incorrect" Detections

Quantifying the TP, FP, and FN metrics requires defining what is considered a "correct" and "incorrect" detection. The Intersection over Union (IoU) is commonly used to establish this. The IoU calculates the overlapping area between the two regions over the union of the regions,

$$IoU = \frac{\text{Area of overlap}}{\text{Area of Union}}.$$

The IoU is visualised in Figure 5.1. An IoU of 1 indicates perfect overlap between the detected bounding box and the ground truth bounding box, resulting in a perfect detection. Detection can be classified as "correct" or "incorrect" by comparing the IoU to a threshold, t. If the IoU is greater than or equal to t the detection is considered "correct", while if IoU is less than t the detection is considered "incorrect". Choosing the threshold for the IoU is not obvious, and different values for the threshold may lead to different numbers of "correct" and "incorrect" classifications. Therefore the applied IoU threshold must always be stated when evaluating a detector. The IoU is always zero for non-overlapping objects, meaning that the IoU error measure for non-overlapping objects is the same regardless of how big the translational error is.

Figure 5.1.: Visualisation of IoU. The green AABB is the ground truth, and the red AABB is the detection. The vessel portrayed in the image is the autonomous ferry milliAmpere, and the image of the vessel is taken from Skoglund (2018).

The squared Euclidean distance is another method for determining correspondence between

ground truth AABB and detected AABB. This metric measures the Euclidean distance between the centre of the ground truth AABB and the detected AABB. The Euclidean distance between two points, $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ is calculated as follows,

$$d(p_1, p_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}.$$

5.6.2. Confusion Matrix of Systems with Multiple Labels that Produce Real-Valued Vector Outputs

When dealing with systems that have multiple labels and produce real-valued vector outputs, it is necessary to use a confusion matrix of size $(k+1) \times (k+1)$, where k represents the number of classes. The additional row and column of the matrix are defined as the "no target of any class present" category, often referred to as the background. An example of such a confusion matrix is displayed in Table 5.4, where $k = 2$.

The number of false positives in such a confusion matrix is calculated by summing all cases that do not belong to class a but are classified as class a, including instances where the background is mistakenly identified as class a. For instance, the false positives in the example Matrix 5.4 for class "Sailboat" amount to $1 + 2 = 3$. The false negatives for class a encompass the sum of instances belonging to class a that are miss classified as a different class or remain undetected (i.e., classified as background). In the given matrix, the number of false negatives for class "Sailboat" is $4 + 1 = 5$. The number of undetected objects is referred to as the number of drop-outs. In the example, the number of drop-outs for class "Sailboat" is 1. In the context of systems with real-valued vector outputs, the number of true negatives is undefined, resulting in the cell $c_{k+1,k+1}$ of the confusion matrix being empty.

To derive a probabilistic confusion matrix with multiple labels, the counts in each cell of the confusion matrix are replaced with the corresponding probabilities $\mathsf{Pr}(c_{ij}) = \mathsf{Pr}(d = i \mid t = j)$ where t is the correct classification result and d is the actual output of the detector. These probabilities can be estimated from empirical case numbers using Equation (5.25). However, since the number of true negatives is not defined, it is impossible to derive the probability of background being detected as a class from empirical case numbers. Therefore, column $k + 1$ of the probabilistic confusion matrix, representing background as the true class, is undefined. One approach to address this issue is to define the last column as the probability that a false positive is classified as class a, given that it is of type background and is not a true negative ($\mathsf{Pr}(c_{a,k+1}) = \mathsf{Pr}(d = a | d \neq k+1 \wedge t = k+1)$). This can be derived from empirical case numbers by,

$$\mathsf{Pr}(c_{a,k+1}) = \mathsf{Pr}(d = a | d \neq k+1 \wedge t = k+1) \approx \frac{c_{a,k+1}}{\sum_i c_{i,k+1},\ i \in 1, ..., k} \tag{5.26}$$

Conversion of Table 5.4 to a probabilistic confusion matrix with this approach can be viewed in Table 5.5.

		Actual values (t)		
		Sailboat	**Kayak**	**Background**
Detected value (d)	**Sailboat**	15	1	2
	Kayak	4	16	2
	Background	1	3	

Table 5.4.: An example of a confusion matrix of systems with multiple labels that produce AABBs.

		Actual values (t)		
		Sailboat	**Kayak**	**Background**
Detected value (d)	**Sailboat**	$\frac{15}{15+4+1} = 0.75$	$\frac{1}{1+16+3} = 0.05$	$\frac{2}{2+2} = 0.5$
	Kayak	$\frac{4}{15+4+1} = 0.2$	$\frac{16}{1+16+3} = 0.8$	$\frac{2}{2+2} = 0.5$
	Background	$\frac{1}{15+4+1} = 0.05$	$\frac{3}{1+16+3} = 0.15$	

Table 5.5.: An example of an estimated probabilistic confusion matrix estimated from empirical case numbers of systems with multiple labels that produce AABBs. The last column is the probability of which class a detected background belongs to.

5.6.3. Characterising the Positional and Dimensional Uncertainty in Bounding Box Detection

An AABB is characterised by four values; the centre coordinates of the AABB, (x, y), and the width, w, and height, h of the AABB. These values can be compiled into a 4-element vector. The system's uncertainty, or lack of precision, in the placement and dimensions of these bounding boxes, can be characterised by the distribution of the error vector of these output vectors. Describing the actual distribution is usually not possible. However, it is possible to characterise the distribution of the error vector by a covariance matrix and expected values of the error vector elements. This allows for the approximation of the distribution of the error vectors[2].

The error vector represents the difference between the ground truth (gt), i.e., the actual value, and the detected values. The error vector $\vec{e}$ is defined as follows,

$$\vec{e} = \begin{bmatrix} e_x \\ e_y \\ e_w \\ e_h \end{bmatrix}$$

[2]This section is inspired by Rudolf Mester's script on "Statistical Pattern Recognition" (Mester (2022))

Here e_x, e_y, e_w, and e_h are random variables representing an estimated distribution of the error in x, y, w, and h, respectively.

Covariance is a measure of the joint variability of two random variables, X and Y . It is necessary to distinguish between the two cases of random variables, namely discrete-valued and continuous-valued random variables. This distinction is essential because probabilities need to be associated with events. When having a continuous-valued random variable X with $\mathbb{R}$ as the value range, it does not make sense to talk of the probability of X taking a specific value $v \in \mathbb{R}$. This is because the set of real numbers is not countable, thereby taking the sum of all the probabilities would not sum up to 1. Instead of considering the probability that a specific value is taken, as done for discrete-valued random variables, one considers the probability that the continuous-valued random variable X takes a value inside a defined interval. This distinction leads to two different definitions of the expectation value, $\mathsf{E}[X]$.

The expectation value of a discrete-valued random variable X is defined as,

$$\mathsf{E}[X] = \sum_j x_j \cdot \mathsf{Pr}(X = x_j), \tag{5.27}$$

and for a continuous-valued random variable X, it is defined as,

$$\mathsf{E}[X] = \int_{-\infty}^{+\infty} \xi \cdot p_X(\xi)\, d\xi, \tag{5.28}$$

where $\mathsf{Pr}(X = x_j)$ is the probability of X (discrete), taking x_j as value and $p_X(\xi)$ is the probability density function of X (continuous).

The expectation of a random vector, x, is the expected value vector, also known as the mean vector. It represents the expected values of the random variables that make up the random vector. The mean vector is defined as follows:

$$\vec{m}_{\vec{x}} = \mathsf{E}[\vec{x}] = \begin{bmatrix} \mathsf{E}[x_1] \\ \vdots \\ \mathsf{E}[x_n] \end{bmatrix}. \tag{5.29}$$

The variance of a random variable X is defined as the expectation value of the squared deviation of X from its expected value $\mathsf{E}[X]$,

$$\mathsf{Var}[X] = \mathsf{E}\left[(X - \mathsf{E}[X])^2\right]. \tag{5.30}$$

Since the calculation of expected value differs for discrete-valued and continuous-valued random variables, so does the variance. The variance of a discrete-valued random variable X is defined as,

$$\mathsf{Var}[X] = \sum_j (x_j - \mathsf{E}[X])^2 \cdot \mathsf{Pr}(X = x_j), \tag{5.31}$$

and for a continuous-valued random variable X, it is defined as

$$\mathsf{Var}[X] = \int_{-\infty}^{+\infty} (\xi - \mathsf{E}[X])^2 \cdot p_X(\xi)\, d\xi, \tag{5.32}$$

The variance is commonly denoted as σ^2, and its square root, σ, is referred to as the standard deviation. Standard deviation is often easier to interpret than variance since it is expressed in the same units as the original data, while variance is expressed in squared units.

The covariance between two random variables, X and Y, is defined as follows,

$$\mathsf{Cov}\left[X,Y\right] = \mathsf{E}\left[(X - \mathsf{E}\left[X\right])(Y - \mathsf{E}\left[Y\right])\right] = \mathsf{E}\left[X \cdot Y\right] - \mathsf{E}\left[X\right] \cdot \mathsf{E}\left[Y\right] \tag{5.33}$$

A covariance matrix indicates how each value pair of a vector vary together. The covariance matrix can be created by calculating the covariance of each value pair in the error vector. The covariance matrix of the error vector is given by,

$$\mathbf{C}_{\vec{e}} = \begin{bmatrix} \mathsf{Cov}\left[e_x, e_x\right] & \mathsf{Cov}\left[e_x, e_y\right] & \mathsf{Cov}\left[e_x, e_w\right] & \mathsf{Cov}\left[e_x, e_h\right] \\ \mathsf{Cov}\left[e_y, e_x\right] & \mathsf{Cov}\left[e_y, e_y\right] & \mathsf{Cov}\left[e_y, e_w\right] & \mathsf{Cov}\left[e_y, e_h\right] \\ \mathsf{Cov}\left[e_w, e_x\right] & \mathsf{Cov}\left[e_w, e_y\right] & \mathsf{Cov}\left[e_w, e_w\right] & \mathsf{Cov}\left[e_w, e_h\right] \\ \mathsf{Cov}\left[e_h, e_x\right] & \mathsf{Cov}\left[e_h, e_y\right] & \mathsf{Cov}\left[e_h, e_w\right] & \mathsf{Cov}\left[e_h, e_h\right] \end{bmatrix}$$

A covariance matrix $\mathbf{C}_{\vec{x}}$ of vector $\vec{e}$ and each element c_{ij} of the covariance matrix have several properties. First of all, the matrix is square and symmetric. Let x_i and x_k be a pair of vector components of vector $\vec{x}$. If x_i and x_k tend to increase together, then $c_{ik} > 0$. If x_i tends to decrease when x_k increases, then $c_{ik} < 0$. If x_i and x_k are uncorrelated, then $c_{ik} = 0$. It is important to note that the magnitude of the covariance does not provide a direct measure of the strength of the relationship between two random variables. Each element of the covariance matrix can be calculated as follows

$$c_{ii} = \sigma_i^2 = \mathsf{Var}\left[x_i\right] \;\; \textit{and} \; c_{ik} = \rho_{ik}\sigma_i\sigma_k = c_{ki} \tag{5.34}$$

Where ρ_{ik} is the covariance coefficient, σ_i^2 is the variance of the random variable x_i and σ_i is the standard deviation of the random variable x_i. The covariance coefficient is often referred to as the correlation coefficient. However, this thesis will consistently refer to it as the covariance coefficient according to Mester (2022). The covariance coefficient ρ_{ik} is defined as follows,

$$\rho_{i,k} = \frac{\mathsf{Cov}\left[x_i, x_k\right]}{\sigma_i \cdot \sigma_k} \tag{5.35}$$

The covariance coefficient is a standardisation of the covariance and ranges between -1 and 1. This number measures the strength and direction of the linear relationship between two random variables. A value of 1 indicates a perfect positive linear relationship, where an increase in one variable corresponds to an equal increase in the other. A value of -1 indicates a perfect negative linear relationship, where an increase in one variable corresponds to an equal decrease in the other. The magnitude of the covariance coefficient reflects the degree of linearity: values closer to 1 (or -1) indicate a stronger positive (or negative) linear relationship. If x_i and x_k are uncorrelated, then $\rho_{ik} = 0$.

Empirical Estimation from Sample Sets

The described statistics can be derived empirically from a sample set of a given random variable. Let M denote a sample set consisting of n realisations of a random variable X,

$$M = \{x_1, x_2, \ldots, x_n\}$$

A common approach to derive the empirical expected value of a random variable, X, is by the sample mean,

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} x_i. \tag{5.36}$$

Similarly, the empirical variance of X can be derived by the sample variance,

$$S_{XX}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2. \tag{5.37}$$

The empirical standard deviation of X can be derived by the sample standard deviation,

$$S_{XX} = \sqrt{S_{XX}^2}. \tag{5.38}$$

For two random variables X and Y, the empirical covariance can be estimated using the sample covariance,

$$S_{XY}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}). \tag{5.39}$$

Here, x_i represents the individual realisations of the random variable X, y_i represents the individual realisations of the random variable Y, and n is the total number of realisations in the sample. The use of $n-1$ in the sample formulas ensures an unbiased estimate of the variance, standard deviation, and covariance.

The empirical expected value vector, also referred to as empirical mean vector, of the error vector can be obtained by,

$$\hat{\vec{m}}_{e_x} = \begin{bmatrix} \bar{e_x} \\ \bar{e_y} \\ \bar{e_w} \\ \bar{e_h}, \end{bmatrix} \tag{5.40}$$

where $\bar{e_x}$, $\bar{e_y}$, $\bar{e_w}$, and $\bar{e_h}$ are the sample means of e_x, e_y, e_w, and e_h.

The empirical covariance matrix of the error vector can be obtained by,

$$\mathbf{S}_{\vec{e}} = \begin{bmatrix} S_{e_x,e_x} & S_{e_x,e_y} & S_{e_x,e_w} & S_{e_x,e_h} \\ S_{e_y,e_x} & S_{e_y,e_y} & S_{e_y,e_w} & S_{e_y,e_h} \\ S_{e_w,e_x} & S_{e_w,e_y} & S_{e_w,e_w} & S_{e_w,e_h} \\ S_{e_h,e_x} & S_{e_h,e_y} & S_{e_h,e_w} & S_{e_h,e_h} \end{bmatrix} \tag{5.41}$$

Approximation of Error Vector Distribution by Multivariate Normal Distribution

To approximate the distribution of the error vector, it is assumed that each element (i.e., e_x, e_y, e_h, and e_w) can be well approximated by a univariate normal distribution. The distribution of the error vector as a whole can be approximated by a multivariate normal distribution. By employing the multivariate normal distribution, the model considers not only the individual characteristics of each error component but also the interrelationships and dependencies among them. A multivariate normal distribution is a generalisation of the univariate normal distribution to multiple variables, where the joint distribution of the variables is described by a mean vector and a covariance matrix. The multinormal distribution with mean vector $\vec{m} \in \mathbb{R}^p$ and a positive definite $(p \times p)$ covariance matrix $\mathbf{C}$ has the density

$$f(\vec{x}) = (2\pi)^{-\frac{p}{2}} |\mathbf{C}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\vec{x}-\vec{\mu})^T \mathbf{C}^{-1}(\vec{x}-\vec{\mu})\right). \tag{5.42}$$

In the case of the error vector, $p = 4$ since it consists of four elements.

Impact of IOU Threshold on Error Vector Distribution

It should be noted that the IoU threshold plays a critical role in the distribution of the error vectors. The error vectors represent the difference between the ground truth and the detected AABBs and are calculated only for the bounding boxes classified as "correct" detections. Decreasing the IoU threshold allows more error in the centre, height and width of the "correct" AABBs, leading to an increase in the number of "correct" detections with larger error vectors.

5.7. Tuning a Detector

Most detectors can be tuned by some internal parameters. Assuming there exists a single tunable parameter λ. Then the dependency of *pairs* of the metrics for performance measures described in Section 5.2.1 can be visualised as they change when the internal parameter λ is varied.

One such visualisation is the receiver operating characteristic (ROC) curve. The ROC curve illustrates the trade-off between the true positive rate and the false positive rate for a single classifier at different thresholds for classifying an instance as positive. It is important to note that if a detector is tuned to be more sensitive to targets (low probability Pr (d = B | t = T)), the complementary error probability Pr (d = T | t = B) (false positive rate) will typically increase. The ROC curve can be used to find a classification threshold best suited for the specific problem. One can map a model's ROC curve into a single quality number by calculating the Area Under the Curve (AUC). The higher the AUC number, the better the model distinguishes between positive and negative cases.

Figure 5.2 shows an example of a ROC curve. The grey line indicates a random classifier, and the red line indicates a perfect classifier with a true positive rate of 1.0 and a false positive rate of 0.0. The blue and green curves represent two different classifiers. The goal is a classifier that maintains a high true positive rate while also having a low false positive rate. Such a classifier would be close to the red curve in the plot.

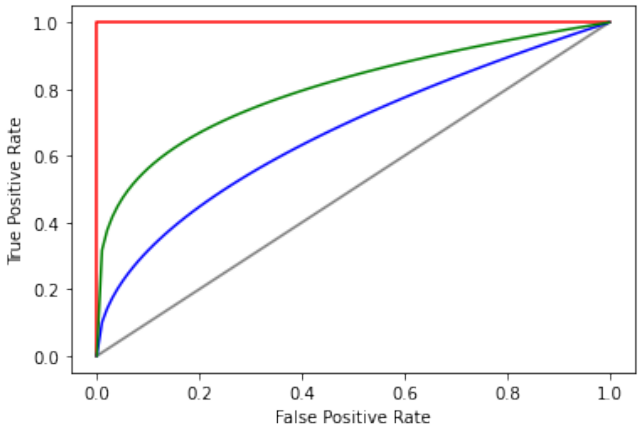


Figure 5.2.: A theoretical receiver operating characteristic (ROC) curve. The curve is only for visualisation and is not generated from real data.

Another visualisation method is the precision-recall curve. The precision-recall curve provides a graphical presentation of the relationship between precision and recall over various thresholds. The recall will increase if the detector is tuned to be more sensitive to the target (low probability $\Pr(d = B \mid t = T)$), and the precision will decrease as the complementary error probability $\Pr(d = T \mid t = B)$ (false positive rate) increases [3].

Figure 5.3 shows an example of a precision-recall curve for a dataset with an equal number of positives and negatives. The grey line indicates a baseline model, where all instances are classified as positives. The red line represents an ideal classifier where positive cases always are classified correctly. The blue and green curves are examples of precision-recall curves for different classification models.

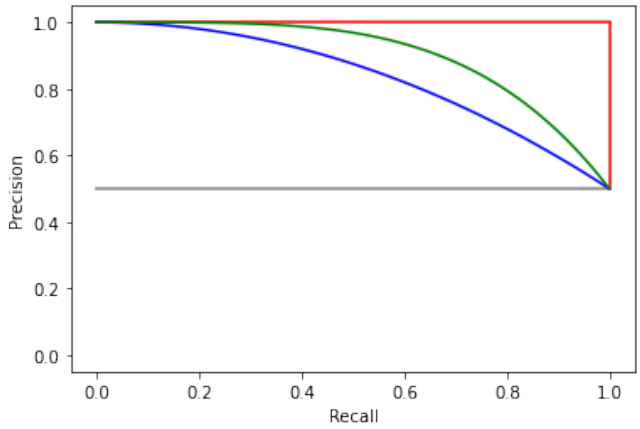


Figure 5.3.: A theoretical precision-recall curve. The curve is only for visualisation and is not generated from real data.

The precision-recall curve can be mapped into a single quality number by calculating the Area Under the Precision-Recall Curve (AUC-PR), where AUC-PR equal to one is ideal. Unfortunately, the precision-recall curve often has a zigzag-like structure, making accurately measuring the AUC-PR difficult. The AUC-PR can instead be estimated by two different approaches: the 11-point interpolation and all-point interpolation.

The 11-point interpolation summarises the precision-recall curve by averaging the maximum precision values at a set of 11 equally spaced recall levels,

$$AP_{11} = \frac{1}{11} \sum_{R \in 0, 0.1, \ldots, 0.9, 1} P_{interp}(R), \tag{5.43}$$

where the P_{interp} is the maximum precision value whose recall value, $\tilde{R}$, is greater than or equal to the current recall value level R,

$$P_{interp}(R) = \max_{\tilde{R}:\tilde{R}>=R} P(\tilde{R}).$$

The all-point interpolation, instead of using 11 points, interpolates through all points. It is defined as follows,

[3] The section about the precision-recall curve and AP is inspired by Padilla et al. (2020)

$$AP_{all} = \sum_{n} (R_{n+1} - R_n) P_{interp}(R_{n+1}), \tag{5.44}$$

where the P_{interp} is the maximum precision value whose recall value, $\tilde{R}$, is greater than or equal to R_{n+1},

$$P_{interp}(R) = \max_{\tilde{R}:\tilde{R}>=R_{n+1}} P(\tilde{R}). \tag{5.45}$$

To measure the accuracy of object detection over multiple classes, the mean AP (mAP) is often used. The mAP takes the average AP over all classes,

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i, \tag{5.46}$$

where AP_i is the AP of the ith class and N is the total number of classes.

The ROC curve cannot be utilised for object detectors that produce real-valued vector outputs since the number of true negatives can not be measured. It is instead common to employ the precision-recall curve, AUC-PR, AP_{11} (5.43) and AP_{all} (5.44) and mAP (5.46).

6. Characterising Object Detector Performance under Varying Conditions

In real-world deployments, object detectors face numerous challenges that can complicate the task of detecting objects, resulting in performance fluctuations. This chapter explores theoretical concepts that can be utilised to model these variations in performance for visual object detectors. Further, it examines how different detection conditions affect the detector's performance by analysing relevant literature on detector performance in adverse conditions.

6.1. Temporal Model of Detector Performance

The quality state of an object detector is contingent upon the conditions under which detection is performed. For instance, object detection of marine vessels is easier on a clear day than on a foggy evening. The performance of an object detector can be characterised by a probabilistic confusion matrix and the distribution of the bounding box error vector. Since detection conditions vary over time, these performance metrics will also vary over time. Therefore, developing a temporal model of detector performance with a finite set of states representing various detection conditions, such as excellent, good, poor, and terrible, and a corresponding confusion matrix for each state is necessary. Such a model can be constructed using a stochastic or probabilistic automaton.

A stochastic automaton is a mathematical model for a process that generates random sequences of symbols. It is characterised by a set of discrete states, $s_i,\ i = 1..n$, and a stochastic transition matrix A. The transition matrix A consists of conditional probabilities describing the probability of moving from state s_j to state s_i at time k. Assume that the transition matrix A does not depend on the time index k, hence the transition process is stationary. The stochastic automata generalise the concepts of a Markov chain and thus satisfy the Markov property, meaning that the probability of moving to a state at time k is solely based on the state at time $k-1$.

Transition Matrix

Let there be n possible states, the transition matrix will then be a $n \times n$ matrix, such that entry a_{ij} is the probability of the automaton being in state s_i at time k given that the automaton is in state s_j at time $k-1$,

$$a_{ij} = \mathsf{Pr}\left(s(k) = s_i \mid s(k-1) = s_j\right). \tag{6.1}$$

The matrix $\mathbf{A}$ is then defined as follows,

$$\mathbf{A} = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix}. \tag{6.2}$$

The stochastic transition matrix obeys the general rules of probabilities, where for events $e_i = e_1, e_2, ..., e_n$ the following holds,

$$\sum_{\{e_i\}} \Pr(e_i) = 1 \tag{6.3}$$

and thus also

$$\sum_{\{e_i\}} \Pr(e_i \mid c_j) = 1 \tag{6.4}$$

where c_j is an arbitrary condition. This means that at any given time k, the automaton can be in exactly one of the finite number of states,

$$\sum_{i=1}^{n} \Pr(s(k) = s_i) = 1 \tag{6.5}$$

and thus that each column in the transition matrix sum to one,

$$\sum_{i=1}^{n} \Pr(s(k) = s_i \mid s(k-1) = s_j) = 1, \forall j \in [1...n]. \tag{6.6}$$

State Probability Vector

Let $\vec{p}$ be a state probability vector and the sequence $\{\vec{p}[k]\} : \vec{p}[1], \vec{p}[2], ..., \vec{p}[k]$ be the sequence of the state probability vectors from time $t = 1$ to time $t = k$. The elements $p_i[k]$ of a state probability vector $\vec{p}[k]$ gives the probability that the automaton is in state s_i at time $t = k$,

$$p_i[k] = \Pr(s[k] = s_i) \tag{6.7}$$

Consider an example with three states $n = 3$ and the following transition matrix $\mathbf{A}$,

$$\mathbf{A} = \begin{bmatrix} \frac{1}{3} & \frac{1}{6} & \frac{1}{20} \\ \frac{1}{3} & \frac{2}{3} & \frac{1}{20} \\ \frac{1}{3} & \frac{1}{6} & \frac{9}{10} \end{bmatrix}. \tag{6.8}$$

Let it be known that the automaton was in state 2 at time $k = 1$, $s[1] = s_2$. Then the state probability vector is defined as follows,

$$\vec{p}[1] = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}. \tag{6.9}$$

The state probability vector at time $k+1$, $\vec{p}[2]$, is calculated by multiplying the transition matrix $\mathbf{A}$ with the state vector $p[1]$,

$$\vec{p}[2] = \mathbf{A} \cdot \vec{p}[1] = \begin{bmatrix} \frac{1}{6} \\ \frac{2}{3} \\ \frac{1}{6} \end{bmatrix}. \tag{6.10}$$

If this process is performed recursively to determine the state vector at time $k = 5$, $\vec{p}[5]$, the resulting equation is as follows:

$$\vec{p}[5] = \mathbf{A} \cdot \vec{p}[4] = \mathbf{A}^2 \cdot \vec{p}[3] = \mathbf{A}^3 \vec{p}[2] = \mathbf{A}^4 \cdot \vec{p}[1]. \tag{6.11}$$

This can be generalised to find a state vector $\vec{p}[k]$ at time k when a state vector $\vec{p}[m]$ is given where $k > m$,

$$\vec{p}[k] = \mathbf{A}^{k-m} \cdot \vec{p}[m]. \tag{6.12}$$

If the square $n \times n$ matrix $\mathbf{A}$ has n distinct eigenvectors, it is diagonalisable, simplifying the equation above. An eigenvector of matrix $\mathbf{A}$ is a nonzero vector, v, that satisfies the following equation,

$$\mathbf{A} \cdot \vec{v} = \lambda \vec{v}, \tag{6.13}$$

for some scalar λ, known as the the eigenvalue for $\vec{v}$. By defining P as the matrix whose columns are the eigenvectors $\vec{v}_1, ..., \vec{v}_n$ and D is the diagonal matrix whose diagonal entries are the corresponding eigenvalues $\lambda_1, ..., \lambda_n$. The diagonalisable matrix $\mathbf{A}$ can be written as,

$$\mathbf{A} = PDP^{-1}. \tag{6.14}$$

The k-th power of the diagonalisable matrix $\mathbf{A}$ can then easily be computed,

$$\mathbf{A}^k = PD^kP^{-1}, \tag{6.15}$$

and the state vector $\vec{p}[k]$ at time k when a state vector $\vec{p}[m]$ is given where $k > m$ can be calculated as follows,

$$\vec{p}[k] = \mathbf{A}^{k-m} \cdot \vec{p}[m] = PD^{k-m}P^{-1} \cdot \vec{p}[m]. \tag{6.16}$$

Steady State Vector

Suppose that the Markov chain is irreducible, meaning that every state can be reached from every other state in finite time,

$$\forall s_i, s_j \in S, \exists m < \infty : Pr(s(k+m) = s_i | s(k) = s_j) > 0. \tag{6.17}$$

Additionally, assume every state in the state space is aperiodic, meaning there are no fixed periods for returning to a state. Then the probability vector $\vec{p}[k]$ converges towards a steady state vector as $k \to \infty$,

$$\lim_{k \to \infty} \mathbf{A} \cdot \vec{p}[k] = \vec{p}[k]. \tag{6.18}$$

Meaning that the steady state vector $\vec{x}$ is the vector that satisfies the following equation,

$$\mathbf{A} \cdot \vec{x} = \vec{x}. \tag{6.19}$$

Hence the steady state vector is the eigenvector of matrix $\mathbf{A}$ with eigenvalue λ equal to 1.

6.2. Calculating the Transition Matrix from Observation

Assuming that only the observed states are known and the goal is to calculate the transition matrix $\mathbf{A}$, the sequence of observed states can be denoted by $S : s[1], ..., s[n+1], ..., s[L]$. Let O_{ij} represent the number of transitions from state i to state j. The probability of transitioning from state j to state i, a_{ij}, can then be defined as follows:

$$a_{ij} = \frac{O_{ij}}{L} \tag{6.20}$$

Further, it is assumed that there is at least a marginal probability for each transition to occur. To account for transitions that are never observed during the sequence of states, one observation for each transition is fabricated,

$$a_{ij} = \frac{O_{ij} + 1}{L + 1}. \tag{6.21}$$

6.3. Transition Matrix for Object Detection in Dynamic Environments

When considering the performance of an object detector in dynamic environments with changing conditions, it is generally expected that the probability of staying in a particular state is higher than the probability of transitioning to a new state. This expectation can be exemplified by considering the detection of vessels in marine situations where weather conditions alternate. Considering a situation where a cloud appears and partly occludes a vessel, leading to a "bad state" in terms of detector performance. It is reasonable to assume that, in the subsequent time frame, the occluding cloud is likely to persist and continue to obstruct the visibility of the vessel. This assumption arises from the understanding that weather conditions tend to change gradually rather than abruptly. The transition matrix of the stochastic automaton can reflect this by assigning higher probabilities to stay in the same state and lower probabilities to transition to a new state. Consequently, the diagonal elements of the transition matrix should be close to 1, while the remaining off-diagonal elements should be close to 0.

Assume three different weather condition states, namely 0, 1, and 2, where 0 is excellent weather, 1 is average weather, and 3 is poor weather. Considering these three states and that the probability of staying in the same state is higher than changing state for the next timestep, one can define a transition matrix $\mathbf{B}$. Note that these numbers are chosen arbitrarily while keeping the diagonal numbers close to 1 and ensuring that the columns add up to 1.

$$\mathbf{B} = \begin{bmatrix} 0.97 & 0.01 & 0.01 \\ 0.01 & 0.98 & 0.04 \\ 0.02 & 0.01 & 0.95 \end{bmatrix} \tag{6.22}$$

The stochastic automaton for these three states with the transition matrix $\mathbf{B}$ can be viewed in Figure 6.1.

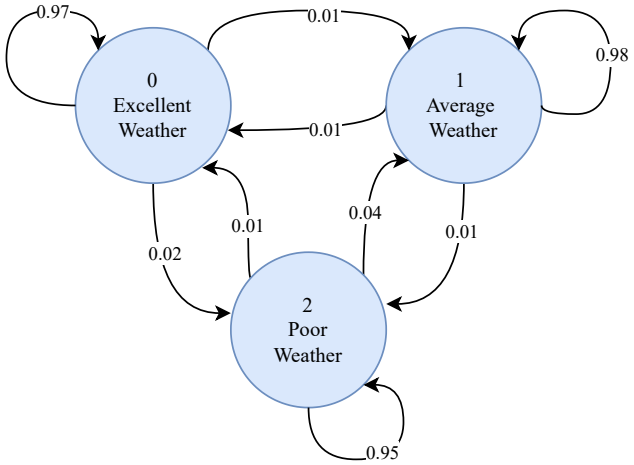


Figure 6.1.: A stochastic automaton with states 0, 1, and 2, where 0 is excellent weather, 1 is average weather, and 3 is poor weather using transition matrix $\mathbf{B}$.

Using this transition matrix $\mathbf{B}$ to generate a state sequence of length 100 results in the graph shown in Figure 6.2. One can see that the stochastic automaton does not change state very often. Now, let the next state be independent of the current state. Then the probability of moving to another state is the same regardless of which state one is currently in. In this case, all the columns in the transition matrix are identical. Let matrix $\mathbf{C}$ be a transition matrix where the next state is independent of the current state. These probabilities are also arbitrarily chosen while satisfying the properties of a transition matrix.

$$\mathbf{C} = \begin{bmatrix} \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{2}{3} & \frac{2}{3} & \frac{2}{3} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \end{bmatrix} \tag{6.23}$$

Figure 6.3 shows the result after generating a state sequence of length 100 using the transition matrix $\mathbf{C}$. Compared to Figure 6.2, where the next state is dependent on the current state, one can see that the stochastic automaton alternates between the different states more frequently. Further, it stays more often in state 1 (average weather), independently of the current state, as this is the most likely state using transition matrix $\mathbf{C}$.

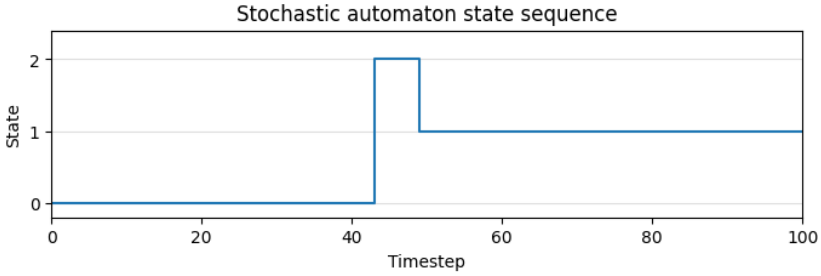


Figure 6.2.: A stochastic automaton state sequence for 100 timesteps where the next state depends on the current state.

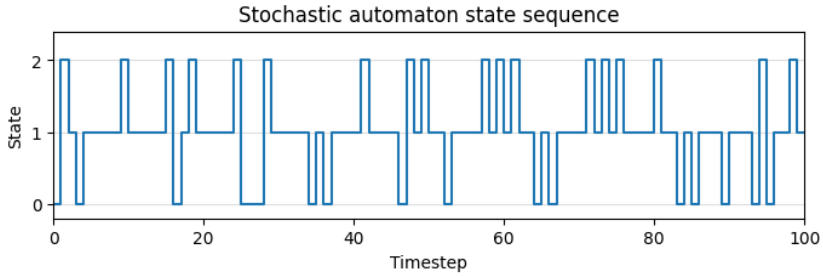


Figure 6.3.: A stochastic automaton state sequence for 100 timesteps where the next state is independent of the current state.

6.4. Influence of Weather and Light Conditions on Visual Detector Performance

Adverse weather conditions can significantly degrade the performance of visual object detection. Various factors, such as fog, rain, snow, dust, and low light conditions, substantially impact image quality (Al-Haija et al. (2022)).

One important aspect affecting the performance of visual object detectors is the amount and quality of light in the environment. During daylight hours, there is ample natural light, enhancing the visibility of objects. In contrast, low-light, nighttime, dusk or dawn conditions can pose significant challenges for object detectors due to the low illumination levels. However, even in optimal daylight conditions, excessively bright or harsh lighting can also pose challenges for object detectors.

Weather conditions exhibit variations due to different types, sizes, and amounts of particles in the atmosphere. Air molecules are small in size with reference to the wavelength of light, giving minimal atomic scattering. On *clear day*, the atomic scattering is approximately equal to pure air scattering, and visibility is optimal. Larger particles, such as water molecules, produce more atomic scattering and can thereby cause glare, distortion, and reduced visibility (Narasimhan and Nayar (2002)). As a result, adverse weather conditions such as snow, rain, and fog can significantly degrade the performance of visual object detection.

The presence of fog, consisting of microscopic water droplets in the air, introduces scattering and attenuation of light, leading to reduced visibility and degraded image quality. The scattering of light causes objects to appear hazy and indistinct, making it difficult for detectors to extract meaningful features and distinguish between different classes (Rothmeier and Huber (2021)). Additionally, fog-induced attenuation diminishes the contrast between objects and their background, further complicating the detection process (Zang et al. (2019)).

Furthermore, many algorithms used in outdoor vision systems operate assuming that pixel intensities in images are directly proportional to scene brightness. However, dynamic weather conditions, such as rain, snow, and hail, introduce sharp fluctuations in intensity in images and videos, thereby degrading the quality of images and videos and, thus, violating this basic assumption (Zang et al. (2019)). For instance, raindrops create patterns that decrease image intensity and blur underlying patterns, while heavy snow and hail increase image intensity and obscure object patterns, making them unrecognisable. Additionally, raindrops, hailstones, and snowflakes can also directly impact the physical camera, potentially leading to issues such as frost formation on the lens.

As adverse weather conditions are a common problem for object detection, researchers and engineers are actively developing weather-specific object detection algorithms and techniques. Multiple approaches exist, and these methods are often combined and tailored based on the specific weather condition. Some commonly used approaches, along with some example research, are the following:

- **Weather-specific datasets**: Creating specialised datasets focusing on adverse weather conditions can help train object detectors to be more robust in challenging conditions. These datasets include images or videos captured in adverse weather conditions, such as rain, fog, and snow. For instance, Kenk and Hassaballah (2020) made the DAWN dataset of real-world images collected under various adverse weather conditions in a diverse traffic environment. Sindagi et al. (2019b) introduced a new large-scale crowd-counting dataset (JHU-CROWD) which includes several images with weather degradation.
- **Data augmentation**: Augmenting the training data with synthetic weather conditions can enhance the robustness of object detectors. Volk et al. (2019) presented an approach to enhance object detection algorithms by augmenting training data with synthetic rain variations. Another study by Bernuth et al. (2019) focused on reusing established and labelled datasets by augmenting them with adverse weather effects such as snow and fog.
- **Sensor fusion**: Combining data from multiple sensors can enhance object detection in adverse weather conditions. For instance, valuable complementary information can be obtained by integrating visual data with data from thermal or radar sensors. Chaturvedi et al. (2022) proposed a GLA framework to tackle object detection in adverse weather by performing adaptive sensor fusion of camera, gated camera, and lidar sensing streams at two fusion stages.
- **Transfer learning and domain adaptation**: Leveraging pre-trained models on large-scale datasets and fine-tuning them on smaller weather-specific datasets can improve object detection performance. Transfer learning allows models to benefit from the knowledge learned in a different yet related domain. Sindagi et al. (2019a) proposed an unsupervised prior-based domain adversarial object detection framework specifically designed to adapt detectors to hazy and rainy conditions. Hnewa and Radha (2021) introduced a novel MultiScale Domain Adaptive YOLO (MS-DAYOLO) framework that incorporated multiple domain adaptation paths and corresponding domain classifiers at different scales of the YOLOv4 object detector to generate domain-invariant features. Their experiments demonstrated significant improvements when training YOLOv4 using MS-DAYOLO and testing on target data representing challenging weather conditions in the context of autonomous driving applications. Chen et al. (2018) assumed that adverse weather conditions lead to domain shift and proposed a domain adaptive Faster-RCNN approach that addresses domain shift at both the image-level and instance-level.
- **Dehazing and rain removal**: Pre-processing techniques such as dehazing or rain removal can be utilised to improve image visibility. These techniques are designed to mitigate the effects of haze, fog, or rain, resulting in clearer images that facilitate more accurate object detection. Fattal (2008), He et al. (2009), and Zhang and Patel (2018a) introduced dehazing methods, while Zhang et al. (2017) and Zhang and Patel (2018b) introduced rain removal methods.
- **Adaptive algorithms**: Object detection performance can be improved by developing adaptive algorithms that adjust their parameters or strategies based on weather conditions. Liu et al. (2022) proposed a novel Image-Adaptive YOLO (IA-YOLO) framework where each image can be adaptively enhanced. They incorporated a differentiable image

processing (DIP) module that accounts for adverse weather conditions within the YOLO detector. The DIP module's parameters were predicted using a small convolutional neural network (CNN-PP). The experimental results demonstrated the effectiveness of the proposed IA-YOLO method in improving object detection performance in foggy and low-light scenarios.

Quantifying the performance of object detectors in adverse weather conditions is challenging due to the inherent differences among detectors. Variations in detector architectures, training strategies, feature representation, hyperparameter settings, and weather-specific optimisation make it challenging to establish a universal framework for performance evaluation. Each detector may have unique strengths and weaknesses, and their performance in adverse weather conditions can vary.

7. A Proposed System for Object Detection Simulation for Autonomous Surface Vessel Certification

This chapter presents the proposed system for simulating a marine visual object detector for autonomous vessels that aims to fulfil the system requirements outlined in Section 1.3. The subsequent sections provide a detailed description and justification of the different design choices and explain how to utilise the MODSIM system.

7.1. Overview of the Marine Object Detector Simulator (MODSIM)

This section presents an overview of the complete MODSIM system, outlining the main components and the system pipeline. The overall structure of the MODSIM system is visualised in Figure 7.1, illustrating its components and their interconnections.

The first component of the system is the dynamic scene generator, which is a simple representation of the real world that describes the scene in which detections are to be simulated. The marine surface is simplified to be 2D, and the marine vessels are represented by 3D shoe boxes surrounding the original shape. The dynamic scene contains information about the extent of the vessel, the classification of the vessel and the pose (i.e. positions and orientations) of each vessel for a sequence of timesteps. The pose sequences can be generated directly in the dynamic scene generator or by replaying previously recorded pose sequences. The dynamic scene generator is presented in more detail in Section 7.2.

The defined scene can be viewed from various angles by incorporating one or more virtual cameras. To achieve this, a camera setup needs to be defined, specifying the number of cameras, their position and orientation, and the internal parameters of each camera. A camera can be stationary or mounted on a vessel, i.e. moving along the track of the vessel it is mounted on. Section 7.3 elaborates on the camera setup. When the camera is placed on board instead of onshore, tracking becomes more challenging because the objects in the image will move around due to wave motion. To reflect the motion caused by waves, a varying amount of deviation is added to the defined orientation of the vessel. This is only included on the vessel that the camera is mounted on. The introduced wave motion is further described in Section 7.4.

The next part of the system is the detector simulator, which simulates detections of the marine vessels visible from each virtual camera perspective. First, the 3D corner points of each vessel are projected onto a 2D image using a mathematical camera model. Then, the 2D bounding boxes are computed, and the vessels are classified with the labels defined in the dynamic scene, producing the annotations of the scene. This process is further described in Section 7.5. Following this, the error generator distorts the annotations to simulate errors that may occur for visual object detectors. These errors involve variations in the bounding boxes' centre, height, and width, varying confidence scores for classification, occasional miss-classifications of vessels, and generating false

detections and drop-outs (missed detections). The amount of error varies over time according to a temporal model that reflects changes in detector performance. The error generator is quantified by inputting error statistics for the detector that is to be simulated. These distorted detections are the final output of the simulator. Section 7.6 elaborates on the error generator.

The system repeats the process for each timestep, allowing for simultaneous input of new pose data and generation of simulated detections. For the detections to be streamed into a tracker, they need to be converted into the format required by the tracker. Optionally, the output of each component can be visualised, but this visualisation can only be produced after the simulation.

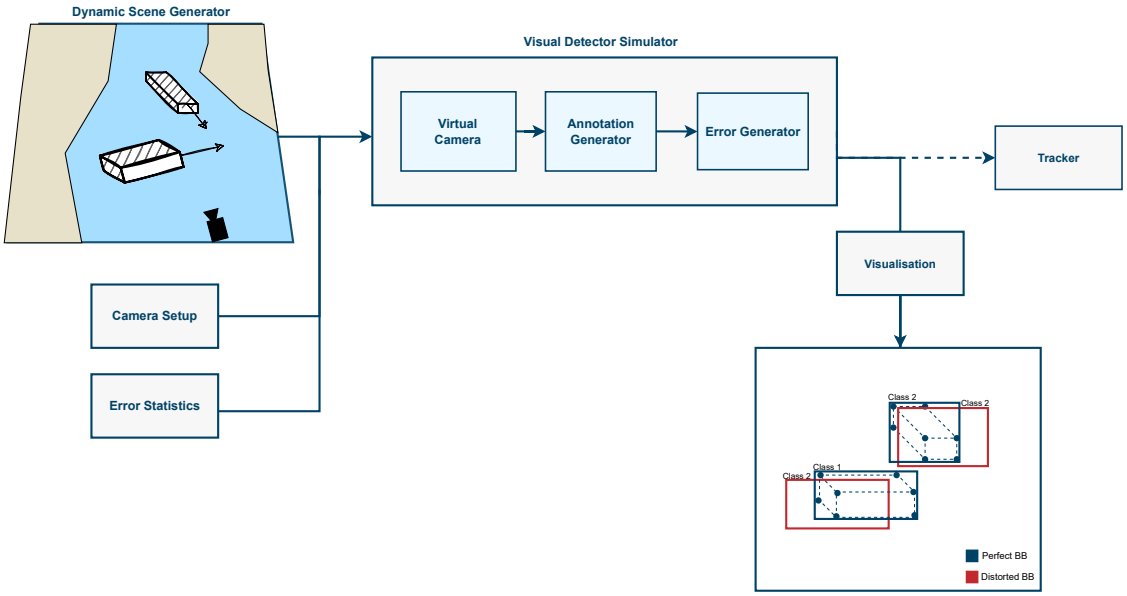


Figure 7.1.: A block diagram illustrating the structure of the MODSIM system.

7.2. Dynamic Scene Generator

This section describes the structure and functionality of the dynamic scene generator in the MODSIM system. The dynamic scene generator is responsible for representing the scene where detections will be simulated. It should be noted that creating diverse scenes is not the primary objective of this project. Consequently, the MODSIM system must be designed to be compatible with various pose sequence generators. In Subsection 7.2.1, the scene information required by MODSIM is presented. Subsection 7.2.2 explains how to express the corner points of the vessels in the WCF. Additionally, Subsection 7.2.3 and Subsection 7.2.4 provide details about the pose sequences utilised in this thesis.

7.2.1. Information Represented in the Dynamic Scene Generator

The dynamic scene consists of one or several vessels, which are simplified to be 3D shoe boxes. The user must input each vessel's air draft, beam, and length to represent the shoe boxes. The air draft refers to the height of the vessel above the waterline, while the beam is the vessel's width, and the length represents the vessel's overall length. Additionally, the user must input the label (correct classification) for each vessel. These vessel characteristics remain constant throughout the simulation. To establish a connection with a pose sequence, each vessel characteristic requires

an unique ID. The vessel characteristics can be inputted using a JSON file with the structure displayed in Figure 7.2.

Further, the pose sequence of each vessel must be defined. The pose sequences includes the centre position, orientation of the vessel, and the time stamp of the pose. There are two methods for defining pose sequences, namely generating it directly in the MODSIM system using a motion model or reading from a pose file. The information that must be represented in the dynamic scene generator is summarised in Table 7.1 and Table 7.2.

```
{
    "<vesselID>": {
        "air_draft_m": <air_draft_m>
        "beam_m": <beam_m>
        "length_m": <length_m>
        "label": "<label>"
    },
    ...
}
```

Figure 7.2.: Required structure of the JSON file containing vessel characteristics.

Parameters	**Description**	**Unit**
Air draft	Height of the vessel above the waterline	Metre (m)
Beam	The vessel's width	Metre (m)
Length	The vessel's length	Metre (m)
VesselID	Unique ID for the vessel characteristics	

Table 7.1.: Required information for the vessel characteristics in the dynamic scene generator in the MODSIM system.

Parameters	**Description**	**Unit**
VesselID	ID connecting the pose sequence to a vessel characteristic	
Timestamp	A timestamp for each registered position	Seconds (s)
Centre position	The vessel's centre position in WCF at the current timestamp	Metre (m)
Heading	The vessel's heading vector at the current timestamp	Radians (rad)

Table 7.2.: Required information for the pose sequences in the dynamic scene generator in the MODSIM system.

Generate Pose Sequences Directly Using a Motion Model

The first supported method for obtaining pose sequences in the MODSIM system is to generate them directly during simulation by using a motion model. The motion model must be able to generate positions and orientations for each vessel at every time step of the simulation. During

this project, a simplified motion model has been defined, namely the circular motion scene model. Section 7.2.3 describes this scene model further.

Read Pose Sequences from a Pose File

The second supported method for obtaining pose sequences in the MODSIM system is reading from pose files. These pose files contain pose sequences that can be generated in either a real or synthetic environment and can be pre-recorded or generated continuously during the simulation. The MODSIM system reads one time-step at a time, allowing for simultaneous generation of the pose files. For the pose files to be compatible with the system, they must include the time stamp, vessel ID, centre position in metres, and the heading vector in radians. The vessel ID is necessary to connect the pose sequence to a specific vessel characteristic. The supported format for the pose files in MODSIM is JSON, Figure 7.3 displays the file structure. If using externally generated pose sequences, they must follow this file structure to be compatible with the system. Using pose files is a flexible and convenient method for obtaining pose sequences in MODSIM, as it allows for the integration of pose sequences generated by an external part. Section 7.2.4 elaborates on pose sequences obtained from an external part in this project.

```
{
    "<timestamp>": {
        "<vesselID>": {
            "center_position_m": [
                <w1>,
                <w2>,
                <w3>
            ],
            "heading_rad": <alpha>
        },
        ...
    },
    ...
}
```

Figure 7.3.: Required structure of the JSON file containing pose sequences.

7.2.2. Coordinate Transformation from nVCF to WCF

The pose sequences provide the vessel's centre position and heading vector on the 2D marine surface described in WCF. The eight 3D corner points of the rectangle enclosing the ship can easily be expressed by nVCF coordinates (c_i^{nV} for $i \in 1, \ldots, 8$) using the given vessel characteristic. To project the corner points onto the image plane, they must first be converted to WCF coordinates through rotation and translation. To achieve this, it is necessary to determine the rotation matrix $\mathbf{R}_W^{nV}$ and the translation vector $\vec{t}_W^{nV}$ such that the coordinates of a corner point c_i^{nV} expressed in nVCF are related to the equivalent point c_i^W in WCF by the following equation:

$$c_i^W = \mathbf{R}_W^{nV} c_i^{nV} + \vec{t}_W^{nV} \tag{7.1}$$

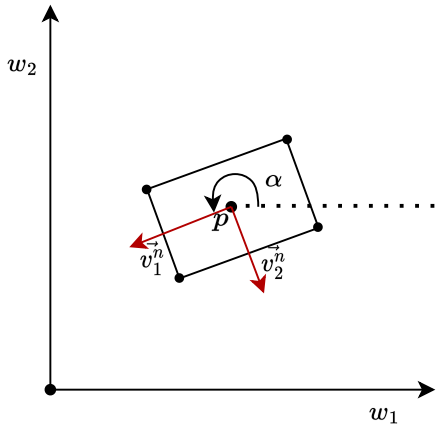


Figure 7.4.: The relation between the orientation of the Nominal Vehicle Coordinate Frame (nVCF) and the World Coordinate Frame (WCF) for a nominal, smooth, and deterministic path.

The nVCF is defined such that the third unit vectors of WCF and nVCF are always parallel ($\vec{w}_3 \parallel \vec{v}_3^n$). Therefore, the vehicle's orientation is defined only by the rotation about the $\vec{w}_3$ axis in the WCF by an angle α, shown in Figure 7.4. This gives the following rotation matrix,

$$\mathbf{R}_W^{nV} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7.2}$$

The vehicle's heading vector, $\vec{d}$, is parallel to the first unit vector of the nVCF, $\vec{v}_1^n$ ($\vec{d} \parallel \vec{v}_1^n$). The angle α is therefore found by using the dot product between the heading vector of the vehicle and the first unit vector of the WCF, $\vec{w}_1$, and solving for α,

$$\alpha = \cos^{-1}\left(\frac{\vec{d}\cdot\vec{w}_1}{|\vec{d}|\cdot|\vec{w}_1|}\right). \tag{7.3}$$

To find the translation vector between the nVCF and WCF, the utilisation of the known position of the ship's centre expressed in WCF, denoted as p, is employed. The translation vector is equal to the position vector for point p expressed in WCF,

$$\vec{t}_W^{nV} = \vec{p}^W. \tag{7.4}$$

7.2.3. Generate Pose Sequences Using a Circular Motion Model

A circular motion model have been defined in order to generate pose sequences directly in the dynamic scene generator. Despite the resulting pose data not representing an interesting and realistic scene, its ease and speed of implementation make it valuable for acquiring pose data in the early stages of the project. Further, generating pose sequences is not the main focus of this master's thesis, which reasons for such a simplistic approach.

Assume there are N ships, represented as a rectangular box (shoe box), circulating with some given speed on concentric circles on a planar ground. All circles have a different radius, i.e. there

are no collisions. The scene is observed from a camera almost on the planar ground or a few metres above. This scene is illustrated in Figure 7.5. From the camera's perspective, it appears as though several ships are moving back and forth, sometimes covering each other. The motion of ships can be described by a coordinated turn model.

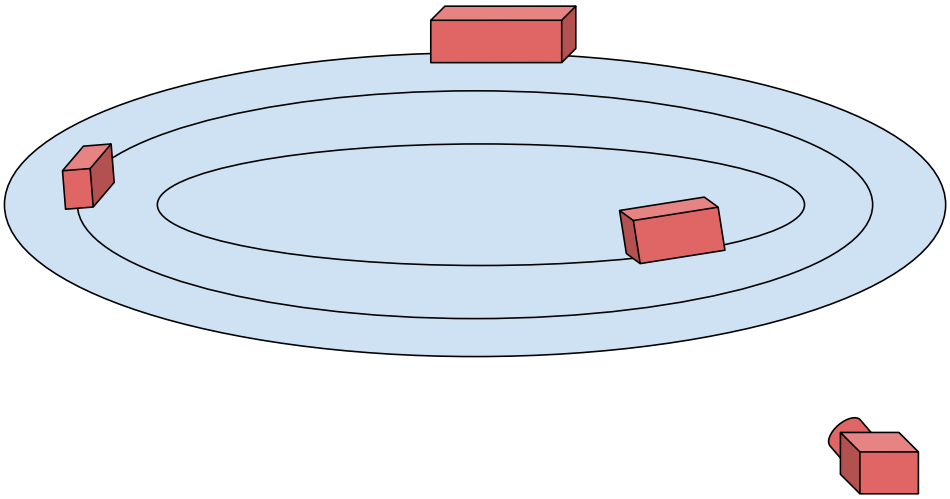


Figure 7.5.: Illustration of the dynamic scene model with generated pose sequence using a circular motion model. The three shoe boxes represent three vessels. In the right corner, one camera is placed.

Coordinated Turn Model

The coordinated turn model is a motion model where the heading direction of a vehicle is always identical to the momentary velocity vector. Hence, the rotation of the vehicle's motion vector and the vehicle's heading direction are synchronised. Most regular vehicles, including ships, cars, and aeroplanes, follow this motion path. Some vehicles that do not move accordingly to this motion model are helicopter drones, rafts, and hot air balloons. A circular path is the simplest case of motion generated by a coordinated turn model. Then the heading vector of the vehicle is always tangential to the circular motion path.[1]

Time-Discrete Motion Equation for a Circular Path

Assume the centre p of the ship moves along a circular path, as shown in Figure 7.6. Given a rotation rate ω [rad/sec], the time-discrete motion equation for the point p^W expressed in the WCF can be defined. ΔT is the time difference between the discrete time steps, n is the number of time steps, R is the circle's radius, and $\vec{p}_0^W$ is the position vector of the circle's centre. Let $\phi = \omega n \Delta T$, then the motion of the point p^W expressed in the WCF can be described as follows,

$$\begin{bmatrix} p_1^W(n) \\ p_2^W(n) \end{bmatrix} = \vec{p}_0^W + R \begin{bmatrix} \cos\phi \\ \sin\phi \end{bmatrix}. \tag{7.5}$$

As ships follow the coordinated turn model, the heading vector $\vec{d}$ of the ship will always be identical to the momentary velocity vector, which is always perpendicular to the position vector

[1] This paragraph is inspired by notes from our supervisor Rudolf Mester.

$\vec{r}$ from the centre of the circle to the position of the centre of the ship p. $\vec{r}$ can be defined as

$$\vec{r} = R \begin{bmatrix} \cos\phi \\ \sin\phi \end{bmatrix}. \tag{7.6}$$

By normalising and rotating $\vec{r}$ counter-clockwise 90° the time-discrete motion equation of the heading vector $\vec{d}$ expressed in WCF is obtained,

$$\vec{d}^W = \begin{bmatrix} \cos(\phi + \pi/2) \\ \sin(\phi + \pi/2) \end{bmatrix} = \begin{bmatrix} -\sin\phi \\ \cos\phi \end{bmatrix}. \tag{7.7}$$

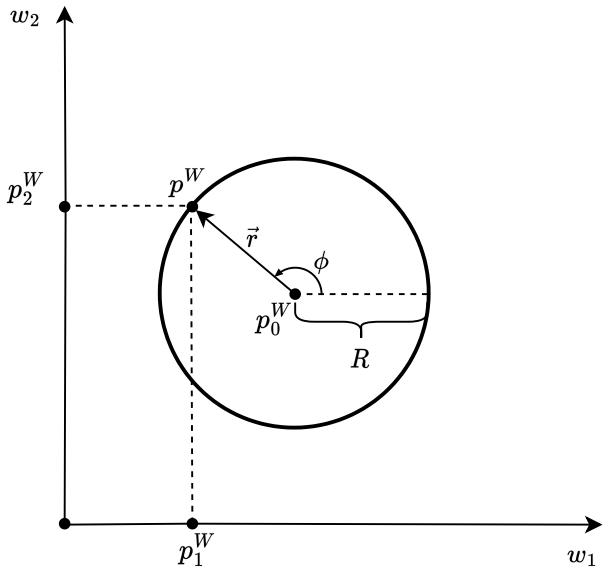


Figure 7.6.: Illustration of the circular motion of a ship where p is the ship's centre.

7.2.4. Read Pose Sequences from a Synthetic Environment Created by Henrik Fjellheim

In this thesis, another pose sequence was incorporated, which was generated by Henrik Fjellheim, a master's student at NTNU. The utilisation of his pose files showcase the potential of simulating a detector in a synthetic environment generated by an external source. Henrik Fjellheim has researched short-term trajectory planning for a non-holonomic robot vehicle. His approach involves combining reinforcement learning with a predefined vehicle model. In the initial stages of the project, a reactive agent was developed to evaluate the robot platform's vehicle model and sensory equipment. This reactive agent operates within a multi-agent environment, utilising various sensory data to avoid collisions and maintain a pre-defined speed. After testing the platform, the agent was only used as a dynamic obstacle for training the actual reinforcement learning agents. Since the reactive agents behave mostly like cars, the pose files produced by running the reactive agents in a closed-off environment do not accurately represent ship movement. However, they are still interesting for this thesis, as it offers more complex driving patterns than the simple circles. Figure 7.7 shows an example pose sequence resulting from the project, where six agents move around in the closed-off environment, avoiding collisions. Each coloured line represents the track of one of the agents.

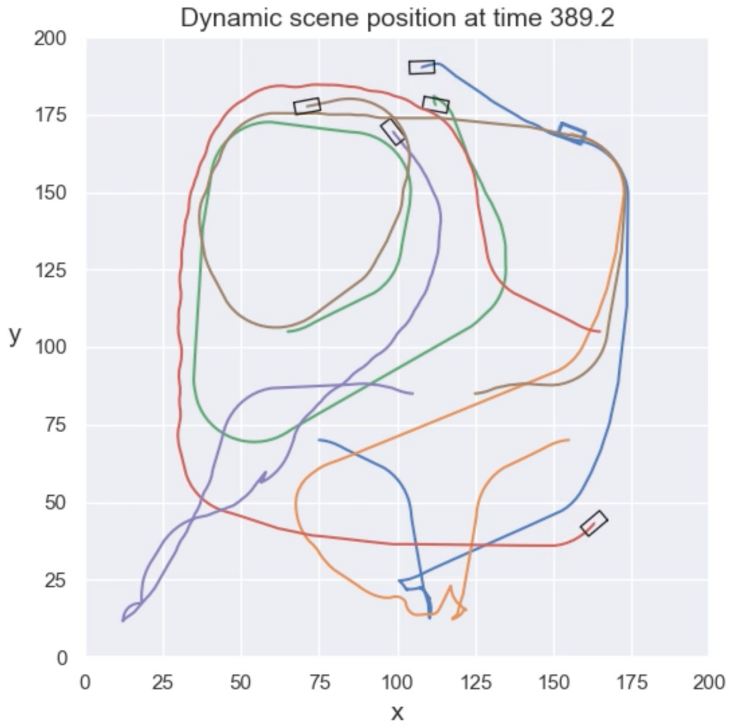


Figure 7.7.: A plot of a pose sequence from the agents made by Henrik Fjellheim at timestep 3892. This scene has six agents, and each coloured line represents an agent's track.

7.3. Camera Setup in MODSIM

This section describes the camera setup in the MODSIM system. The camera setup can either be stationary or mounted on one of the moving vessels, containing one or several virtual cameras. The virtual cameras projects the 3D corner points of the vessels in the scene onto a 2D image plane using a camera model described in Chapter 4. In order to perform projections, it is necessary to define the extrinsic and intrinsic parameters of the camera. Subsection 7.3.1 explains how to define the camera pose with respect to the WCF. How to determine the extrinsic and intrinsic camera parameters are presented in Subsections 7.3.3 and 7.3.2. Subsection 7.3 describes the information required to define a camera setup in MODSIM.

7.3.1. Describing the Camera Pose in the MODSIM System

The camera pose consists of the camera centre point in World Coordinate Frame (WCF) and the Camera Coordinate Frame (CCF) orientation relative to WCF. To describe the camera pose, it is crucial to establish a base pose that serves as the reference point for determining the camera's position and orientation. In the MODSIM system, the base pose is defined with the camera centre point in the origo of WCF and the camera viewing direction along the first unit vector of the WCF (i.e. $\vec{c}_3 \parallel \vec{w}_1$). Further, the first unit vector of CCF, $\vec{c}_1$, points in the negative direction of $\vec{w}_2$ and the second unit vector of the CCF, $\vec{c}_2$, points in the negative direction of $\vec{w}_3$ ($\vec{c}_1 \uparrow\downarrow \vec{w}_2 \wedge \vec{c}_2 \uparrow\downarrow \vec{w}_3$). The base pose can be viewed in Figure 7.8.

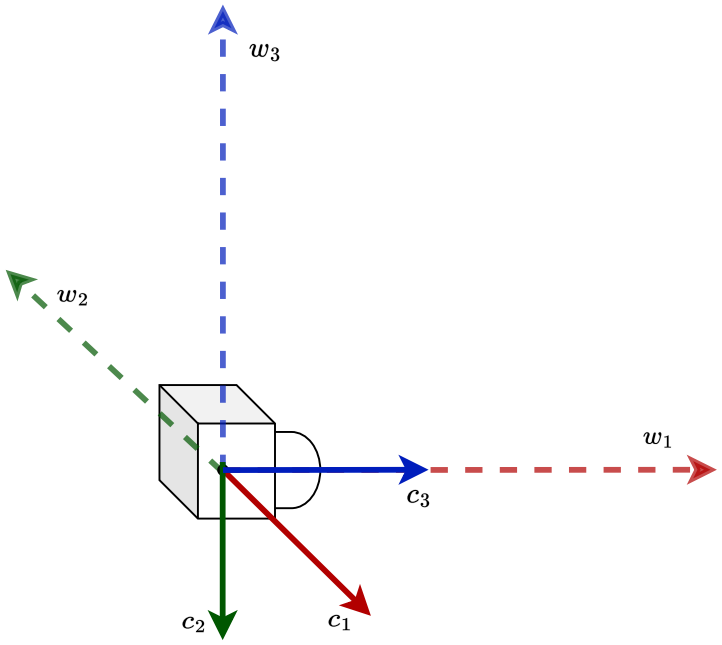


Figure 7.8.: Camera base pose in MODSIM.

The camera orientation with reference to the base pose can be described by rotation about the three unit vectors of the CCF using Euler angles, denoted θ, ψ, and ϕ Section 3.2.3 elaborates on Euler's angles. The three possible rotations are denoted *roll* (roll angle: θ), *pitch* (pitch angle: ϕ), and *yaw* (yaw angle: ψ), and can be viewed in Figure 7.9. In the base pose, all rotation angles are equivalent to 0 ($\theta = \phi = \psi = 0$).

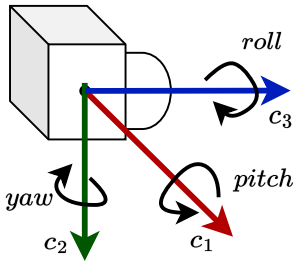


Figure 7.9.: Roll, pitch and yaw in the Camera Coordinate Frame (CCF).

A roll in the CCF is defined as a counterclockwise rotation following the right-hand rule about the third unit vector, $\vec{c}_3$. Assume frame C_r is the resulting frame after a roll rotation of CCF by the roll angle θ. A point p expressed in C_r as p^{C_r} is related to the point expressed in CCF as p^C by the following equation,

$$p^C = \mathbf{R}_{\vec{c}_3}(\theta) p^{C_r}, \tag{7.8}$$

where the rotation matrix $\mathbf{R}_{\vec{c}_3}(\theta)$ is defined as

$$\mathbf{R}_{\vec{c}_3} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7.9}$$

A pitch in the CCF is defined as a counterclockwise rotation following the right-hand rule about the first unit vector, $\vec{c}_1$. Let C_p be the coordinate system resulting from rotating the CCF by the pitch angle ϕ. If a point p is expressed in C_p as p^{C_p}, the point expressed in CCF, denoted as p^C, can be derived by the following equation,

$$p^C = \mathbf{R}_{\vec{c}_1}(\phi)p^{C_p}, \tag{7.10}$$

where

$$\mathbf{R}_{\vec{c}_1}(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}. \tag{7.11}$$

A yaw in the CCF is defined as a counterclockwise rotation following the right-hand rule about the second unit vector, $\vec{c}_2$. Let C_y be the coordinate system resulting from a yaw of the CCF by the yaw angle ψ. Given a point p expressed in C_y as p^{C_y}, p^C can be found by the following equation,WCF

$$p^C = \mathbf{R}_{\vec{c}_2}(\psi)p^{C_y}, \tag{7.12}$$

where

$$\mathbf{R}_{\vec{c}_2}(\psi) = \begin{bmatrix} \cos\psi & 0 & \sin\psi \\ 0 & 1 & 0 \\ -\sin\psi & 0 & \cos\psi \end{bmatrix} \tag{7.13}$$

7.3.2. Extrinsic Camera Parameters in the MODSIM System

The extrinsic parameters comprise a rotation matrix from WCF to CCF, denoted $\mathbf{R}_C^W$, and a translation vector from WCF to CCF, denoted $\vec{t}_C^W$. In the case of a stationary camera, these rotation and translation parameters remain constant. The camera pose changes over time for a dynamic camera mounted on a vessel. Thus, the extrinsic parameters are subject to temporal variations.

Stationary Camera

The rotation matrix in a stationary camera comprises five consecutive rotation steps, summarised in Table 7.3.

Rotation step	Rotation about unit vector	Angle	Frame
1	$\vec{c}_1$	$\pi/2$	Initial
2	$\vec{c}_2$	$3\pi/2$	Initial
3	$\vec{c}_2$	ψ	Initial
4	$\vec{c}_1$	ϕ	Initial
5	$\vec{c}_3$	θ	Initial

Table 7.3.: Rotation steps from WCF to CCF. The angles are given in radians.

Since the camera's orientation is defined with respect to the camera base pose (i.e. when $\theta = \phi = \psi = 0$), it is necessary to rotate from the WCF to the base pose CCF. This is performed by

two consecutive rotation steps; first rotating $\pi/2$ radians counterclockwise about $\vec{c}_1$ (pitch), and then $3\pi/2$ radians counterclockwise about $\vec{c}_2$ (yaw). The two rotation steps can be joined into a common rotation matrix,

$$\mathbf{R}_{base} = \mathbf{R}_{\vec{c}_2}\left(\frac{3\pi}{2}\right) \cdot \mathbf{R}_{\vec{c}_1}\left(\frac{\pi}{2}\right) = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{bmatrix} \tag{7.14}$$

The remaining steps in the rotation process account for the camera's orientation, which is described by the Euler angles θ, ψ, and ϕ. The third step is a yaw rotation, which is a counterclockwise rotation about the $\vec{c}_2$ axis by the yaw angle ψ. This step is represented by the rotation matrix $\mathbf{R}_{\vec{c}_2}(\psi)$ (Equation (7.13)). The fourth step is a counterclockwise rotation about the $\vec{c}_2$ axis by the pitch angle ϕ and is represented by the rotation matrix $\mathbf{R}_{\vec{c}_1}(\phi)$ (Equation (7.11)). Lastly, the fifth rotation step is a rotation about the $\vec{c}_3$ axis by the roll angle θ, which is represented by the rotation matrix $\mathbf{R}_{\vec{c}_3}$ (Equation (7.9)).

The complete rotation from WCF to CCF is the following,

$$\mathbf{R}_C^W = \mathbf{R}_{\vec{c}_3}(\theta) \cdot \mathbf{R}_{\vec{c}_1}(\phi) \cdot \mathbf{R}_{\vec{c}_2}(\psi) \cdot \mathbf{R}_{base}. \tag{7.15}$$

The translation vector points from the origo of CCF to the origo of WCF. The origo of the CCF expressed in WCF is given by the position of the centre point of the camera $\vec{s}^W = (\vec{s}_1^W, \vec{s}_2^W, \vec{s}_3^W)$. Therefore, the translation vector expressed in WCF is the negation of $\vec{s}^W$. It is necessary to express the translation vector in the CCF. This is done by rotating the negation of $\vec{s}^W$ using the rotation matrix $\mathbf{R}_C^W$ derived above,

$$\vec{t}_C^W = \mathbf{R}_C^W \cdot -\vec{s}^W. \tag{7.16}$$

Dynamic Camera Mounted on a Moving Vessel

When a camera is mounted on a moving vessel, the camera pose remains fixed with reference to the VCF, meaning that the position and orientation of the camera on the vessel remain constant throughout the simulation. However, as the vessel moves through space, the position and orientation of the camera with reference to WCF will alter along with the VCF, ultimately influencing the camera's extrinsic parameters.

At each time step, the vessel's motion causes the translation vector to change, representing the camera's position with respect to the WCF. Moreover, as the vessel changes its orientation in space, the rotation matrix also changes, indicating the camera's orientation with respect to the WCF. Consequently, to obtain the correct extrinsic parameters for a dynamic camera mounted on a moving vessel, it is essential to determine the camera pose at every discrete time step.

As described in Section 7.3.1, the orientation of the camera with reference to the base pose can be described by a roll (roll angle θ), pitch (pitch angle ϕ) and yaw (yaw angle: ψ). For a dynamic camera mounted on a vessel, these angles are affected by the fixed camera orientation on the vessel and the vessel's orientation. The fixed camera orientation on the vessel can be described by a fixed roll (θ_f), pitch (ϕ_f), and yaw (ψ_f) angle from CCF to VCF. The angles are defined with a positive direction following the right-hand rule with regard to CCF. θ_f, ϕ_f, and ψ_f are 0 when $\vec{c}_3 \parallel \vec{v}_1$, $\vec{c}_1 \downarrow\uparrow \vec{v}_2$, and $\vec{c}_2 \downarrow\uparrow \vec{v}_3$. Figure 7.10 illustrates the ψ_f angle for a dynamic camera mounted on a vessel.

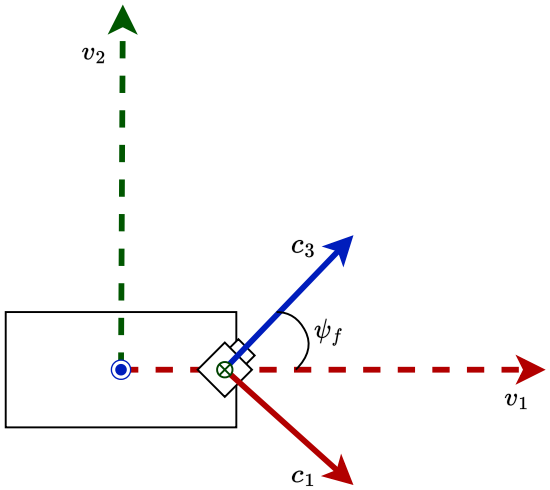


Figure 7.10.: Illustration of the fixed yaw angle, ψ_f, for a dynamic camera mounted on a vessel.

The pose sequences resulting from the dynamic scene generator are described in the nVCF. Under the assumption that there exists no rotation about the first or second unit vector of the VCF, VCF is equivalent to nVCF. Consequently, for a camera mounted on a moving vessel, the pitch and roll angles that describe the camera orientation with reference to the camera base pose are static. However, the yaw angle of the camera is influenced by the direction of the vessel's heading and is consequently dynamic.

Since VCF is equivalent to nVCF, the third unit vector of VCF is parallel to the third unit vector of WCF ($\vec{v}_3 \parallel \vec{w}_3$). Additionally, the second unit vector of CCF in the camera base pose is antiparallel to the third unit vector of WCF ($\vec{c}_2 \downarrow\uparrow \vec{w}_3$), which means that $\vec{c}_2$ in the camera base pose is antiparallel to $\vec{v}_3$,

$$\vec{v}_3 \parallel \vec{w}_3 \wedge \vec{c}_2 \downarrow\uparrow \vec{w}_3 \implies \vec{c}_2 \downarrow\uparrow \vec{v}_3 \tag{7.17}$$

Based on this relationship and the assumption that there exists rotation around $\vec{v}_1$ and $\vec{v}_2$ in VCF, the roll and pitch angles of the camera with respect to the camera base pose are equal to the fixed orientation of the camera on the vessel. The yaw angle, on the other hand, is the sum of the fixed orientation of the camera on the vessel and the angle between the heading direction of the vessel and the first unit vector of WCF for a timestep t. The roll, pitch, and yaw angles of the camera with respect to the camera base pose are defined as,

$$\theta = \theta_f \tag{7.18}$$

$$\phi = \phi_f \tag{7.19}$$

$$\psi_t = \psi_f + \alpha_t \tag{7.20}$$

Where ψ_t is the yaw angle for a given time step t and α_t is the angle between the heading direction of the vessel and the first unit vector of WCF at the time step t. α_t is derived in Equation (7.3).

Equally to the stationary camera, the rotation matrix for a dynamic camera is composed of five consecutive rotation steps, summarised in Table 7.4.

Rotation step	Rotation about unit vector	Angle	Frame
1	$\vec{c}_1$	$\pi/2$	Initial
2	$\vec{c}_2$	$3\pi/2$	Initial
3	$\vec{c}_2$	ψ_t	Initial
4	$\vec{c}_1$	ϕ	Initial
5	$\vec{c}_3$	θ	Initial

Table 7.4.: Rotation steps from WCF to CCF for a dynamic camera at time step t. The angles are given in radians.

The complete rotation from WCF to CCF for a given time step t is the following

$$\mathbf{R}_C^W(t) = \mathbf{R}_{\vec{c}_3}(\theta) \cdot \mathbf{R}_{\vec{c}_1}(\phi) \cdot \mathbf{R}_{\vec{c}_2}(\psi_t) \cdot \mathbf{R}_{base}. \tag{7.21}$$

Where $\mathbf{R}_{base}$ is defined in Equation (7.14).

The position of the centre point of the camera in WCF at a given time step t is defined as $\vec{s}_t^{\,W}$, giving the following translation vector.

$$\vec{t}_C^{\,W}(t) = \mathbf{R}_C^W(t) \cdot -\vec{s}_t^{\,W}. \tag{7.22}$$

7.3.3. Intrinsic Camera Parameters in the MODSIM System

The intrinsic parameters of the virtual camera(s) in the MODSIM system are dependent on the specific camera(s) that is being simulated. In order to execute the desired simulation, a range of input parameters are required, namely, the focal length (f), horizontal pixel size (Δp_1), vertical pixel size (Δp_2), the principal point (a_1, a_2), and skew (s) of the axis of the image plane. These values can typically be obtained from the camera model's datasheet or by camera calibration.

7.3.4. Incorporating a Camera Setup in MODSIM

In the MODSIM system, the camera setup encompasses the position and orientation of one or more virtual cameras. These virtual cameras can be stationary or dynamically mounted on a moving vessel. In the case of a dynamic camera setup, all cameras within the camera setup are mounted on the same vessel. The required user input for creating a static camera in the MODSIM software is listed in Table 7.5, and the required user input for creating a dynamic camera in the MODSIM software is listed in Table 7.5

User input	Description
Focal length (f)	The distance between the lens and the image sensor or film when the lens is in focus. Given in metres.
Horizontal pixel size (Δp_1)	The width of each individual pixel that makes up an image in the horizontal direction. Given in metres.
Vertical pixel size (Δp_2)	The height of each individual pixel that makes up an image in the vertical direction. Given in metres.
Principal point (a_1, a_2)	Where the optical axis strikes the image plane. Given in pixels.
Image bounds	The spatial extent of an image (i.e. the maximum coordinates in both the horizontal and vertical directions). Given in pixels.
Camera position in WCF	Placement of the camera in WCF. Given in metres.
Roll (θ)	The roll angle of the camera. Given in radians.
Pitch (ϕ)	The pitch angle of the camera. Given in radians.
Yaw (ψ)	The yaw angle of the camera. Given in radians.

Table 7.5.: Required user input for defining a static camera in the MODSIM system.

User input	Description
Focal length (f)	The distance between the lens and the image sensor or film when the lens is in focus. Given in metres.
Horizontal pixel size (Δp_1)	The width of each individual pixel that makes up an image in the horizontal direction. Given in metres.
Vertical pixel size (Δp_2)	The height of each individual pixel that makes up an image in the vertical direction. Given in metres.
Principal point (a_1, a_2)	Where the optical axis strikes the image plane. Given in pixels.
Image bounds	The spatial extent of an image (i.e. the maximum coordinates in both the horizontal and vertical directions). Given in pixels.
Vessel ID	The ID of the vessel the camera should be mounted on.
Camera position in VCF	Placement of the camera on the vessel in VCF. Given in metres.
Fixed roll (θ_f)	The camera's fixed roll angle when mounted on the vessel. Given in radians.
Fixed pitch (ϕ_f)	The pitch angle of the camera in radians. Given in radians.
Fixed yaw (ψ_f)	The yaw angle of the camera in radians. Given in radians.

Table 7.6.: Required user input for defining a dynamic camera in the MODSIM system.

7.4. Wave-Induced Motion

The pose sequences defined in the dynamic scene generator are referred to as the nominal pose (nVCF), where the 3D shoe boxes always are parallel to the 2D surface. When there are no rotations around the first or second unit vector of the VCF ($\vec{v}_1$ and $\vec{v}_2$), the VCF is equal to the nVCF. However, when a vessel moves on water, it experiences random motion due to waves and ocean movement. When introducing these wave effects on the vessels, the VCF and nVCF are not always equal. The VCF will follow the vessel in the motion caused by the waves, while nVCF will remain fixed to the nominal pose. This section introduces wave motion and explains the methodology adopted for simulating wave motion in the MODSIM system.

7.4.1. Ship Dynamics

In general, a ship has six degrees of freedom (DOF), namely, surge ξ, sway η, heave ζ, roll ϕ, yaw θ, and pitch ψ, as described by Kornev (2012). These motions are illustrated in Figure 7.11. Heave refers to the vertical motion of the ship caused by the waves moving up and down beneath the ship. Sway refers to the side-to-side motion of the ship, which is caused by the waves hitting the ship from the side. Surge is the forward and backward motion of the ship caused by the waves pushing the ship back and forth. Roll is the motion of the ship around its longitudinal axis, while pitch is the motion around the transverse axis. Finally, yaw refers to the motion of the ship around the vertical axis. Kornev (2012) elaborates on ship oscillations. Heave, sway, and surge are translational motions, whereas roll, pitch, and yaw are rotational motions. Figure 7.12 shows how the water affects vessels, causing pitch, roll, and yaw motion.

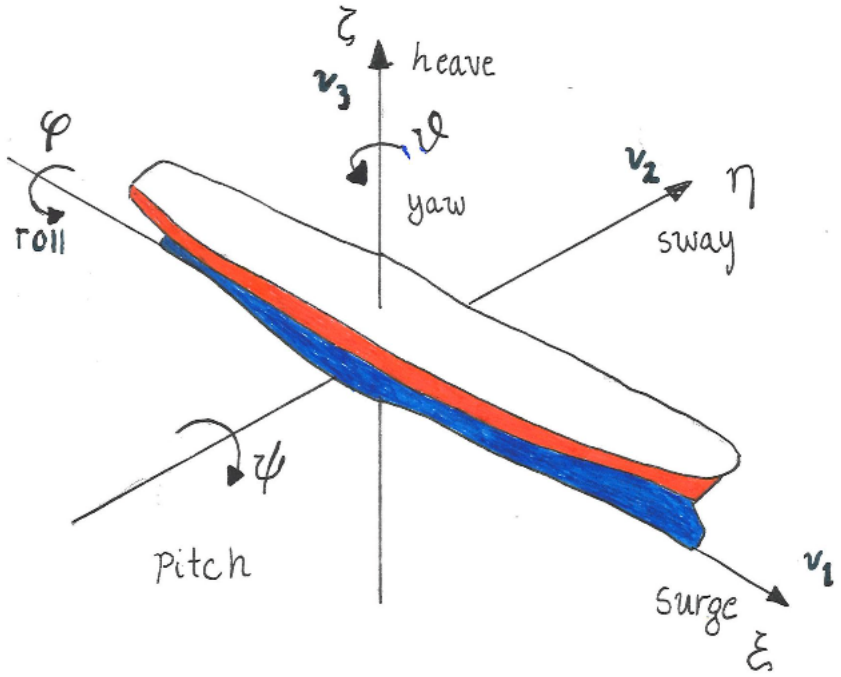


Figure 7.11.: Ship motion with 6 degrees of freedom, inspired by Kornev (2012).

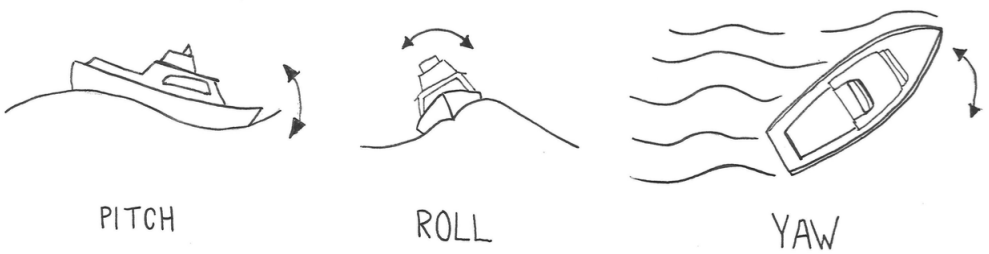


Figure 7.12.: Illustration of how a vessel rotates on water causing pitch, roll and yaw motion.

7.4.2. Effect of Wave Motion on Image Mapping for a Camera Setup Mounted on a Vessel

When a camera is mounted on a vessel and wave motion is introduced to the ownship, the camera setup will move along with the ownship as it is influenced by the wave motion. This causes the objects to move around in the virtual image plane, making tracking more challenging.

Wave motion induces both translational motion and rotational motion to the camera. A translational motion changes the position of the camera setup with respect to the WCF. A rotational motion changes the rotational angles of the camera. Adjusting the camera's roll angle alters the orientation of the ocean horizon relative to the horizontal line of the image, while increasing or decreasing the pitch angle of the camera causes objects to move upwards or downwards in the image frame, respectively, corresponding to a vertical movement in the image plane. Similarly, a change in the yaw angle of the camera results in a horizontal movement (left or right) in the image plane.

For the purpose of this simulation, it is reasonable to neglect translational motion as it is typically in the order of centimetres or a few tens of centimetres and, therefore, not significant in the image mapping when the objects being tracked are several metres away. In contrast, even a small rotational motion can substantially impact the image mapping.

This can be demonstrated by projecting an arbitrary point, such as the centre of the image, and calculating how much the same point moves in the image during a small translation (Case A) or a small rotation (Case B) of the camera. The theory of projecting real-world coordinates to the image plane is described in Chapter 4.

Assuming a legacy photo camera with a focal length f of 50 mm, a film size of 24 x 36 mm, and a resolution of 2400 x 3600 pixels. Position the camera in the base pose, i.e. in the world's centre, $p_c^W = (0, 0, 0)$, with $\vec{c}_3 || \vec{w}_1$, illustrated in Figure 7.8. Since the camera is positioned at the world's centre, no translation is necessary during coordinate transformation, i.e. $\vec{t}_C^W = \vec{0}$. Further, assuming a stationary camera, the rotation matrix is stationary and is defined in Section 7.3.2.

Define a point in the centre of the image, $p_0^W = (20, 0, 0)$, which is a point placed right in front of the camera 20 metres away. The projection of point p_0^W onto the image plane is achieved using Equation (4.18), yielding $p_0^C = (1800, 1200)$, which corresponds to the centre of the image.

In CASE A, where translational motion is considered, the range of motion caused by waves typically ranges from a few centimetres to a few tens of centimetres. For this scenario, assume a motion of 20cm in each direction. Translating the camera 20cm towards p_0^W yields $p_{0w_1}^C = (1800, 1200)$. Similarly, translating the camera by 20cm in positive $\vec{w}_2$ and $\vec{w}_3$ directions results in $p_{0w_2}^C = (1805, 1200)$ and $p_{0w_2}^C = (1800, 1205)$, respectively.

In CASE B, the mapping distance is calculated for small rotations of a camera of $\pi/40$ radians. A roll of $\pi/40$ radians results in $p_{0roll}^C = (1800, 1200)$. A pitch of $\pi/40$ radians results in $p_{0pitch}^C = (1800, 806.49)$, while a yaw of $\pi/40$ radians results in $p_{0yaw}^C = (2193.51, 1200)$.

Comparing these two cases, the translational motion only moves the point by 5 pixels, while the rotational motion results in a distance of approximately 394 pixels. Note that a roll rotation has no effect on a point in the centre of the image but would be visible for a point that is not along the c_3-axis. Additionally, moving the point further away along the c_3-axis will always project the same point.

In conclusion, these results demonstrate that even a small rotational motion can significantly affect the image mapping, while translational motion can reasonably be neglected for the purpose

of this thesis.

7.4.3. Simulating Wave Motion on a Vessel

Simulating wave motion on a vessel is done by generating a deviation between VCF and the nominal pose (nVCF) for each timestep t. This can be achieved using a stochastic process that generates a sequence of rotation matrices, $\mathbf{R}_w(t)$, between nVCF and VCF for each time step t. These matrices should only change moderately fast over time. The rotation matrix is modelled as the concatenated rotation around all three axes, performed in a fixed sequence, as described when defining the Euler angles in Section 3.2.3. Hence, three independent, slowly varying random processes are required for the three angles that constitute the rotation matrix.

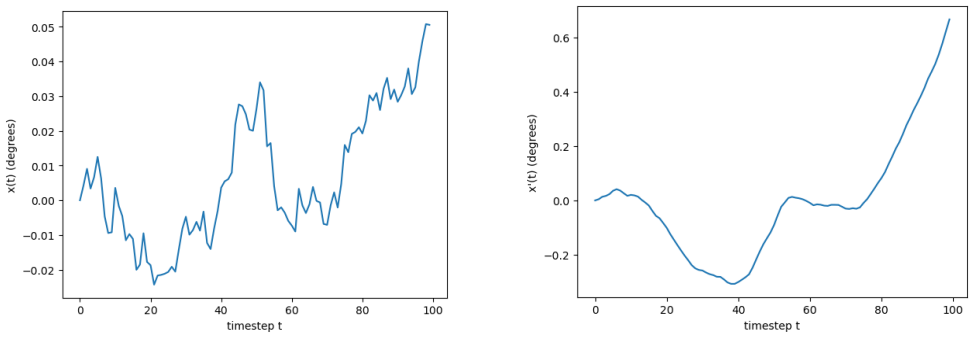
Such a process can be generated by first using a first-order autoregressive process,

$$x(t+1) = a \cdot x(t) + e(t+1) \tag{7.23}$$

Where a is slightly smaller than 1, and $e(t)$ is a zero mean random noise. For simulating waves in the MODSIM system, Gaussian noise is utilised, which is a statistical noise that follows a normal distribution. Then, one can build a second-order process by connecting two copies of the above generation equation:

$$x'(t+1) = b \cdot x'(t) + x(t+1) \tag{7.24}$$

The extent and rate at which the process should fluctuate can be regulated by modifying the quantity of zero-mean random noise. The choice of utilising second-order processes was made in order to make the process smoother. Figure 7.13 displays the plots of a first-order and second-order process, where $a = b = 0.99$, $e(t)$ is a normal distribution with a standard deviation of 0.0001 radians ($\sigma = 0.0001$). The initial values of x and x' are 0 ($x(0) = x'(0) = 0$). The plots show 100 timesteps with the angular deviation presented in degrees for the purpose of facilitating comprehension. The observation can be made that the second-order process exhibits smoother behaviour, which is more representational when dealing with waves.



(a) First-order autoregressive process, $x(t)$ (b) Second-order autoregressive process, $x'(t)$

Figure 7.13.: Plots of first-order and second-order autoregressive processes. The x-axis is the timestep t, and the y-axis is the angular deviation in degrees. 100 timesteps are generated using $a = b = 0.99$, $e(t)$ is a normal distribution with $\sigma = 0.0001$ radians, and $x(0) = x'(0) = 0$.

By generating three values from three independent random processes, one can build the rotation matrix $\mathbf{R}_w(t)$, one process for each of the rotation motions of the vessel (roll, pitch and yaw). The output of these processes should be regarded as an angular deviation from the nominal pose, where the wave VCF and the nVCF are perfectly aligned. The resulting rotation matrix $\mathbf{R}_w(t)$ is defined as follows,

$$\mathbf{R}_w(t) = \mathbf{R}_{\vec{c}_3}(\theta_w(t)) \cdot \mathbf{R}_{\vec{c}_1}(\phi_w(t)) \cdot \mathbf{R}_{\vec{c}_2}(\psi_w(t)) \tag{7.25}$$

Where the three independent random processes for roll, pitch and yaw wave motion are defined as:

$$x^{roll}(t+1) = a^{roll} \cdot x^{roll}(t) + e^{roll}(t+1) \tag{7.26}$$

$$\theta_w(t+1) = b^{roll} \cdot \theta_w(t) + x^{roll}(t+1) \tag{7.27}$$

$$x^{pitch}(t+1) = a^{pitch} \cdot x^{pitch}(t) + e^{pitch}(t+1) \tag{7.28}$$

$$\phi_w(t+1) = b^{pitch} \cdot \phi_w(t) + x^{pitch}(t+1) \tag{7.29}$$

$$x^{yaw}(t+1) = a^{yaw} \cdot x^{yaw}(t) + e^{yaw}(t+1) \tag{7.30}$$

$$\psi_w(t+1) = b^{yaw} \cdot \psi_w(t) + x^{yaw}(t+1) \tag{7.31}$$

It is important to note that the above method simplifies wave behaviour. In reality, waves are complex phenomena that are difficult to model accurately. It is also worth noting that the generated rotation matrices represent a deviation from the nominal pose rather than reflecting the actual motion induced by the waves. In practice, the motion of a vessel in waves is a complex process that involves various factors such as wave height, wave period, vessel speed, and vessel characteristics. However, this stochastic process provides a simplified method to approximate wave motion on the ownship for simulation purposes.

7.4.4. Adjusting Camera Pose for Wave Motion

Section 7.3.2 describes the camera's extrinsic parameters in instances where VCF matches nVCF. In the context of wave motion rotation around all axes of VCF is possible and VCF can no longer be assumed to align with nVCF. This means that for a dynamic camera mounted on a vessel, the camera's orientation will change along with the wave motion. Thus, the roll, pitch, and yaw angles relative to the camera base pose should also reflect the angular deviation introduced in this section, namely θ_w, ϕ_w, and ψ_w. As a result, the orientation angels of the camera with respect to the camera pose can now be defined as follows,

$$\theta_t = \theta_f + \theta_w \tag{7.32}$$

$$\phi_t = \phi_f + \phi_w \tag{7.33}$$

$$\psi_t = \psi_f + \alpha_t + \psi_w \tag{7.34}$$

Here, θ_t, ϕ_t, and ψ_t are the roll, pitch and yaw angles for a given timestep t, α_t is the angle between the heading direction of the vessel and the first unit vector of WCF at the time step t, and θ_f, ϕ_f, and ψ_f describe the fixed camera orientation on the vessel. The previously defined extrinsic and intrinsic camera parameters remain the same, and only the orientation angles are modified to account for wave motion.

7.4.5. A Simplified Approach to Modelling the Impact of Waves for Simulation Purposes

The extent and manner waves affect a vessel can vary depending on several factors. Attempting to replicate all of these complexities would be time-consuming and beyond the scope of this project. Consequently, a simplification has been made by introducing multiple degrees of impact that waves can have on the vessel. In other words, the simulated wave effect is a measure of how much the ownship is impacted by waves rather than the actual quantity of waves present in the scene.

In the current version of MODSIM system, there are five pre-defined degrees of wave impact on the ownship and, thereby, the camera. Namely, *no-wave-impact*, *small-wave-impact*, *medium-wave-impact*, *large-wave-impact*, and *huge-wave-impact*. This wave impact parameter is user-defined and remains constant throughout the simulation. The different degrees use the same method for simulating waves described above in Section 7.4.3 but with different standard deviations, σ, for the normal distribution that models the random noise, $e(t)$. Further, it was decided to only introduce wave motion to the ownship that the camera setup is mounted on, as introducing wave motion to the other vessels in the scene seen from the camera's perspective will not affect their position in the image significantly. As described in Section 7.4.2, the translational motion induced by waves is disregarded, and the sole focus is on rotational motions. The introduced approach provides a simplified representation that introduces wave-like movement that the tracker must be capable of handling. Although this approach can be further developed and refined, it provides a basic framework for making the simulated detections more realistic.

7.5. Annotation Generator

Annotations, in the context of visual object detectors, refer to marking or labelling specific objects or features within an image or video. The purpose of annotations is to provide information about the image content, which can be used to train and test visual object detectors. Annotations may take various forms, such as bounding boxes around objects, labelling of different parts or attributes of an object, or other types of markings that help identify and locate objects within an image or video. The annotations in the MODSIM system include Axis-Aligned Bounding Boxes (AABBs) and object classifications. The annotation generator produces classification labels corresponding to the user's specified vessel characteristics described in Section 7.2.1. These labels represent the true classifications of the vessels.

In the MODSIM system, the simulated detections are based on generated annotations and inputted error statistics reflecting the performance of the detector to be simulated. The annotations

generated by MODSIM must align with the annotation approach used when obtaining the error statistics to simulate the provided detector performance accurately. However, annotators may employ different methods and standards, especially when dealing with partially or fully covered vessels. To address this challenge, the MODSIM system provides three different annotation modes, allowing users to select the most appropriate annotation method reflecting how their test data was annotated while acquiring the error statistics.

7.5.1. Annotation Modes

Annotation mode 0, 1 and 2 have been introduced to account for different annotation methods. With annotation mode 0, all vessels in the image will have a perfectly enclosing bounding box, regardless of whether the vessel is fully or partially covered. This mode is illustrated in Figure 7.14. Annotation mode 1 creates bounding boxes around every vessel that is not fully covered; see Figure 7.15. Lastly, annotation mode 2 does not annotate vessels that er fully covered. If two neighbouring corners of the vessel are covered, only the visible part is annotated; see Figure 7.16.

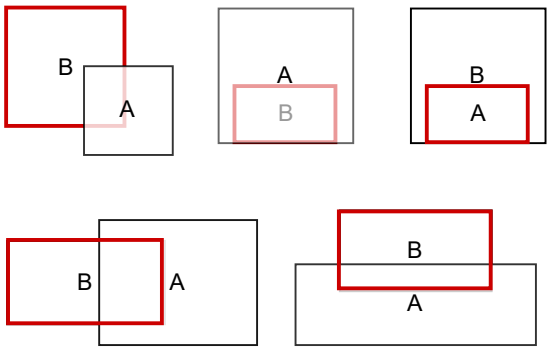


Figure 7.14.: An illustration of annotation mode 0. Square A is always closer to the camera than square B. The thick red line marks the resulting bounding box for square B.

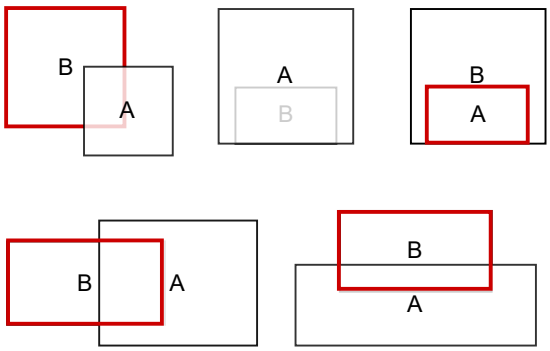


Figure 7.15.: An illustration of annotation mode 1. Square A is always closer to the camera than square B. The thick red line marks the resulting bounding box for square B.

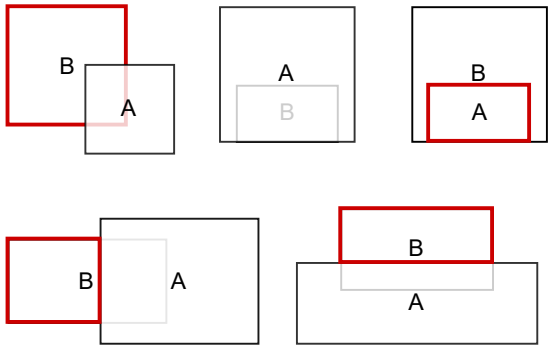


Figure 7.16.: An illustration of annotation mode 2. Square A is always closer to the camera than square B. The thick red line marks the resulting bounding box for square B.

7.6. Error Generator

Visual object detectors rarely work perfectly. Therefore, it is essential that MODSIM reflects the errors made by detectors. One of the main advantages of the MODSIM system is its capability to simulate errors and error modes through statistics and statistical distributions based on empirical data without the need to simulate the complex underlying causes of these errors. The error generator is responsible for simulating these errors and generating detector output that reflects the varying performance of object detectors in different conditions. The performance of an object detector differs across detectors. Consequently, the simulator needs to incorporate different parameters to represent the performance of the specific detector being simulated accurately. The following subsections elaborates on the type of errors object detector produces, how they are included in MODSIM, and the required user input for the error generator.

7.6.1. Types of Errors

Chapter 5 provides a comprehensive presentation on how to characterise a visual object detector and elaborates on the types of errors a visual object detector makes. These errors include false positives, drop-outs, localisation errors, and false classifications.

- **False Positives**: A false positive occurs when the detector incorrectly identifies an object not present in the image. For example, a detector might incorrectly identify an object in the water as a vessel when it is actually a wave crest or a patch of seaweed.
- **Drop-out**: A drop-out occurs when the detector fails to identify an object that is present in the image. For example, the detector fails to detect a small boat or kayak in the water, leading to a collision risk.
- **Localisation Errors**: Localisation errors occur when the detector correctly identifies an object but fails to locate it within the image accurately. This error includes misplacement of the bounding box, as well as drawing it too large or too small.
- **False classification**: A false classification occurs when the detector fails to assign the correct label to a detected object in the image.

7.6.2. Incorporating Errors

The error generator incorporates the errors in detectors by utilising the performance metrics for visual object detectors described in Chapter 5.

To incorporate false classification errors and drop-outs the error generator utilises a probabilistic confusion matrix. The matrix is of size $(k + 1) \times (k + 1)$, where k represents the number of possible classes and the additional row and column of the matrix represents the background (see Section 5.6.2). Given an annotated AABB of the true class a, the error generator will falsely classify it as another class according to the probabilities for false classification given by column a in the probabilistic confusion matrix. If the detected class is background (class $k+1$) the error generator creates a drop-out, meaning no identification of an object.

For a detection problem with only one class ($k = 1$), the error generator will only generate drop-outs and not false classifications. The rate at which drop-out should occur in single-class problems can be found in the probabilistic confusion matrix and is equal to the False Negative Rate (Equation (5.7)).

Section 5.6 explains that determining the number of true negatives in a visual object detector that generates AABBs is not feasible. Consequently, the false positive rate is undefined and can not be used for generating false positives. As an alternative approach, the False Discovery Rate (FDR) (Equation (5.6)) must be defined along with the desired confidence threshold (t). The direct calculation of the FDR from the probabilistic confusion matrix is not possible, but it can be calculated using the empirical confusion matrix. The implemented error generator produces false positives in an image based on the false discovery rate, the number of detections in the image (n), and the confidence threshold. Specifically, the probability of one false positive occurring is equal to the FDR. The maximum number of false positives in an image is defined by

$$FP_{max} = FDR * n - t. \tag{7.35}$$

To decide the total number of false positives in the image, a choice between 0 (no FP) or 1 (FP) is made FP_{max} times, where the probability of choosing 1 is equal to FDR and the probability of choosing 0 is equal to 1-FDR. The placement of the false positive detections are generated randomly from a uniform distribution between the horizon line and the bottom of the image.

The error generator determines localisation errors using a multivariate normal distribution with the given mean vector and covariance matrix of the bounding box error vectors (see Section 5.6).

The error generator assigns a confidence score to the detections using a normal distribution with a standard deviation of 0.1 and an expected value of 0.67. To constrain the generated values within a specific range, the normal distribution is cut to provide values only between the confidence threshold and 1. Consequently, if a value falls outside this designated range, it is adjusted to the nearest boundary value. This approach primarily gives the possibility of a confidence score rather than explicitly representing the distribution of confidence scores for the simulated detector.

7.6.3. Temporal Model

The performance of a detector will vary over time as the operating conditions changes. As discussed in Chapter 6, this type of behaviour can be modelled using a temporal model.

The temporal model consists of a finite set of states, with a given initial state. A transition matrix specifies the probabilities of transitioning between states, given the current state. Each state is associated with error metrics representing the performance of the object detection system in that state. These error metrics include the confusion matrix, false discovery rate, and standard deviation and expected value for the error distribution of the BB centre, height and width. For the error generator to properly reflect the performance of a visual object detector, the error metrics should be based on error statistics for the detector performance under different detection conditions.

Modelling the performance of object detectors in adverse weather conditions presents challenges due to differences among detectors. Each detector may have unique strengths and weaknesses, and their performance in adverse weather conditions can vary. For instance, one detector may perform poorly in low light conditions while another may excel in those conditions but struggle in heavy rain. Defining states in the temporal model based on factors like weather that influence detector performance is, therefore, challenging. To address this challenge, states in the temporal model are defined based on the resulting performance level of the detector rather than the specific causes of poor performance. This way, the temporal model reflects the varying performance of the detector over time. In the current iteration of MODSIM, there are four states defined, including *Terrible performance*, *Poor performance*, *Good performance*, and *Excellent performance*.

7.6.4. User Input Requirements in the Error Generator

The error metrics incorporated in the error generator must be defined by the user of the MODSIM system. The user must define the desired number of temporal states and the transition matrix that specifies the probabilities of transitioning between these states. The default number of states are the four states presented in the previous subsection. Additionally, the user must provide the initial state of the temporal model. For each state in the temporal model, the user must provide the mean vector and covariance matrix of the bounding box error vectors, a probabilistic confusion matrix, and a false discovery rate. Lastly, the user must provide the possible labels the detector can classify the detected objects as and a confidence threshold defining the minimum confidence level required for a detection to be outputted by the detector. The required user input is summarised in Table 7.7.

User input	Description
Possible labels	The possible labels the detector can classify the detected objects as.
Number of states	The desired number of temporal states in the model.
Initial state	The initial state of the temporal model.
Transition matrix	A matrix that specifies the probabilities of transitioning between the temporal states.
Mean vector	The mean vector, i.e. expected value vector, of the bounding box error vectors. One for each finite state in the temporal model.
Covariance matrix	The covariance matrix of the bounding box error vectors. One for each finite state in the temporal model.
Confidence threshold	A pre-defined minimum level of confidence required for a detection to be outputted by the detector
Probabilistic confusion matrices	One for each finite state in the temporal model reflecting the desired performance of the simulated detector under different detection conditions.
False discovery rate	One for each finite state in the temporal model reflecting the desired false discovery rate of the simulated detector under different detection conditions.

Table 7.7.: The necessary information that the error generator requires from the user in the MODSIM system.

7.7. Output Structure and Visualisation

The output of the MODSIM system consists of distorted bounding boxes, classifications, and confidence scores for each detected vessel at each timestamp. The detections are continuously written to a JSON file as they are generated for each timestep. Figure 7.17 illustrates the structure of the output generated by MODSIM. When testing a tracker, this output needs to be converted into the format specified by the tracker.

The MODSIM system implementation includes the possibility for visualisation. It provides visualisations for the dynamic scene, virtual camera output, annotations, and detections. These visualisations are valuable for detecting errors during code implementation, understanding the flow within the MODSIM system, and observing the level of error in the detections. The visualisation is not produced simultaneously with the simulation.

```
{
    "<timestamp>": {
        "<vesselID>": {
            "label": <string>,
            "confidenceScore": <float>,
            "bbox": {
                "centre": {
                    "x": <float>,
                    "y": <float>
                },
                "height": <float>,
                "width": <float>
            }
        },
        ...
    },
    ...
}
```

Figure 7.17.: The structure of the output generated by MODSIM.

8. Learning the Statistical Parameters from a Real Detector

This chapter presents the conducted experiments aimed at obtaining statistical parameters that represent a real detector. The objective of these experiments is to acquire the necessary statistical parameters for the error generator, discussed in Section 7.6, of the MODSIM system. To account for variations in different detection conditions, it is crucial to determine these parameters across a range of conditions that yield different levels of detection performance. The conducted experiments enable the generation of simulations using more realistic values and assess the feasibility of obtaining the required system parameters.

8.1. Available Datasets

For this master's thesis, two separate datasets were made available by DNV: the Hurtigruten dataset and synthetic images under various weather conditions.

8.1.1. Hurtigruten Dataset

The Hurtigruten dataset contains 4994 annotated images taken from a Hurtigruten ferry sailing along the coast of Norway. The images originate from NRK's TV program "Hurtigruten minutt for minutt" (NRK (2011)). They have a resolution of 1920×1088 pixels and were taken on different days at different times, resulting in varying light and weather conditions. The annotations in the dataset include Axis-Aligned Bounding Boxes (AABBs) surrounding vessels and their corresponding class labels. There are a total of 26 vessel classes, including "Open pleasure craft", "Semi-open pleasure craft", "Ferry", "Sailboat motor multiple", "Fishing vessel", and others. Most vessels in the dataset fall under the categories of "Open pleasure craft", "Semi-open pleasure craft", or "Enclosed pleasure craft". In total, there are 20 529 annotations in all the images, with 4978 of the 4994 images having an annotated object present. Figure 8.1 displays some example images from the dataset.

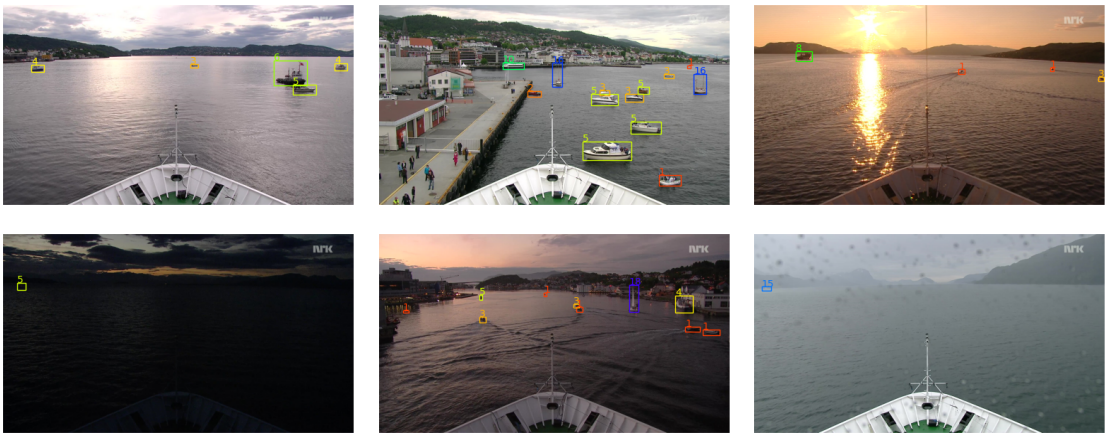


Figure 8.1.: Examples of annotated images in the Hurtigruten dataset, displaying varying light and weather conditions.

8.1.2. Synthetic Dataset under Various Weather Conditions

In order to generate images for various detection conditions, a synthetic dataset was created by DNV using the AILiveSim simulator (AILiveSim (2023)). The dataset was generated from a single scenario of 120 seconds, using an ownship with three cameras, which produced one image per camera every 0.533 seconds. The images have a resolution of 2464×2056 pixels. In addition to the ownship, the scenario contains three other vessels. The same scenario was generated in 15 different adverse weather conditions. The annotations for this dataset consist of AABBs not labelled with a class. Each detection condition in the dataset includes 675 images. The detection conditions cover a range of weather and lighting conditions, including "Noon clear", "Noon cloudy", "Noon cloudy rain", "Afternoon clear", "Afternoon cloudy", "Afternoon cloudy rain", "Evening clear", "Evening cloudy", "Evening cloudy rain", "Foggy clear", "Foggy cloudy", "Night clear", "Night cloudy", "Stormy clouds", "Stormy clouds rain". Figure 8.2 displays some example images from the dataset.

The precision of annotation in the synthetic dataset exhibits significant variation. Some AABB tightly enclose the corners of the vessels, while others encompass a larger portion of the surrounding areas. Figure 8.3 provides examples of different annotation styles employed in the dataset, highlighting this variability. Further, all vessels in the image are annotated regardless of whether or not they are visible. For instance, if a vessel is occluded by fog or another vessel, it is still annotated.

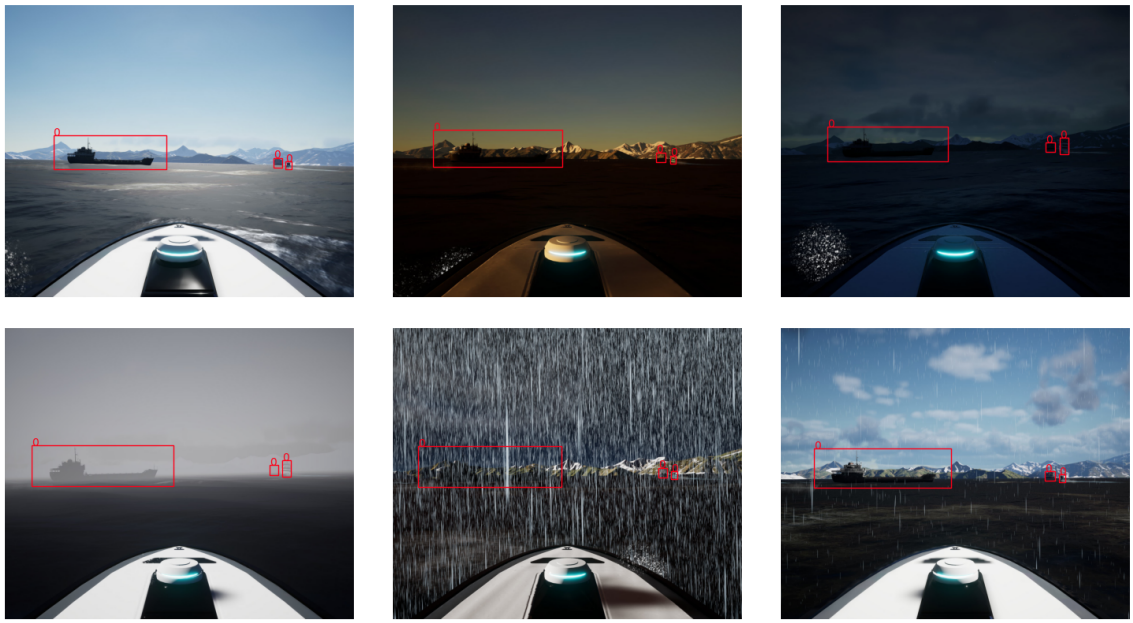


Figure 8.2.: Examples of annotated images taken at the same timestep in different conditions in the synthetic dataset. The images display the "Noon clear", "Evening clear", "Night cloudy", "Foggy clear", "Stormy clouds rain", and "Noon rain" in the respective order.

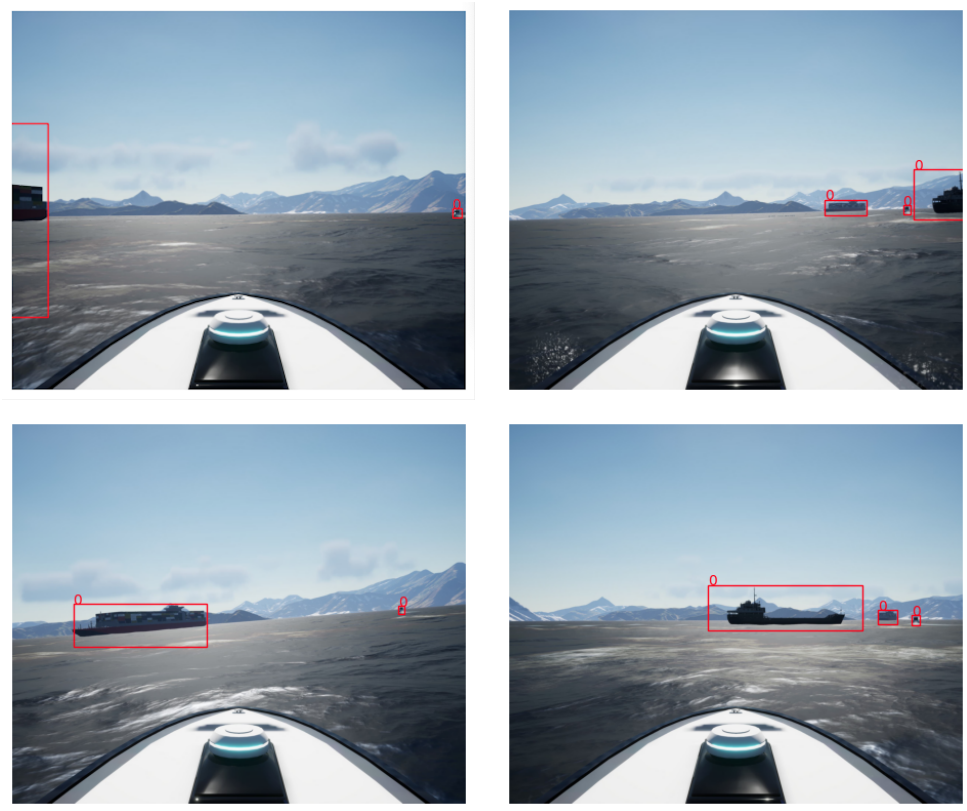


Figure 8.3.: Examples of the varying annotation quality.

8.2. Experimental Model

The experimental model employed was YOLOv8, a highly regarded object detection system renowned for being fast and accurate (Jocher et al. (2023)). As a one-stage detector, YOLO prioritises speed, rendering it particularly well-suited for real-time applications, such as those involving autonomous vessels. YOLO's user-friendly implementation and adaptable architecture enable it to be tailored to meet the specific requirements of various applications. YOLOv8 represents the most recent iteration of the YOLO series, offering even greater speed and precision than its predecessors.

8.3. Experimental Setup

8.3.1. Creating Training, Validation, and Test Datasets

The Hurtigruten dataset was shuffled and split into a training, validation, and test set, where 70%, 15%, and 15% of the images go into the different sets, respectively.

Further, the synthetic dataset was split into training, validation, and test sets. Initially, each detection condition was sorted chronologically based on time. Subsequently, they were split into two parts. The initial 60 timesteps constituted the training and validation set, with an 80/20 split between them respectively. The images were shuffled before splitting into training and validation. The final 60 timesteps were designated as the test set, and these images were not shuffled. During training, a combination of all the weather conditions was utilised, while each condition was individually assessed during testing.

Splitting the scenario ensures that the model does not train on a particular time step from one condition and subsequently gets tested on the same time step from a different condition. This methodology reduces the risk of potential biases or overlaps that may arise from having boats in identical locations across different conditions within the same scenario. Since the images are captured only 0.533 seconds apart, timesteps that are close to each other exhibit high similarity. Sorting the scenario chronologically based on time ensures that the model is not trained on certain timesteps and then tested on a timestep within close range of these trained timesteps. However, some overlap will exist at the split point.

8.3.2. Training of Models

Pre-trained weights were utilised to expedite the training process. The model was trained with a batch size of 16 and an input image size of 1024×1024. The default hyperparameters in YOLOv8 were used. This includes a Stochastic Gradient Descent (SGD) optimiser employed with a learning rate of 0.01, momentum of 0.937, and weight decay of 0.0005. The learning rate controls the amount by which the model's parameters are adjusted during training, with lower rates resulting in slower convergence but possibly better performance. Momentum affects how much the optimiser should rely on previous gradient updates when calculating the current update, with higher momentum values leading to faster convergence and potentially better generalisation. The weight decay parameter was utilised as a form of regularisation to prevent overfitting, penalising weights with large magnitudes to avoid over-reliance on specific features in the training data. The default patience of 50 was applied during training to prevent overfitting, causing the training process to stop if the validation loss failed to improve for 50 epochs. The default data augmentation techniques, such as random flips, translations, and scales, were used to enhance the model's ability to generalise to new data.

8.4. Experimental Plan

These experiments aim to learn the statistical parameters required by the MODSIM system. This is achieved by evaluating the performance of a visual object detector under different detection conditions. In order to evaluate and compare the model's performance across different conditions, various metrics and statistical measures were computed. These included precision (5.1), recall (5.2), F_1-score (5.9), False Negative Rate (FNR) (5.7), and False Discovery Rate (FDR) (5.6). Additionally, to evaluate the bounding box placement and dimension errors, the empirical covariance matrix (5.41) and mean vector (5.40) of the bounding box error vectors were calculated. The standard deviation and covariance coefficient were calculated from the covariance matrix to more easily examine the variability and relationships within the bounding box errors. These metrics are described in Chapter 5. Since the synthetic dataset only annotates vessels without classifying them, the experiments are conducted using a single class ("Vessel"). Consequently, the resulting probabilistic confusion matrix will have a size of 2×2, with one class representing "Vessel" and the other class representing the background. All the experiments use an Intersection over Union (IoU) threshold of 0.5 and a confidence threshold of 0.2.

8.4.1. Experiment 1: Real-World Dataset

The first experiment involves the Hurtigruten dataset. The YOLOv8 is trained for 100 epochs on the training set of Hurtigruten and tested on the Hurtigruten test set to assess its performance on real-world data. The model was trained on a single class ("Vessel"), to accommodate accommodate the classless annotation method in the synthetic dataset. To achieve this, all vessel classes within the Hurtigruten dataset are merged, and classes that do not fall under the vessel category, such as buoys, are removed.

8.4.2. Experiment 2: Synthetic Dataset with Good Conditions

In the second experiment, the trained model from Experiment 1 is tested on a synthetic dataset with good conditions. "Noon clear" is a condition with no adverse weather or lighting conditions that could impact the visibility or appearance of the vessels. This condition was chosen as a reference point to evaluate and establish a baseline for the model's performance.

8.4.3. Experiment 3: Synthetic Dataset with All Conditions

As an attempt to improve the model's performance on synthetic data, Experiment 3 involves fine-tuning the model by training for 50 epochs on the synthetic training dataset containing all conditions. After training, the model is tested on each individual condition of the synthetic test dataset separately. This evaluation allows for assessing how well the model can generalise to different conditions and adapt to variations in the synthetic dataset.

8.5. Experimental Results

8.5.1. Experiment 1

In Experiment 1 YOLOv8 was trained on the Hurtigruten training set and tested on its test set. Table 8.1 lists the resulting precision, recall, F_1-score, FNR, and FDR. Table 8.2 displays the expected value and the standard deviation of bounding box errors, while Table 8.3 displays the

covariance coefficient for each value pair of the bounding box error vectors. Lastly, Table A.1 in Appendix A displays the bounding box errors' mean vector and covariance matrix.

Detection condition	**Precision**	**Recall**	F_1**-score**	**FNR**	**FDR**
Real-world	0.881	0.858	0.869	0.142	0.119

Table 8.1.: **Experiment 1**: Precision, recall, F_1-score, False Negative Rate (FNR) and False Discovery Rate (FDR) for real-world conditions.

	Centre x		**Centre y**		**Width**		**Height**	
Detection condition	Exp	Std	Exp	Std	Exp	Std	Exp	Std
Real-world	-0.1	3.6	-0.9	3.5	0.1	7.2	0.0	7.7

Table 8.2.: **Experiment 1**: Bounding box errors in pixels (px) for real-world conditions, where Exp is the expected value, Std is the standard deviation, and the IoU threshold is 0.5.

Condition	ρ_{e_x,e_y}	ρ_{e_x,e_w}	ρ_{e_x,e_h}	ρ_{e_y,e_w}	ρ_{e_y,e_h}	ρ_{e_w,e_h}
Real-world	0.13	0.1	-0.09	-0.01	-0.52	0.14

Table 8.3.: **Experiment 1**: Covariance coefficients for the bounding box error vector value pairs for real-world conditions (IoU threshold: 0.5).

8.5.2. Experiment 2

In Experiment 2 the trained model from the first set of experiments was evaluated using the synthetic test set under the "noon clear" condition. This is considered to be a simple condition for the detector. Table 8.4 lists the resulting precision, recall, F_1-score, FNR, and FDR. Table 8.5 displays the expected value and the standard deviation of bounding box errors. Table 8.6 displays the covariance coefficient for each value pair of the bounding box error vectors. Table A.2 in Appendix A displays the mean vector and covariance matrix of the bounding box errors.

Detection condition	**Precision**	**Recall**	F_1**-score**	**FNR**	**FDR**
Noon clear	0.818	0.685	0.746	0.315	0.182

Table 8.4.: **Experiment 2**: Precision, recall, F_1-score, False Negative Rate (FNR) and False Discovery Rate (FDR) after training on Hurtigruten dataset and testing for good conditions ("Noon clear").

Detection condition	Centre x		Centre y		Width		Height	
	Exp	Std	Exp	Std	Exp	Std	Exp	Std
Noon clear	4.4	3.8	8.2	12.2	-5.4	10.3	13.3	35.2

Table 8.5.: **Experiment 2**: Bounding box errors in pixels (px) for good conditions ("Noon clear"), where Exp is the expected value, Std is the standard deviation, and the IoU threshold is 0.5.

Condition	ρ_{e_x,e_y}	ρ_{e_x,e_w}	ρ_{e_x,e_h}	ρ_{e_y,e_w}	ρ_{e_y,e_h}	ρ_{e_w,e_h}
Noon clear	-0.15	0.55	-0.07	-0.61	0.37	-0.19

Table 8.6.: **Experiment 2**: Covariance coefficients for the bounding box error vector value pairs for good conditions ("Noon clear") (IoU threshold: 0.5).

8.5.3. Experiment 3

During the third experiment, the previously used model was fine-tuned on the training set of the synthetic dataset, where all conditions are represented. The model was tested on each condition separately, using the test sets of the synthetic dataset. The results from Experiment 3 are presented in Table 8.7, Table 8.8 and Table 8.9. Table 8.7 lists the resulting precisions, recalls, F_1-scores, FNR, and FDR, while Table 8.8 displays the expected values and the standard deviations of bounding box errors. Table 8.9 displays the covariance coefficient for each value pair of the bounding box error vectors. Table A.3 and Table A.4 in Appendix A display the mean vector and covariance matrix of the bounding box errors for each individual condition.

Detection condition	Precision	Recall	F_1-score	FNR	FDR
Noon clear	0.911	0.867	0.889	0.133	0.089
Afternoon cloudy rain	0.908	0.866	0.886	0.134	0.092
Afternoon cloudy	0.893	0.861	0.877	0.139	0.107
Noon cloudy rain	0.905	0.829	0.865	0.171	0.095
Stormy clouds	0.889	0.825	0.856	0.175	0.111
Night clear	0.841	0.863	0.852	0.137	0.159
Stormy clouds rain	0.914	0.789	0.847	0.211	0.086
Noon cloudy	0.894	0.796	0.842	0.204	0.106
Night cloudy	0.867	0.802	0.833	0.198	0.133
Evening cloudy rain	0.900	0.761	0.825	0.239	0.100
Evening cloudy	0.876	0.761	0.815	0.239	0.124
Afternoon clear	0.881	0.738	0.803	0.262	0.119
Evening clear	0.848	0.734	0.787	0.266	0.152
Foggy clear	0.902	0.657	0.76	0.343	0.098
Foggy cloudy	0.745	0.619	0.676	0.381	0.255

Table 8.7.: **Experiment 3**: Precision, recall, F_1-score, False Negative Rate (FNR) and False Discovery Rate (FDR) for each detection condition after training on Hurtigruten dataset and synthetic mixed conditions sorted descending by the F_1-score.

	Centre x		Centre y		Width		Height	
Detection condition	Exp	Std	Exp	Std	Exp	Std	Exp	Std
Noon clear	4.3	48.5	6.8	19.8	-8.1	73.5	-0.9	35.0
Afternoon cloudy rain	-1.1	54.7	8.3	21.8	-10.6	80.3	-1.9	39.9
Afternoon cloudy	0.1	49.9	9.2	22.1	-10.6	68.4	-5.6	35.8
Noon cloudy rain	-4.0	43.8	3.5	14.8	-3.2	64.3	2.9	27.0
Stormy clouds	-4.7	72.7	10.2	25.2	4.3	102.7	-11.1	44.8
Night clear	10.3	62.9	9.4	22.2	-1.2	103.3	-3.0	26.5
Stormy clouds rain	-5.8	41.6	9.3	23.9	-3.3	63.9	-4.8	25.6
Noon cloudy	1.4	26.5	3.9	12.8	-3.0	45.2	2.8	25.5
Night cloudy	2.9	59.7	8.3	21.6	-0.2	98.8	-5.4	29.6
Evening cloudy rain	2.5	32.4	6.0	20.1	-10.0	57.6	-2.4	22.8
Evening cloudy	5.2	31.9	4.6	13.1	-12.6	50.7	-3.4	19.7
Afternoon clear	-1.5	21.5	5.3	19.1	-7.7	44.2	0.9	22.4
Evening clear	3.7	21.4	3.1	12.6	-6.8	41.1	-1.3	22.8
Foggy clear	5.3	47.7	0.2	13.1	7.8	65.8	2.6	23.6
Foggy cloudy	4.0	42.0	8.2	18.3	-24.1	71.9	-8.9	29.8

Table 8.8.: **Experiment 3**: Bounding Box Errors in pixels (px) for individual conditions sorted descending by the F_1-score for each condition. The tests are conducted with an IoU threshold of 0.5.

Condition	ρ_{e_x,e_y}	ρ_{e_x,e_w}	ρ_{e_x,e_h}	ρ_{e_y,e_w}	ρ_{e_y,e_h}	ρ_{e_w,e_h}
Noon clear	0.22	0.32	-0.1	-0.16	-0.56	0.04
Afternoon clear	-0.19	-0.62	-0.47	-0.5	0.17	0.03
Evening clear	0.37	0.04	-0.27	-0.58	0.02	0.19
Night clear	-0.01	0.55	0.22	-0.12	-0.04	0.17
Noon cloudy	0.19	-0.29	-0.23	-0.33	0.53	-0.09
Afternoon cloudy	-0.04	-0.05	-0.19	-0.2	-0.4	0.1
Evening cloudy	0.27	0.39	-0.33	-0.49	-0.39	0.12
Night cloudy	-0.0	0.3	0.11	-0.1	-0.4	0.09
Noon cloudy rain	0.17	-0.71	-0.14	-0.22	-0.64	-0.05
Afternoon cloudy rain	0.23	-0.42	-0.39	-0.27	-0.43	0.03
Evening cloudy rain	-0.06	0.12	-0.08	-0.41	0.18	-0.11
Foggy clear	0.22	0.92	-0.09	0.1	0.18	0.08
Foggy cloudy	0.22	0.59	-0.25	-0.45	-0.74	0.39
Stormy clouds	0.06	-0.15	-0.04	-0.01	-0.7	-0.05
Stormy clouds rain	-0.13	-0.68	-0.3	-0.45	-0.23	0.22

Table 8.9.: **Experiment 3**: Covariance coefficients for the bounding box error vector value pairs for individual conditions. The tests are conducted with an IoU threshold of 0.5.

8.6. Discussion of Experimental Results

This section discusses the results obtained from the experiments conducted and provides insights into the performance of the model under different conditions.

Experiment 1 involved training and testing the model on the Hurtigruten training and test set, respectively. The obtained results showed promising performance, with an F_1-score of 0.869. The precision value of 0.881 indicated a low number of false positives, while the recall value of 0.858 suggested successfully detecting a significant portion of the true positives. Analysis of the bounding box errors revealed accurate localisation of objects, as the expected values for centre x, centre y, width, and height were close to zero. The relatively small standard deviations for these errors (ranging from 3.6 to 7.7 pixels) further demonstrated the model's ability to estimate object boundaries accurately in real-world conditions. Little correlation between most value pairs of the error vector for the placement and dimensions of the bounding box was observed. However, a negative correlation of -0.52 existed between the error in the centre position in the y direction, e_y, and the height, e_h, of the bounding box. This indicates that large (small) errors in the y-centre position are often accompanied by small (large) errors in the height of the bounding box.

In Experiment 2 a significant recall drop was observed when the trained model was tested on the synthetic dataset with good conditions ("Noon Clear"). This indicates that the model missed more detections on the synthetic dataset than on the real-world dataset. Additionally, the false discovery rate increased from 0.119 to 0.182, and precision dropped from 0.881 to 0.818, suggesting an increase in false positives. Figure 8.4 illustrates an example of a false positive, where the model mistakenly identified text in the image as a boat.

The bounding box errors also increased for the synthetic dataset. One possible factor contributing to this increase could be the annotation method used in the synthetic dataset. It was observed that some of the annotated AABBs were much larger than the vessels, while others were accurately placed around the vessels (Figure 8.3). Consequently, even if the detector provided a bounding box tightly enclosing the vessel, the bounding box error could still be large. Figure 8.5a exemplifies the difference between annotated and detected bounding boxes resulting from Experiment 2. It can be observed that the detected bounding box encloses the vessel more accurately than the annotated bounding box. In the synthetic dataset, the vessels in the surrounding environment appear larger in the images than those in the Hurtigruten dataset. As a result, the vessels in the synthetic dataset have larger bounding boxes, and even a slight misplacement of the bounding box can result in a significant displacement in terms of pixels. The covariance coefficients in Experiment 2 differ from those of Experiment 1 in sign and magnitude, indicating different relationships between the value pairs of the bounding box error vectors in Experiment 1 and Experiment 2. The different dependencies could also be due to the inconsistent annotation methods used in the synthetic dataset.

Experiment 3 involved fine-tuning the detector on the synthetic dataset using a combination of all the detection conditions. The experiment resulted in performance improvements. For the "Noon clear" condition, the false discovery rate dropped from 0.182 in Experiment 2 to 0.089 in Experiment 3, and the precision increased from 0.818 to 0.911. These improvements indicate that the model produced fewer false positives. Moreover, the recall increased from 0.685 to 0.867, indicating fewer missed detections. It is worth noting that the bounding box errors increased, suggesting that training on the imprecise annotations in the synthetic dataset might confuse the model and lead to worse performance for bounding box localisation.

Figure 8.5 illustrates the difference in the detected bounding box between Experiment 2 and Experiment 3. It is evident that the bounding box in Experiment 2 tightly encloses the vessel, while in Experiment 3, the bounding box includes a significant portion of the water and sky. This indicates that the model adapts to the annotation style used in the synthetic dataset. The synthetic scenario is divided into two parts, with the first set of timesteps serving as the training set and the last set of timesteps as the test set. Consequently, the model is trained and tested on different boats. The annotation for the large boat in the training set differs from that in the

test set, as shown in Figure 8.6. In the training set, the annotation extends beyond the boat approximately equally in all directions (Figure 8.6a). The annotation in the test set extends mostly in the downward direction, including more water (Figure 8.6b). In Figure 8.5b, it can be observed that the detection of the large boat matches the annotation method used for the large boat in the training set. The difference in training and test annotations could explain the increase in bounding box errors and likely make the resulting statistics of bounding box variations less representative for the detector performance on other datasets.

The results of Experiment 3 revealed varying performance across different detection conditions. The highest F_1-score is observed for "Noon clear" and the lowest for "Foggy cloudy". The model achieved high precision scores for conditions such as "Stormy clouds rain," "Noon clear," and "Noon cloudy rain," indicating a low number of false positives. Precision scores were comparatively lower for conditions like "Foggy cloudy," indicating more false positives. The highest recall rates were observed in "Noon clear" and "Afternoon cloudy rain" conditions, while the lowest recall rates were observed in "Foggy clear" and "Foggy cloudy" conditions. There are few evident tendencies in the results indicating that the detector generally performs worse or better in certain simulated weather or light conditions. For instance, different light conditions are among the best and worst performances. There is no indication that rain provides worse performance. However, both conditions containing fog have the lowest recall rates, indicating a significant number of missed detections during foggy conditions.

There is no evident correlation between the model's detection performance and the spread in bounding box errors across different conditions. Despite the "Noon clear" condition achieving the highest F_1 score, it exhibited a relatively large spread in bounding box errors. Conversely, conditions with lower F_1 scores demonstrated both larger and smaller spreads in bounding box errors. For instance, the "Evening clear" condition exhibited a relatively low F_1 score of 0.787, expected values closer to zero and lower standard deviations for bounding box errors compared to the "Noon clear" condition. This suggests that the factors contributing to false positives and drop-outs may not significantly impact bounding box variations. Furthermore, no weather or light condition clearly led to an increase or decrease in errors in the dimensions and placements of the bounding box. These findings may be influenced by the imprecise annotations for the synthetic dataset. Therefore, it can not be concluded that these observations hold true when more precise annotations are employed.

It is important to note that the synthetic dataset used in the experiments has some limitations. Since the dataset is simulated from the same scenario for various conditions, the images in the dataset are very similar. This similarity may impact the model's ability to generalise well to different scenarios and may not fully represent the complexities and diversity of real-world scenarios. Since the dataset is synthetic, it may not adequately represent actual weather and light conditions. Figure 8.1 and Figure 8.2 illustrates the differences between real-world conditions and the synthetic dataset. Consequently, it is impossible to conclusively determine if the performance observed in these experiments reflects the model's performance on real-life data.

It is worth noting that the amount of time dedicated to fine-tuning the model and selecting optimal hyperparameters and training procedures was relatively limited in these experiments. Given the complexity of the task and the numerous factors that influence model performance, more extensive experimentation and optimisation could yield further improvements in detector performance.

In conclusion, the model's performance varied across different detection conditions. The highest F_1-score was observed in the "Noon clear" condition, and the lowest F_1-scores were observed in the "Foggy clear" and "Foggy cloudy" conditions. Notably, the presence of fog reduced the

model's performance, resulting in an increased number of missed detections. However, no consistent improvement or decrease in performance was observed for the remaining weather and light conditions. Further, there was no clear correlation between detection conditions and bounding box errors. The limitations of the synthetic dataset, including dissimilarity to real-world data and imprecise annotations, could affect the results, potentially limiting their representation of the detector's performance on real-world data. However, the results are sufficient for this thesis' experimental purposes and have provided the necessary statistical parameters for the error generator.

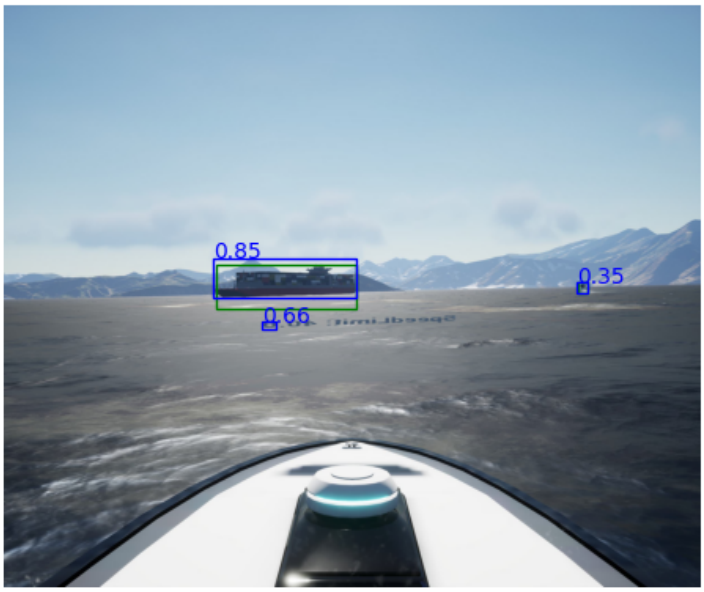


Figure 8.4.: Example of a false positive detection in Experiment 2. A letter in the simulation is detected as a false positive. The green rectangles represent the annotations, the blue rectangles represent the detections, and the blue number is the confidence score.

(a) Experiment 2 (b) Experiment 3

Figure 8.5.: Detections on the same image for training on The Hurtigruten dataset (Experiment 2) and after fine-tuning on the mixed synthetic dataset (Experiment 3). The green rectangles represent the annotations, the blue rectangles represent the detections, and the blue number is the confidence score

(a) Train image (b) Test image

Figure 8.6.: Examples of the varying annotation quality from the synthetic training and test sets.

9. Showcasing the MODSIM System via Simulations

9.1. Simulation Setup

In order to showcase the capabilities of MODSIM, three different simulations have been conducted. These simulations maintain consistent annotation modes and a temporal model but incorporate variations in pose data, wave impact, and camera setup. The experiments in Chapter 8 serve as the foundation for quantifying the error generator for the simulations. All simulations have a confidence threshold of 0.2.

9.1.1. Classification

The object detector trained and evaluated in Chapter 8 to derive the statistical parameters for these simulations was designed to detect a single class: "Vessel". Consequently, the simulations also consist of a single class, "Vessel". This setup gives rise to a probabilistic confusion matrix with dimensions of 2×2, where one class represents vessels while the other class represents the background.

9.1.2. Annotation Mode

In the synthetic dataset utilised to derive the statistical parameters in Chapter 8, the vessels are annotated using Axis-Aligned Bounding Boxes (AABBs), irrespective of whether the vessel is fully or partially covered. This annotation method corresponds to the annotation mode 0, defined in MODSIM (Section 7.5). Annotation mode 0 is therefore applied for all simulations.

9.1.3. Error Generator and Temporal Model

The experiments conducted in Chapter 8 assessed the performance of 15 different detection conditions, forming the basis for quantifying the error generator of MODSIM. Table 9.1 presents the conditions ranked in descending order of their F_1 score. The table also includes separating lines to indicate how the conditions are categorised into the four states: *Terrible performance*, *Poor performance*, *Good performance*, and *Excellent performance*. Table 9.2 and Table 9.3 provide the parameters for each temporal state.

As identified in Chapter 8, the "Noon clear" condition demonstrates the highest overall performance and the highest F_1 score, making it representative of the *Excellent performance* state. Conversely, the "Foggy cloudy conditions" yield the worst performance and will be utilised as the foundation for the *Terrible performance* state.

To establish the basis for *Good performance*, the average performance metrics are calculated from the following conditions: "Afternoon cloudy rain", "Afternoon cloudy", "Noon cloudy rain", "Stormy clouds", and "Night clear". These conditions exhibit higher F_1 scores ranging from 0.852

to 0.886 and lower false negative rates ranging from 0.134 to 0.175, compared to the remaining conditions.

The error metrics for *Poor performance* are derived by averaging over the remaining conditions, namely "Stormy cloud rain", "Noon cloudy", "Night cloudy", "Evening cloudy rain", "Evening cloudy", "Afternoon clear", "Evening clear", and "Foggy clear". As evident in Table 9.1, these conditions display an increase in false negatives and a decrease in the F_1 score.

In the experiments conducted in Chapter 8, no clear correlation was found between the model's detection performance and the variations in bounding box errors across different conditions. Therefore, the bounding box variations are not considered when sorting the conditions into the four states. As a result, the bounding box performance varies across the different performance states, and no state demonstrates a clear superior bounding box performance.

Detection condition	**Precision**	**Recall**	**F_1-score**	**FNR**	**FDR**
Noon clear	0.911	0.867	0.889	0.133	0.089
Afternoon cloudy rain	0.908	0.866	0.886	0.134	0.092
Afternoon cloudy	0.893	0.861	0.877	0.139	0.107
Noon cloudy rain	0.905	0.829	0.865	0.171	0.095
Stormy clouds	0.889	0.825	0.856	0.175	0.111
Night clear	0.841	0.863	0.852	0.137	0.159
Stormy clouds rain	0.914	0.789	0.847	0.211	0.086
Noon cloudy	0.894	0.796	0.842	0.204	0.106
Night cloudy	0.867	0.802	0.833	0.198	0.133
Evening cloudy rain	0.900	0.761	0.825	0.239	0.100
Evening cloudy	0.876	0.761	0.815	0.239	0.124
Afternoon clear	0.881	0.738	0.803	0.262	0.119
Evening clear	0.848	0.734	0.787	0.266	0.152
Foggy clear	0.902	0.657	0.76	0.343	0.098
Foggy cloudy	0.745	0.619	0.676	0.381	0.255

Table 9.1.: Precision, recall, F_1-score, False Negative Rate (FNR) and False Discovery Rate (FDR) for each detection condition after training on Hurtigruten dataset and synthetic mixed conditions sorted descending by the F_1-score, resulting from Experiment 3 in Chapter 8. The dashed lines represent the chosen division of conditions that should influence the defined states.

Performance	**TNR**	**FNR**	**FDR**
Excellent	0.867	0.133	0.089
Good	0.849	0.151	0.113
Bad	0.755	0.245	0.155
Terrible	0.619	0.381	0.255

Table 9.2.: The True Negative Rate (TNR), False Negative Rate (FNR), and False Discovery Rate (FDR) used in the temporal error model for the simulations.

Performance	Mean vector	Covariance matrix
Excellent	$\begin{matrix} e_x \\ e_y \\ e_w \\ e_h \end{matrix}\begin{bmatrix} 4 \\ 6 \\ -8 \\ 0 \end{bmatrix}$	$\begin{matrix} \\ e_x \\ e_y \\ e_w \\ e_h \end{matrix}\begin{matrix} \begin{matrix} e_x & e_y & e_w & e_h \end{matrix} \\ \begin{bmatrix} 2354 & 206 & 1157 & -169 \\ 206 & 390 & -227 & -389 \\ 1157 & -227 & 5404 & 115 \\ -169 & -389 & 115 & 1223 \end{bmatrix} \end{matrix}$
Good	$\begin{matrix} e_x \\ e_y \\ e_w \\ e_h \end{matrix}\begin{bmatrix} 0 \\ 8 \\ -4 \\ -4 \end{bmatrix}$	$\begin{matrix} \\ e_x \\ e_y \\ e_w \\ e_h \end{matrix}\begin{matrix} \begin{matrix} e_x & e_y & e_w & e_h \end{matrix} \\ \begin{bmatrix} 3328 & 87 & -301 & -220 \\ 87 & 462 & -254 & -354 \\ -301 & -254 & 7296 & 93 \\ -220 & -354 & 93 & 1262 \end{bmatrix} \end{matrix}$
Bad	$\begin{matrix} e_x \\ e_y \\ e_w \\ e_h \end{matrix}\begin{bmatrix} 2 \\ 5 \\ -4 \\ -1 \end{bmatrix}$	$\begin{matrix} \\ e_x \\ e_y \\ e_w \\ e_h \end{matrix}\begin{matrix} \begin{matrix} e_x & e_y & e_w & e_h \end{matrix} \\ \begin{bmatrix} 1407 & 20 & 349 & -127 \\ 20 & 309 & -315 & -14 \\ 349 & -315 & 3717 & 104 \\ -127 & -14 & 104 & 584 \end{bmatrix} \end{matrix}$
Terrible	$\begin{matrix} e_x \\ e_y \\ e_w \\ e_h \end{matrix}\begin{bmatrix} 4 \\ 8 \\ -24 \\ -8 \end{bmatrix}$	$\begin{matrix} \\ e_x \\ e_y \\ e_w \\ e_h \end{matrix}\begin{matrix} \begin{matrix} e_x & e_y & e_w & e_h \end{matrix} \\ \begin{bmatrix} 1768 & 167 & 1796 & -319 \\ 167 & 336 & -594 & -402 \\ 1796 & -594 & 5172 & 826 \\ -319 & -402 & 826 & 888 \end{bmatrix} \end{matrix}$

Table 9.3.: The mean vector and covariance matrix of the bounding box errors used in the temporal error model for the simulations.

Transition Matrix

As discussed in Section 6.3, it is commonly expected that in dynamic environments with changing conditions, the probability of remaining in a particular state is higher than the probability of transitioning to a new state. Additionally, transitions to nearby states are more likely than transitions to states that are further away. For example, it is more probable to transition from the *Excellent performance* state to the Good performance state than from the *Excellent performance* state to the *Terrible performance* state. While real-world conditions may change over hours or days, the simulations aim to present this change within a shorter timeframe, such as seconds or minutes. This allows for illustrating the impact of changing performance states within the simulation duration. All of these factors are taken into account when constructing the following transition matrix,

$$\mathbf{T} = \begin{bmatrix} 0.99000 & 0.00475 & 0.00050 & 0.00050 \\ 0.00850 & 0.99000 & 0.00475 & 0.00100 \\ 0.00100 & 0.00475 & 0.99000 & 0.00850 \\ 0.00050 & 0.00050 & 0.00475 & 0.99000 \end{bmatrix}. \tag{9.1}$$

9.1.4. Simulation 1

Pose Data

In the first simulation, the pose data is generated using a circular motion model, as described in Section 7.2.3. Each vessel's radius and rotation rate are randomly selected within the specified minimum and maximum values. The frequency, i.e. time difference between each time step, is 0.2 seconds. The parameters presented in Table 9.4 were employed to generate pose data.

Parameter	**Value**	**Unit**
Number of ships (n)	5	
Timesteps (t)	1500	
Min radius (R_{min})	5	m
Max radius (R_{max})	500	m
Position vector of circle centre ($\vec{p}_0$)	$[1000, 1000]^T$	m
Min rotation rate (ω_{max})	0.005	rad/s
Max rotation rate (ω_{min})	0.03	rad/s
Frequency (ΔT)	0.2	s

Table 9.4.: Parameters for generation of pose sequences using a circular motion model in Simulation 1.

Camera Setup

The camera setup in the first simulation consists of one static "simple legacy camera". The camera is placed at a distance, d, such that the camera captures the entire circle with the largest possible radius of the circular motion model. The camera is placed at the height of 20 metres and points towards the centre of the circles. The calculations of the extrinsic camera parameters are described in Appendix B. The resulting parameters are summarised in Table 9.5.

Parameter	**Value**	**Unit**
Focal length (f)	50	mm
Horizontal pixel size (Δp_1)	0.01	mm
Vertical pixel size (Δp_2)	0.01	mm
Principle point (a_1, a_2)	(1800, 1200)	px
Width of sensor	36	mm
Image bounds	(3600, 2400)	px
Camera position WCF	(2444, 1000, 20)	m
Roll (θ)	0	rad
Pitch (ϕ)	0.008	rad
Yaw (ψ)	π	rad

Table 9.5.: Camera setup in simulation 1.

Wave Motion

For the first simulation, there is no wave impact on the camera.

9.1.5. Simulation 2

Pose Data

The second simulation employs the pose data made available by Henrik Fjellheim (see Section 7.2.4). For the used pose sequence, six vessels are moving around in the closed-off environment, avoiding collisions. The time difference between each time step is 0.1 seconds. The number of timesteps is 5000. The parameters employed in Simulation 2 are displayed in Table 9.6.

Parameter	**Value**	**Unit**
Number of ships (n)	6	
Timesteps (t)	5000	
Frequency (ΔT)	0.1	seconds

Table 9.6.: Parameters for dynamic scene generation in Simulation 2.

Camera Setup

The camera setup in the second simulation consists of a dynamic camera placed on one of the vessels in the pose sequence. It is placed in the middle of the front of the vessel, at the highest point. The orientation of the camera is the same as the orientation of the vessel, except for a pitch of $\frac{\pi}{200}$ radians. The camera has the same intrinsic parameters as the camera used for creating the images in the synthetic dataset in Chapter 8. The parameters used for the camera setup are described in Table 9.7.

Parameter	**Value**	**Unit**
Focal length (f)	23.7	mm
Horizontal pixel size (Δp_1)	0.001	mm
Vertical pixel size (Δp_2)	0.00083	mm
Principle point (a_1, a_2)	(1232, 1028)	px
Image bounds	(2464, 2056)	px
Camera position VCF	(4, 0, 1)	m
Fixed roll (θ_f)	0	rad
Fixed pitch (ϕ_f)	$\pi/200$	rad
Fixed yaw (ψ_f)	0	rad
VesselID	'vehicle1'	

Table 9.7.: Camera setup in simulation 2.

Wave Motion

In the second simulation, the wave impact on the camera is of type *small-wave-impact*. The initial values for the first and second autoregressive process that simulates the wave motion are set to 0 for all angles. A summary of the parameters employed for the wave motion in Simulation 2 is presented in Table 9.8.

Parameter	Value (radians)
a^{roll}	0.99
a^{pitch}	0.99
a^{yaw}	0.99
b^{roll}	0.99
b^{pitch}	0.99
b^{yaw}	0.99
σ^{roll}	0.000050
σ^{pitch}	0.000030
σ^{yaw}	0.000015

Table 9.8.: Wave motion parameters for Simulation 2.

9.1.6. Simulation 3

Pose Data

The third simulation also uses the pose data made available by Henrik Fjellheim (see Section 7.2.4). For the used pose sequence, six vessels are moving around in the closed-off environment, avoiding collisions. The time difference between each time step is 0.1 seconds. The number of timesteps is 5000. The parameters employed in Simulation 2 are displayed in Table 9.9

Parameter	Value	Unit
Number of ships (n)	6	
Timesteps (t)	5000	
Frequency (ΔT)	0.1	seconds

Table 9.9.: Parameters for dynamic scene generation in Simulation 2.

Camera Setup

The camera setup in the third simulation involves the placement of two dynamic cameras at the front of a vessel. More specifically, the first camera (*Camera 1*) is positioned on the left side of the vessel, while the second camera (*Camera 2*) is positioned on the right side of the vessel. Both cameras have the same intrinsic parameters as the camera used for creating the images in the synthetic dataset in Chapter 8. The parameters used for the camera setup are described in Table 9.10 and 9.11.

Parameter	Value	Unit
Focal length (f)	23.7	mm
Horizontal pixel size (Δp_1)	0.001	mm
Vertical pixel size (Δp_2)	0.00083	mm
Principle point (a_1, a_2)	(1232, 1028)	px
Image bounds	(2464, 2056)	px
Camera position VCF	(4, 2, 1)	m
Fixed roll (θ_f)	0	rad
Fixed pitch (ϕ_f)	$\pi/200$	rad
Fixed yaw (ψ_f)	$\pi/8$	rad
VesselID	'vehicle1'	

Table 9.10.: Camera setup in Simulation 3 for *Camera 1*.

Parameter	Value	Unit
Focal length (f)	23.7	mm
Horizontal pixel size (Δp_1)	0.001	mm
Vertical pixel size (Δp_2)	0.00083	mm
Principle point (a_1, a_2)	(1232, 1028)	px
Image bounds	(2464, 2056)	px
Camera position VCF	(4, -2, 1)	m
Fixed roll (θ_f)	0	rad
Fixed pitch (ϕ_f)	$\pi/200$	rad
Fixed yaw (ψ_f)	$-\pi/8$	rad
VesselID	'vehicle3'	

Table 9.11.: Camera setup in Simulation 3 for *Camera 2*.

Wave Motion

In the third simulation, the wave impact on the camera is of type *large-wave-impact*. The initial values for the first and second autoregressive process that simulates the wave motion are set to 0 for all angles. A summary of the parameters employed for the wave motion in Simulation 3 is presented in Table 9.12.

Parameter	Value (radians)
a^{roll}	0.99
a^{pitch}	0.99
a^{yaw}	0.99
b^{roll}	0.99
b^{pitch}	0.99
b^{yaw}	0.99
σ^{roll}	0.00020
σ^{pitch}	0.00012
σ^{yaw}	0.00006

Table 9.12.: Wave motion parameters for simulation 3.

9.2. Simulation Results

The visualisations of the resulting MODSIM output after the simulations is presented in this section.

9.2.1. Simulation 1

The visualisations obtained from Simulation 1 are compiled in a YouTube playlist, accessible via scanning Quick-Response (QR) code presented in Figure 9.1, or by following the hyperlink provided in the corresponding reference Jessen and Mohr (2023b). Only timesteps that contain at least one vessel are visualised, as visualising empty frames is uninteresting and uses unnecessary resources.

Figure 9.1.: QR code to access the visualisations of Simulation 1. See Jessen and Mohr (2023c) to access by following a hyperlink.

Figure 9.2 displays timestep 255, corresponding to a simulation duration of 51.0 seconds, extracted from the dynamic scene visualisation of Simulation 1. The red dot denotes the camera's position, while its orientation is visually conveyed through the small red line originating from the dot. The scene is observed from above, looking down on the 2D marine surface. The vessels within the scene are denoted by dark squares, whereas their respective trajectories are visualised as circular lines. Figure 9.3 illustrates the same timeframe extracted from the visualisation of the projected points in Simulation 1. The coloured points in the image correspond to the projected points of the vessels, and the lines between them draw the shoebox of each vessel. Figure 9.4 presents the same timeframe extracted from the visualisation of the detections in Simulation 1. The light grey squares and labels correspond to the ground truth annotations, while the coloured squares and labels represent the detected objects. Additionally, the confidence score of the detection is displayed next to the detected class label. The background colour in Figure 9.3 and Figure 9.4 distinguishes between the ocean and sky, with darker shades representing the former and lighter shades representing the latter. The horizon line serves as the boundary between the sky and the ocean.

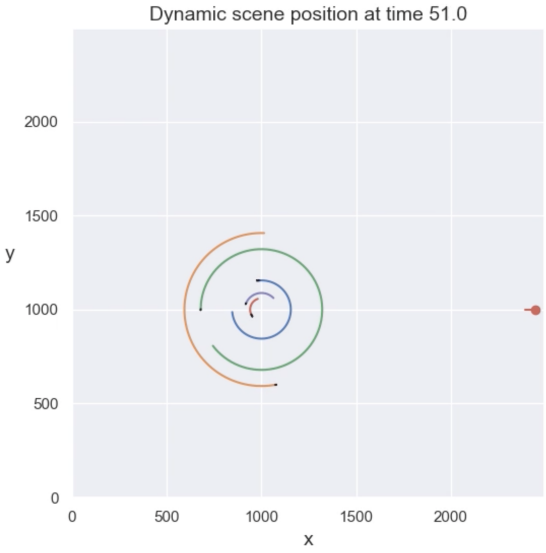


Figure 9.2.: **Simulation 1**: Vessel and camera positions in the dynamic scene at timestep 255 (time 51.0 seconds). A red dot and line denote the camera's position and orientation, while the vessel trajectories are present with circular lines.

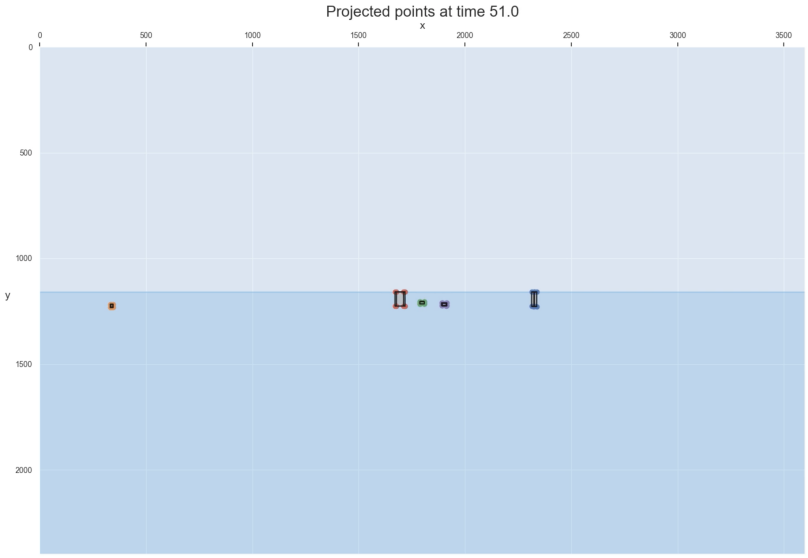


Figure 9.3.: **Simulation 1**: Projected points at timestep 255 (time 51.0 seconds).

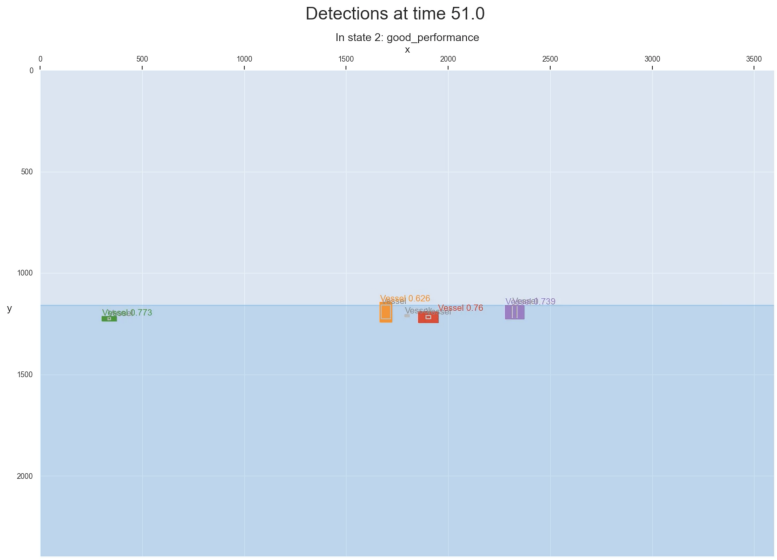


Figure 9.4.: **Simulation 1**: Detections at timestep 255 (time 51.0 seconds). The light grey squares and labels correspond to the ground truth annotations, while the coloured squares and labels represent the detected objects. Additionally, the confidence score of the detection is displayed next to the detected class label.

9.2.2. Simulation 2

The visualisations obtained from Simulation 2 are compiled in a YouTube playlist, accessible via scanning the QR code presented in Figure 9.5, or by following the hyperlink provided in the corresponding reference Jessen and Mohr (2023c). Only timesteps that contain at least one vessel are visualised, as visualising empty frames is uninteresting and uses unnecessary resources.

Figure 9.5.: QR code to access the visualisations of Simulation 2. See Jessen and Mohr (2023c) to access by following a hyperlink.

Figure 9.6 visualises timestep 3899, corresponding to time 389.9 seconds, in the dynamic scene, depicting vessels as squares and their tracks as lines following the vessel. The ownship, where the camera is positioned, is represented by a thicker blue square. The scene is observed from above, looking down on the 2D marine surface. Figure 9.7 showcases the projected points observed from the camera mounted on the boat at the same timestep. The lines between the projected points draw the shoebox of each vessel. Figure 9.8 displays the detections at the same timestep, where the detected vessels are illustrated using coloured squares and coloured labels, while light grey squares and light grey labels represent the ground truth annotations. Additionally, the confidence score of the detection is displayed next to the detected class label. The darker blue colour in both Figure 9.7 and Figure 9.8 indicates the ocean, while the lighter blue indicates the sky. The horizon line serves as the boundary between the sky and the ocean.

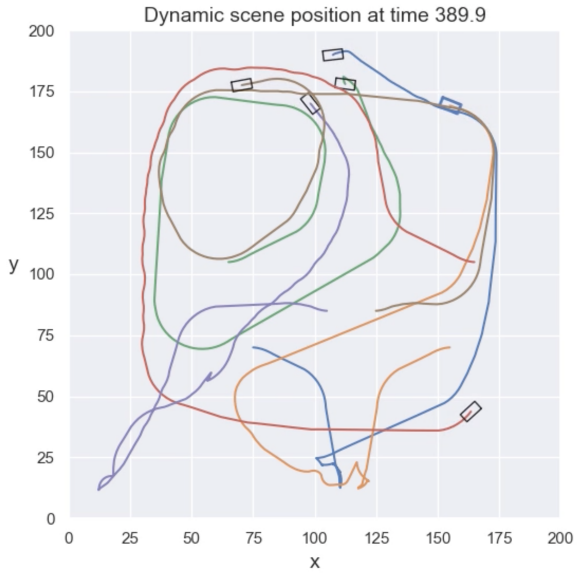


Figure 9.6.: **Simulation 2**: Vessel and camera positions in the dynamic scene at timestep 3899 (time 389.9 seconds). The ownship, where the camera is positioned, is represented by a thicker blue square, while the vessel trajectories are present with circular lines.

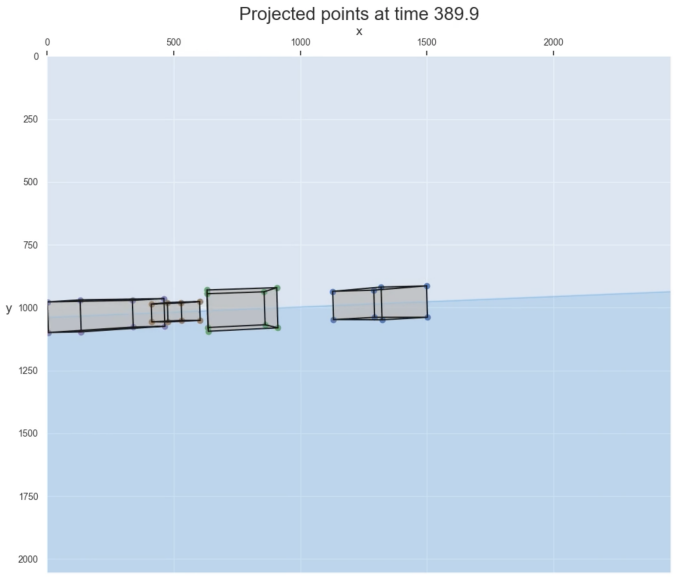


Figure 9.7.: **Simulation 2**: Projected points at timestep 3899 (time 389.9 seconds).

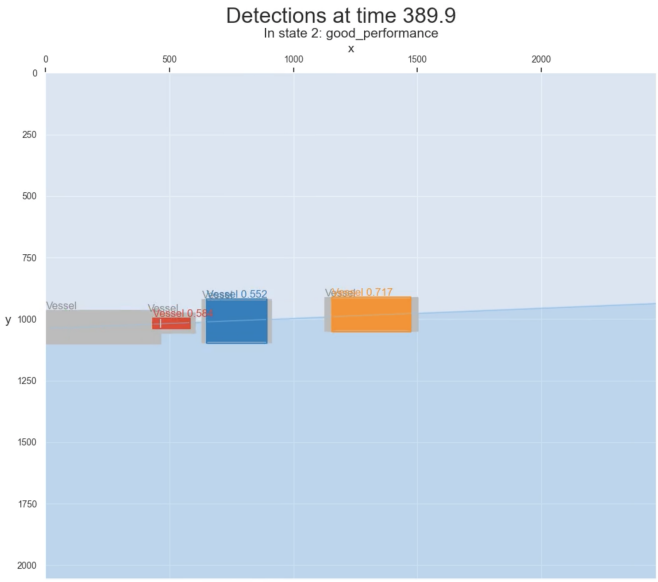


Figure 9.8.: **Simulation 2**: Detections at timestep 3899 (time 389.9 seconds). The light grey squares and labels correspond to the ground truth annotations, while the coloured squares and labels represent the detected objects. The confidence score of the detection is displayed next to the detected class label.

9.2.3. Simulation 3

The visualisations obtained from Simulation 3 are compiled in a YouTube playlist, accessible via scanning the QR code presented in Figure 9.5, or by following the hyperlink provided in the corresponding reference Jessen and Mohr (2023d). Only timesteps that contain at least one vessel are visualised, as visualising empty frames is uninteresting and uses unnecessary resources.

Figure 9.9.: QR code to access the visualisations of Simulation 3. See Jessen and Mohr (2023d) to access by following a hyperlink.

Figure 9.10 displays the dynamic scene, where the vessels are represented by squares, their tracks are illustrated with coloured lines, and the ownship is denoted by a bold blue square. The scene is observed from above, looking down on the 2D marine surface. Figure 9.11 displays the projected points at timestep 2071 (207.1 seconds), with the left image corresponding to the output captured by the left camera (*Camera 1*) and the right image corresponding to the output captured by the right camera (*Camera 2*). The lines between the projected points draw the shoebox of each vessel. Figure 9.12 illustrates the detections at the same timestep, with the left image representing the detections in the image captured by *Camera 1*, and the right image representing the detections in the image captured by *Camera 2*. The light grey squares and labels correspond to the ground truth annotations, while the coloured squares and labels represent the detected

objects. Additionally, the confidence score of the detection is displayed next to the detected class label. Notably, detections are performed independently on each image, leading to the emergence of two distinct detections of the green vessel, as evident from the illustrations.

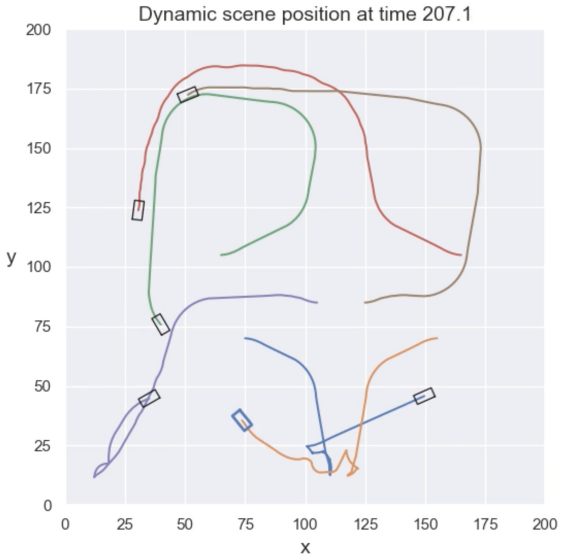


Figure 9.10.: **Simulation 3**: Vessel and camera positions in the dynamic scene at timestep 3899 (time 389.9 seconds). The ownship, where the camera is positioned, is represented by a thicker blue square, while the vessel trajectories are present with circular lines.

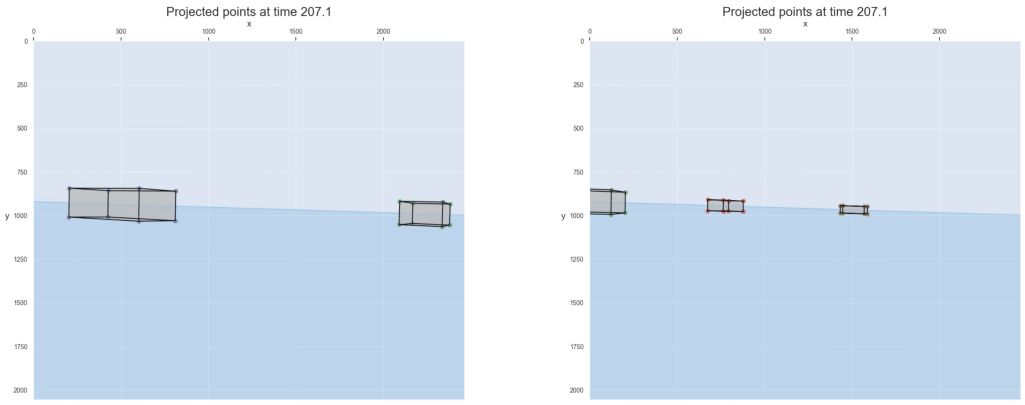


Figure 9.11.: **Simulation 3**: Projected points at timestep 2071 (207.1 seconds). The left image corresponds to the output captured by the left camera (*Camera 1*), and the right image corresponds to the output captured by the right camera (*Camera 2*).

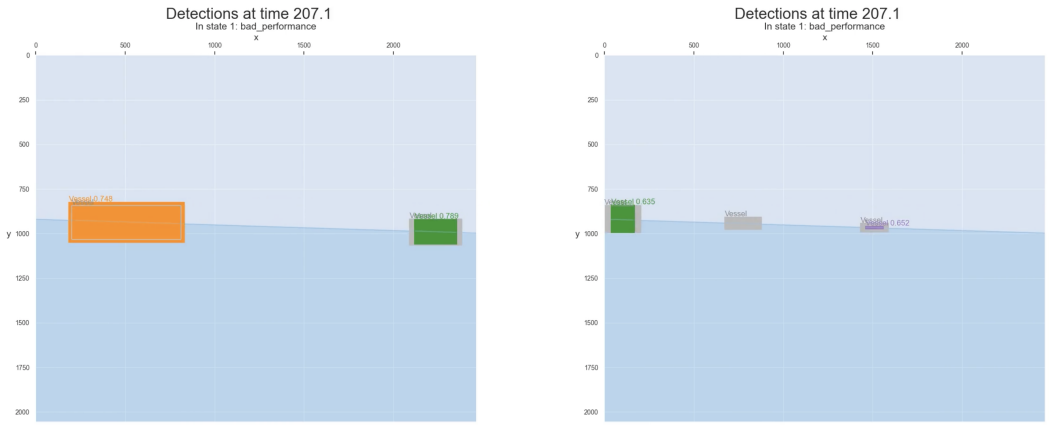


Figure 9.12.: **Simulation 3**: Detections at timestep 2071 (207.1 seconds). The light grey squares and labels correspond to the ground truth annotations, while the coloured squares and labels represent the detected objects. Additionally, the confidence score of the detection is displayed next to the detected class label. The detections in the left image correspond to the detections in the image captured by the left camera (*Camera 1*), and the detections in the right image correspond to the detections in the image captured by the right camera (*Camera 2*).

9.3. Discussion of the Developed Simulator

The simulations showcased the diverse capabilities of the MODSIM system. Firstly, they demonstrated different methods of inputting pose sequences. Simulation 1 exemplifies the direct generation of pose sequences within the dynamic scene, while Simulation 2 and Simulation 3 showcased the use of pre-recorded pose files from an external part. Further, the simulations explored the use of different cameras. Simulation 1 employed a static camera observing the scene from a larger distance, using camera parameters similar to a "simple legacy camera". Simulations 2 and Simulation 3 utilised the camera type used for creating the images for the synthetic dataset described in Chapter 8. Both Simulation 2 and 3 featured a camera mounted on a vessel within the scene, with Simulation 3 highlighting the option of incorporating multiple cameras. In addition, the simulations showcased variations in wave impact on the camera, with Simulation 1 having no wave impact, Simulation 2 including small wave impact, and Simulation 3 having large wave impact. All simulations employ annotation mode 0, which best matches the ground truth annotations provided for the synthetic dataset used to obtain the error statistics. They demonstrated diverse bounding box variations, dropouts, and false positives, reflecting the performance of the object detector trained and evaluated in Chapter 8. These factors varied for different detection states as modelled by the temporal model. Finally, the videos of the simulations illustrated the implemented visualisation included in the MODSIM system.

Detections were generated at each timestep with a frequency of 0.1 and 0.2 seconds, resulting in many variations occurring within a brief timeframe. As subsequent time steps progressed, the overall error gradually converged towards the inputted performance statistics, indicating the system's capability to reflect the provided performance data.

The current simulator incorporates gradual changes in the overall detector performance over time through the temporal model. The system does not account for time dependence at an individual object level. Consequently, annotations in the simulations that are temporally close, in the same performance state, and nearly identical display performance differences. Typically,

detectors tend to detect the same objects when considering almost identical images. Additionally, in cases where an object is initially not detected in an image due to occlusion, the object will likely remain occluded in subsequent time steps. This persistence occurs because the occluding factor requires a certain amount of time to clear away, allowing the detector to regain visibility of the occluded object. These observations indicate the need to incorporate time dependence in performance statistics at an individual vessel level within the MODSIM system.

The bounding box performance varies across the different performance states, and no state demonstrates a superior bounding box performance. Due to imprecise and inconsistent annotations in the dataset used to obtain the statistical parameters for the simulations, it is impossible to determine whether this accurately reflects the detector's performance in the real world under varying detection conditions. The error generator in MODSIM does not consider the actual size of the detected objects when introducing errors in bounding box placement and dimensions. As a result, vessels appearing small in the image may result in detected bounding boxes that significantly deviate from the ground truth. This can be observed in Figure 9.4, where vessels located far away occupy few pixels in the image and exhibits proportionally large variation in the bounding box compared to the ground truth size. Conversely, for vessels appearing large in the image with the same bounding box error distributions displays proportionally smaller variations compared to the true bounding box. This can be observed in Figure 9.8. It may be beneficial to incorporate a correlation between vessel size, distance from the camera, and bounding box variations to address these observations.

The available time and resources imposed limitations on conducting a comprehensive exploration of every aspect of the simulator. As a result, choices and assumptions were made without exploring more detailed approaches to model various aspects of the real world. Moreover, the absence of an evaluation method for the simulator leaves the effectiveness of these choices unverified. When assessing the simulator, it is crucial to consider which enhancements are essential and which ones might impose unnecessary computational resource usage. Although the experiment in Chapter 8 showcase the feasibility of obtaining the necessary parameters, it does not define a practical application of obtaining these parameters. This is a complex and crucial task that extends beyond the scope of this thesis. Further, it is important to note that since the experiments in Chapter 8 are conducted on a synthetic dataset, the resulting error statistics and temporal model may not accurately reflect the detector performance when applied to real data. These experiments serve solely as example parameters for the simulations, and further studies should be conducted to establish how visual object detectors are affected by adverse weather conditions.

In conclusion, the simulations demonstrated the implementation of the MODSIM system, and presented its ability to adapt to defined performance states. The simulator meets the requirements presented in Section 1.3 while being computationally inexpensive. However, the absence of an evaluation of the simulator raises uncertainty regarding the quality of the simulations. Furthermore, the observed outputted detections highlight the possible need for incorporating time dependence in performance statistics at the individual vessel level, as well as incorporating correlation between the vessel's size, distance from the camera, and bounding box variations. The modular design of the MODSIM system facilitates further development and simple integration of enhancements to the different components of the system.

10. Conclusion and Further Work

10.1. Conclusion

This master's thesis makes a contribution to the research field of verification of autonomous vessels, in particular the external SITAW system within the autonomous vessel. The Marine Object Detector Simulator (MODSIM) was developed and implemented to emulate the behaviour of camera-based object detectors on autonomous ships. The simulator efficiently generates test data to assess trackers within the external situational awareness system. A notable advantage of the MODSIM system is its ability to replicate the behaviour of real camera-based object detectors by simulating detector errors through statistical distributions, eliminating the need for complex simulations of underlying causes of error. This approach enhances the efficiency and simplicity of the simulator, as it primarily focuses on reproducing the output behaviour of the detector rather than simulating each step of the detection pipeline. The resulting system presented in this thesis encompasses the entire simulation process, from obtaining pose data to delivering the final output of the simulator. Furthermore, its modular design facilitates easy development and enhancement of each component, adding to the system's versatility and potential for future improvements.

The simulations generated in Chapter 9 showcased the diverse capabilities of the MODSIM system. Three distinct simulations were created, each incorporating different pose data, camera setups, and wave impacts. In Chapter 8, the statistical parameters for the error generator in the MODSIM system were obtained using the YOLOv8 object detector. The detector was trained on one dataset with real-world images and one with synthetic images containing various weather and lighting conditions. The detector's performance was evaluated on separate test sets for each weather and lighting condition. The evaluation results served as a basis for establishing the statistical metrics and distributions for each temporal state in the error generator. The simulations effectively demonstrated that MODSIM fulfils the requirements presented in Section 1.3. Together with the experiment conducted in Chapter 8, they demonstrated the feasibility of acquiring the necessary parameters for error generation and validated the developed system's viability.

The developed system and the research itself had some limitations. Time and resource constraints prevented a comprehensive exploration of every aspect of the simulation. Consequently, several choices and assumptions were made without exploring more in-depth approaches to model the different aspects of the real world. Additionally, the absence of an evaluation method for the simulator left the effectiveness of these choices unproven. There will always be the possibility of modelling aspects in a manner that more accurately reflects the real world. Finding the right balance between accuracy and efficiency requires careful consideration, particularly considering that the main advantage of this approach lies in its efficiency and simplicity. Furthermore, the requirement for error statistics when using the simulator raises the question of how to acquire these parameters. While this thesis has demonstrated the ability to generate simulations efficiently for camera detectors, the investigation of the practical utilisation of this simulator is limited. Although the experiment on learning the parameters for YOLOv8 demonstrates the feasibility of obtaining them, the parameters themselves and the method employed must be thor-

oughly evaluated. The acquisition of error statistics for practical applications presents important considerations and challenges that this thesis does not touch upon.

In summary, this thesis successfully implements the MODSIM, a simulator that efficiently replicates the behaviour of camera-based object detectors deployed in autonomous ships. The simulator generates test data to evaluate the performance of trackers integrated within the external situational awareness system, offering a computationally efficient assessment method. Further explorations into various aspects of the simulator should be conducted to enhance the realism of the generated errors and to ensure a comprehensive evaluation of the simulations' effectiveness.

10.2. Further Work

The main focus of this thesis has been to design and develop the complete Marine Object Detector Simulator (MODSIM) system. Since conducting a detailed examination of each component is not within the scope of this thesis, certain simplifications were made. Throughout the project, various ideas for expanding the system have emerged, which require further research and potential implementation. This chapter presents potential improvements for the system.

10.2.1. Evaluation of the Simulator

In order to advance the development and research of the MODSIM system, it becomes crucial to assess the strengths and weaknesses of the employed modelling approaches. Integrating methods for evaluating the simulator stands as an important future task that was not included in the scope of this thesis. As a result, the simplifications and assumptions made in the MODSIM system have not undergone evaluation, rendering it impossible to determine the adequacy of the applied methods. By evaluating the simulations, it becomes possible to identify which components require further development. It is essential to be able to determine the level of detail necessary for the detector representation and discover which enhancements consume unnecessary computing power. It is important that the simulations achieve a level of accuracy sufficient for tracker assessment purposes. Furthermore, performing a comprehensive analysis of the simulator's time and resource requirements and comparing them to those of a highly accurate simulator could prove beneficial, as the main reason for developing such a computationally inexpensive simulator is to be able to produce test data faster than other simulators that currently are available. Finally, the simulated detections must be streamed into a tracker to assess the tracker's response when processing simulated detection inputs.

10.2.2. Temporal Model

Further research should be conducted to define a more complex and detailed temporal model for the error generator. Characterising and quantifying how the performance of an object detector varies over time, along with the task of developing a suitable modelling approach, pose challenges that demand extensive time and effort. Consequently, these aspects have not been the primary focus of this thesis, and instead, a simplified temporal model has been devised to illustrate the intended purpose and impact of such a model. Evaluating the current level of detail in the existing temporal model can provide insights into whether the existing states adequately capture the detector's behaviour.

Several factors have arisen during this thesis that needs to be considered when further developing the temporal model. Firstly, an important aspect to address is the number of states needed in

the simulator. Determining the appropriate number of states depends on the desired level of detail and what variations in detector performance are desirable to test on the tracker. Having too few states may oversimplify the system, potentially leading to inaccurate results and a limited understanding of its behaviour. On the other hand, having an excessively large number of states may introduce unnecessary complexity and computational overhead without significantly improving the simulation accuracy.

In the existing temporal model, each state is associated with a specific bounding box error distribution for the centre, height, and width. In Chapter 8, an experiment was conducted to explore the possibility of learning the error parameters from a real detector. It was observed that no weather or light condition clearly led to an increase or decrease in errors in the dimensions and placements of the bounding box. However, the experiment had limitations, particularly with imprecise bounding box annotations. Therefore, it is uncertain if these findings apply to detectors in general. Further research is required to determine whether adverse operating conditions affect bounding box errors or if other factors influence the magnitude of errors.

10.2.3. Acquiring the Required Error Statistics

The inclusion of error statistics as a requirement for simulator usage raises the question of how to acquire these parameters. While this thesis has demonstrated the ability to generate simulations efficiently for camera detectors, the practical utilisation of this simulator requires further research. Several questions should be addressed. Firstly, should the simulator aim to replicate the behaviour of a specific real object detector, with parameters changing for each simulated detector? Alternatively, should the simulator reflect general detector performance and then test the tracker's response to varying levels of detector performance without necessarily replicating the exact detector utilised? This allows for the design of an error generator that examines the tracker's behaviour during specific performance drops, such as a sudden increase in the dropout rate. If obtaining the error statistics for each simulated detector is necessary, it is crucial to establish a reliable and standardised method for acquiring these parameters. For example, should the simulator user, such as DNV, define a dataset encompassing various conditions for testing the detector and providing the required statistics? Alternatively, should they instruct the detector owner on how to test their detector under specific predefined conditions?

Further, the temporal model in the error generator requires a transition matrix defining the probabilities of transitioning to another state, given the current state. Quantifying this matrix poses a challenging task that demands careful deliberation. Further, it is important to consider whether the transitions should mirror the gradual environmental changes observed in the real world or simulate more rapid changes to allow for a more efficient assessment of the tracker.

10.2.4. A More In-depth Analysis of Detector Performance

To obtain a comprehensive understanding of detector performance, particularly under challenging operational conditions, it is essential to conduct an in-depth analysis. This analysis should consider multiple factors that can influence detector performance. The findings should be used to improve the error generator in MODSIM, ultimately enhancing the realism and accuracy of the simulator. Several extension ideas for the error generator have emerged during this project and warrant further research and potential implementation.

First of all, the current iteration of the MODSIM system uses an estimation method for determining the number of false positive detections for each image. This approach is based on intuition and is not evaluated. It is necessary to explore alternative approaches that accurately

define the occurrence and distribution of false positives in the simulation. Additionally, the confidence scores are determined from a normal distribution, which is only utilised to introduce the concept of confidence scores. Further consideration of the method to assign confidence scores is needed.

Although the current simulator incorporates time dependency through the temporal model, this only captures slow changes in general detector performance over time. Further exploration should be conducted considering the time-dependent behaviour of individual objects. By accounting for the time-dependent aspects of object presence and absence, the simulator can better reflect the complexities of real-world scenarios. For instance, if a vessel is not detected due to occlusion by a cloud, a wave or other occluding factors, it will likely remain occluded and undetected in subsequent time steps. Likewise, if an object floating on the water is initially detected as a boat, it is more likely to be classified as a boat in the following time frames. This time-dependent behaviour should be analysed in actual detectors. It should be noted that this kind of time-dependent behaviour imposes a greater challenge for the tracker, making it desirable to incorporate in the simulator.

Further, the dropout rate and false positive rate should increase with distance. Objects farther away are more challenging to detect since they generally occupy fewer pixels due to their smaller size. In addition, adverse conditions, such as fog and rain, will enhance this distance-dependent behaviour. Particularly, fog significantly reduces visibility over larger distances. This can be modelled, for instance, using a linear function. Incorporating distance-dependent linearity into the simulator can enhance realism and capture the difficulties associated with foggy conditions.

Evaluating the relationship between dropout (missed detections) and object visibility can provide valuable insights into detector behaviour. If less of the object is visible, it is more likely that the detector will struggle to detect it. However, further investigation is necessary to explore this relationship in depth.

Lastly, the bounding box variations are equal regardless of the number of pixels the vessel occupies in the image. As a result, vessels of smaller sizes or those located further away from the camera exhibit significantly larger bounding box variations relative to their ground truth bounding boxes. To address this issue, exploring the correlation between the vessel's size and distance from the camera and the bounding box variations is desirable.

The possible improvements of the error generator discovered during the development of the MODSIM system deserve careful exploration and consideration. Furthermore, it is important to acknowledge that there may be numerous additional improvements that can be implemented in the error generator.

10.2.5. Dynamic Scene Generator

To enhance the realism and diversity of the simulator, it is important to incorporate desirable information in the dynamic scene. The information included in the current iteration of the simulator is sufficient for simulating detections. However, by expanding the capabilities of the scene generator, the simulator can more effectively simulate complex environments and evaluate the performance of tracking systems under various conditions.

Currently, the only 3D objects in the scene are the vessels. However, other objects may be desirable to represent in the scene, including land or other objects that may obscure the vessels or produce false detections. Further, incorporating other sensor information, such as IMU data, can enhance the realism of the simulator. IMU data provides measurements of vessel motion and orientation, including the motion caused by waves. By incorporating this data, the simulator

can effectively simulate realistic vessel dynamics. Exploring other potential data inputs for the simulator is also worth researching.

Incorporating efficient integration of pre-recorded real-world pose sequences into the simulator is a simple method for generating realistic vessel traffic scenarios. Real-world pose sequences can be obtained from, for instance, Automatic Identification Systems (AIS), Global Navigation Satellite Systems (GNSS) and IMU, which provides relevant details about vessels, such as vessel positions, speeds etc. The MODSIM system should be able to input pose data originating from these sources.

While the current simulator employs simplified shoe box descriptors for object representations, it can be beneficial to expand the variety of vessel representations, for instance, by allowing for more boat-like vehicle descriptors.

10.2.6. Wave Motion

The incorporated wave motion model is a simplistic approach, creating a rotational motion that partially reflects the motion generated by waves. The wave motion is incorporated on the ownship and is modelled to represent the waves' impact on the camera, not the amount and type of waves in the scene. However, to improve the realism of the simulator, a more realistic and complex wave model can be explored and incorporated.

10.2.7. Including Other Sensors

The simulator currently only simulates camera detections, considering the integration of other sensors can enhance the simulator's applicability. By expanding the range of sensors incorporated within the simulator, the simulation of multi-sensor systems becomes possible. This advancement makes the simulator more valuable for testing tracker systems. Section 2.1 elaborates on typical sensors used in autonomous vessels. These are examples of the types of sensors that could be included in the simulator.

10.2.8. Output Format of the Detector

The current simulator outputs detections in the format described in Section 7.7, which is a standard output format for visual object detectors. However, there are some other outputs that could be desirable to include. For instance, some detectors estimate the depth of the detected object by using stereo cameras, and some detectors output a segmentation mask instead of an AABB.

Bibliography

Ahrnbom, M. (2022). *Computer Vision for Automated Traffic Safety Assessment: A Machine Learning Approach*. PhD thesis, Mathematics (Faculty of Engineering).

AILiveSim (2023). Ailivesim. `https://www.ailivesim.com/`. Accessed: 05/23/2023.

Al-Haija, Q. A., Gharaibeh, M., and Odeh, A. (2022). Detection in adverse weather conditions for autonomous vehicles via deep learning. *AI*, 3(2):303–317.

Bernuth, A. v., Volk, G., and Bringmann, O. (2019). Simulating photo-realistic snow and fog on existing images for enhanced cnn training and evaluation. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 41–46.

Brekke, E. F., Eide, E., Eriksen, B.-O. H., Wilthil, E. F., Breivik, M., Skjellaug, E., Helgesen, Ø. K., Lekkas, A. M., Martinsen, A. B., Thyri, E. H., Torben, T., Veitch, E., Alsos, O. A., and Johansen, T. A. (2022). milliampere: An autonomous ferry prototype. *Journal of Physics: Conference Series*, 2311(1):012029.

Chaturvedi, S. S., Zhang, L., and Yuan, X. (2022). Pay "attention" to adverse weather: Weather-aware attention-based object detection.

Chen, Y., Li, W., Sakaridis, C., Dai, D., and Gool, L. V. (2018). Domain adaptive faster R-CNN for object detection in the wild. *CoRR*, abs/1803.03243.

DNV GL (2015). Revolt main report. *Technical Report 2015-0170*.

Dosovitskiy, A., Ros, G., Codevilla, F., López, A. M., and Koltun, V. (2017). CARLA: an open urban driving simulator. *CoRR*, abs/1711.03938.

Endsley, M. (1995). Toward a theory of situation awareness in dynamic systems. human factors journal 37(1), 32-64. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 37:32–64.

Fattal, R. (2008). Single image dehazing. *ACM Trans. Graph.*, 27(3):1–9.

Girshick, R. B., Donahue, J., Darrell, T., and Malik, J. (2013). Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524.

Hartley, R. and Zisserman, A. (2003). *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2 edition.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. B. (2017). Mask R-CNN. *CoRR*, abs/1703.06870.

He, K., Sun, J., and Tang, X. (2009). Single image haze removal using dark channel prior. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1956–1963.

Helgesen, Ø. K., Vasstein, K., Brekke, E. F., and Stahl, A. (2022). Heterogeneous multi-sensor tracking for an autonomous surface vehicle in a littoral environment. *Ocean Engineering*.

Hnewa, M. and Radha, H. (2021). Multiscale domain adaptive YOLO for cross-domain object detection. *CoRR*, abs/2106.01483.

Jessen, A. S. and Mohr, S. J. (2023a). Modsim: an efficient marine object detection simulator. `https://github.com/andreajessen/MODSIM`.

Jessen, A. S. and Mohr, S. J. (2023b). Modsim simulation 1: Dynamic scene. `https://www.youtube.com/playlist?list=PLpBpNY4MnC3900WwnMeN04diavKELxrFn`.

Jessen, A. S. and Mohr, S. J. (2023c). Modsim simulation 2: Dynamic scene. `https://www.youtube.com/playlist?list=PLpBpNY4MnC3_0FGgSr0nLtRk_sgNSDGEa`.

Jessen, A. S. and Mohr, S. J. (2023d). Modsim simulation 3: Dynamic scene. `https://www.youtube.com/playlist?list=PLpBpNY4MnC3_RbQl5cBEWahHkpcUTPX5p`.

Jocher, G., Chaurasia, A., and Qiu, J. (2023). Yolo by ultralytics. `https://github.com/ultralytics/ultralytics`. Version: 8.0.0, Accessed: 2023-05-23.

Kenk, M. A. and Hassaballah, M. (2020). DAWN: vehicle detection in adverse weather nature dataset. *CoRR*, abs/2008.05402.

Kongsberg Maritime (2023). Autonomous ship project, key facts about yara birkeland. `https://www.kongsberg.com/maritime/support/themes/autonomous-ship-project-key-facts-about-yara-birkeland/`. Accessed: 05/08/2023.

Kornev, N. (2012). Ship dynamics in waves (ship theory ii). Technical report, Universität Rostock.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S. E., Fu, C., and Berg, A. C. (2015). SSD: single shot multibox detector. *CoRR*, abs/1512.02325.

Liu, W., Ren, G., Yu, R., Guo, S., Zhu, J., and Zhang, L. (2022). Image-adaptive yolo for object detection in adverse weather conditions.

MathWorks (2017). Generate vision detections for driving scenario. `https://se.mathworks.com/help/driving/ref/visiondetectiongenerator-system-object.html#bvn7au5-6`.

Mester, R. (2022). Statistical pattern recognition.

Narasimhan, S. and Nayar, S. (2002). Vision and the atmosphere. *International Journal of Computer Vision*, 48:233–254.

NRK (2011). Hurtigruten minutt for minutt. `https://tv.nrk.no/serie/hurtigruten-minutt-for-minutt`.

NTNU (2023). Autoferry. `https://www.ntnu.edu/autoferry`. Accessed: 05/08/2023.

Padilla, R., Netto, S. L., and Da Silva, E. A. (2020). A survey on performance metrics for object-detection algorithms. In *2020 international conference on systems, signals and image processing (IWSSIP)*, pages 237–242. IEEE.

Redmon, J., Divvala, S. K., Girshick, R. B., and Farhadi, A. (2015). You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640.

Ren, S., He, K., Girshick, R. B., and Sun, J. (2015). Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497.

Rijsbergen, C. J. V. (1979). *Information Retrieval*. Butterworth-Heinemann, 2nd edition.

Rong, G., Shin, B. H., Tabatabaee, H., Lu, Q., Lemke, S., Mozeiko, M., Boise, E., Uhm, G., Gerow, M., Mehta, S., Agafonov, E., Kim, T. H., Sterner, E., Ushiroda, K., Reyes, M., Zelenkovsky, D., and Kim, S. (2020). LGSVL simulator: A high fidelity simulator for autonomous driving. *CoRR*, abs/2005.03778.

Rothmeier, T. and Huber, W. (2021). Performance evaluation of object detection algorithms under adverse weather conditions. In Martins, A. L., Ferreira, J. C., Kocian, A., and Costa, V., editors, *Intelligent Transport Systems, From Research and Development to the Market Uptake*, pages 211–222, Cham. Springer International Publishing.

Sasaki, Y. (2007). The truth of the f-measure. *Teach Tutor Mater*.

Schöller, F. E. T. (2022). *Camera-based Perception for Autonomous Vessels at Sea*. PhD thesis, Technical University of Denmark.

Shah, S., Dey, D., Lovett, C., and Kapoor, A. (2017). Airsim: High-fidelity visual and physical simulation for autonomous vehicles. *CoRR*, abs/1705.05065.

Sindagi, V. A., Oza, P., Yasarla, R., and Patel, V. M. (2019a). Prior-based domain adaptive object detection for adverse weather conditions. *CoRR*, abs/1912.00070.

Sindagi, V. A., Yasarla, R., and Patel, V. M. (2019b). Pushing the frontiers of unconstrained crowd counting: New dataset and benchmark method. *CoRR*, abs/1910.12384.

Skoglund, U. (2018). Førerløse ferger kan erstatte gangbruer. `https://gemini.no/2018/06/forerlose-ferger-kan-erstatte-gangbruer/`.

Vasstein, K. (2021). A high fidelity digital twin framework for testing exteroceptive perception of autonomous vessels. Master's thesis, Norwegian University of Science and Technology.

Vasstein, K., Brekke, E. F., Mester, R., and Eide, E. (2020). Autoferry gemini: a real-time simulation platform for electromagnetic radiation sensors on autonomous ships. *IOP Conference Series: Materials Science and Engineering*, 929(1):012032.

Volk, G., Müller, S., Bernuth, A. v., Hospach, D., and Bringmann, O. (2019). Towards robust cnn-based object detection through augmentation with synthetic rain variations. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 285–292.

Wilthil, E. F. (2019). *Maritime Target Tracking with Varying Sensor Performance*. PhD thesis, Norwegian University of Science and Technology.

Zang, S., Ding, M., Smith, D., Tyler, P., Rakotoarivelo, T., and Kaafar, M. A. (2019). The impact of adverse weather conditions on autonomous vehicles: How rain, snow, fog, and hail affect the performance of a self-driving car. *IEEE Vehicular Technology Magazine*, 14(2):103–111.

Zeabuz (2023). Zeabuz. `https://www.zeabuz.com/`. Accessed: 05/08/2023.

Zhang, H. and Patel, V. M. (2018a). Densely connected pyramid dehazing network. *CoRR*, abs/1803.08396.

Zhang, H. and Patel, V. M. (2018b). Density-aware single image de-raining using a multi-stream dense network. *CoRR*, abs/1802.07412.

Zhang, H., Sindagi, V., and Patel, V. M. (2017). Image de-raining using a conditional generative adversarial network. *CoRR*, abs/1701.05957.

Appendices

A. Results of Experimental Bounding Box Errors

This appendix presents the mean vector and covariance matrix of the elements of the bounding box error vectors resulting from the experiments described in Chapter 8. Table A.1 displays the results from Experiment 1, Table A.2 displays the results from Experiment 2, Table A.3 and Table A.4 displays the results from Experiment 3.

Detection condition	**Mean vector**	**Covariance matrix**
Real-world	$\begin{matrix} e_x \\ e_y \\ e_w \\ e_h \end{matrix} \begin{bmatrix} -0.1 \\ -0.9 \\ 0.1 \\ 0.0 \end{bmatrix}$	$\begin{matrix} \\ e_x \\ e_y \\ e_w \\ e_h \end{matrix} \begin{matrix} \begin{matrix} e_x & e_y & e_w & e_h \end{matrix} \\ \begin{bmatrix} 13.1 & 1.6 & 2.7 & -2.4 \\ 1.6 & 12.2 & -0.2 & -13.9 \\ 2.7 & -0.2 & 51.5 & 7.8 \\ -2.4 & -13.9 & 7.8 & 59.4 \end{bmatrix} \end{matrix}$

Table A.1.: **Experiment 1**: Mean vector and covariance matrix of the bounding box error vectors in pixels (px) for real-world conditions (IoU threshold: 0.5). e_x and e_y is the error in the placement of the bounding box centre, e_h is the error in bounding box width, and e_w is the error in height.

Detection condition	**Mean vector**	**Covariance matrix**
Noon clear	$\begin{matrix} e_x \\ e_y \\ e_w \\ e_h \end{matrix} \begin{bmatrix} 4.4 \\ 8.2 \\ -5.4 \\ 13.3 \end{bmatrix}$	$\begin{matrix} \\ e_x \\ e_y \\ e_w \\ e_h \end{matrix} \begin{matrix} \begin{matrix} e_x & e_y & e_w & e_h \end{matrix} \\ \begin{bmatrix} 14.7 & -7.2 & 21.8 & -9.4 \\ -7.2 & 149.0 & -76.8 & 158.4 \\ 21.8 & -76.8 & 105.7 & -67.4 \\ -9.4 & 158.4 & -67.4 & 1236.2 \end{bmatrix} \end{matrix}$

Table A.2.: **Experiment 2**: Mean vector and covariance matrix of the bounding box error vectors in pixels (px) for "Noon clear" condition (IoU threshold: 0.5). e_x and e_y is the error in the placement of the bounding box centre, e_h is the error in bounding box width, and e_w is the error in height.

Detection condition	Mean vector	Covariance matrix
Noon clear	$\begin{bmatrix} e_x \\ e_y \\ e_w \\ e_h \end{bmatrix} \begin{bmatrix} 4.3 \\ 6.8 \\ -8.1 \\ -0.9 \end{bmatrix}$	$\begin{matrix} \\ e_x \\ e_y \\ e_w \\ e_h \end{matrix} \begin{matrix} \begin{matrix} e_x & e_y & e_w & e_h \end{matrix} \\ \begin{bmatrix} 2354.1 & 206.3 & 1157.6 & -169.6 \\ 206.3 & 390.4 & -227.8 & -389.5 \\ 1157.6 & -227.8 & 5404.7 & 115.4 \\ -169.6 & -389.5 & 115.4 & 1223.0 \end{bmatrix} \end{matrix}$
Afternoon clear	$\begin{bmatrix} e_x \\ e_y \\ e_w \\ e_h \end{bmatrix} \begin{bmatrix} -1.5 \\ 5.3 \\ -7.7 \\ 0.9 \end{bmatrix}$	$\begin{matrix} \\ e_x \\ e_y \\ e_w \\ e_h \end{matrix} \begin{matrix} \begin{matrix} e_x & e_y & e_w & e_h \end{matrix} \\ \begin{bmatrix} 460.8 & -79.6 & -587.7 & -225.4 \\ -79.6 & 366.0 & -425.3 & 71.1 \\ -587.7 & -425.3 & 1951.3 & 30.2 \\ -225.4 & 71.1 & 30.2 & 502.1 \end{bmatrix} \end{matrix}$
Evening clear	$\begin{bmatrix} e_x \\ e_y \\ e_w \\ e_h \end{bmatrix} \begin{bmatrix} 3.7 \\ 3.1 \\ -6.8 \\ -1.3 \end{bmatrix}$	$\begin{matrix} \\ e_x \\ e_y \\ e_w \\ e_h \end{matrix} \begin{matrix} \begin{matrix} e_x & e_y & e_w & e_h \end{matrix} \\ \begin{bmatrix} 456.6 & 98.9 & 32.9 & -131.7 \\ 98.9 & 158.8 & -299.1 & 6.4 \\ 32.9 & -299.1 & 1687.5 & 175.4 \\ -131.7 & 6.4 & 175.4 & 518.1 \end{bmatrix} \end{matrix}$
Night clear	$\begin{bmatrix} e_x \\ e_y \\ e_w \\ e_h \end{bmatrix} \begin{bmatrix} 10.3 \\ 9.4 \\ -1.2 \\ -3.0 \end{bmatrix}$	$\begin{matrix} \\ e_x \\ e_y \\ e_w \\ e_h \end{matrix} \begin{matrix} \begin{matrix} e_x & e_y & e_w & e_h \end{matrix} \\ \begin{bmatrix} 3957.8 & -13.9 & 3578.7 & 367.9 \\ -13.9 & 493.4 & -265.1 & -24.3 \\ 3578.7 & -265.1 & 10666.5 & 461.1 \\ 367.9 & -24.3 & 461.1 & 702.3 \end{bmatrix} \end{matrix}$
Noon cloudy	$\begin{bmatrix} e_x \\ e_y \\ e_w \\ e_h \end{bmatrix} \begin{bmatrix} 1.4 \\ 3.9 \\ -3.0 \\ 2.8 \end{bmatrix}$	$\begin{matrix} \\ e_x \\ e_y \\ e_w \\ e_h \end{matrix} \begin{matrix} \begin{matrix} e_x & e_y & e_w & e_h \end{matrix} \\ \begin{bmatrix} 702.8 & 65.8 & -350.2 & -157.8 \\ 65.8 & 163.8 & -189.3 & 173.9 \\ -350.2 & -189.3 & 2041.0 & -108.4 \\ -157.8 & 173.9 & -108.4 & 650.2 \end{bmatrix} \end{matrix}$
Afternoon cloudy	$\begin{bmatrix} e_x \\ e_y \\ e_w \\ e_h \end{bmatrix} \begin{bmatrix} 0.1 \\ 9.2 \\ -10.6 \\ -5.6 \end{bmatrix}$	$\begin{matrix} \\ e_x \\ e_y \\ e_w \\ e_h \end{matrix} \begin{matrix} \begin{matrix} e_x & e_y & e_w & e_h \end{matrix} \\ \begin{bmatrix} 2486.0 & -47.8 & -169.7 & -336.6 \\ -47.8 & 490.4 & -307.3 & -318.9 \\ -169.7 & -307.3 & 4682.5 & 241.0 \\ -336.6 & -318.9 & 241.0 & 1284.3 \end{bmatrix} \end{matrix}$
Evening cloudy	$\begin{bmatrix} e_x \\ e_y \\ e_w \\ e_h \end{bmatrix} \begin{bmatrix} 5.2 \\ 4.6 \\ -12.6 \\ -3.4 \end{bmatrix}$	$\begin{matrix} \\ e_x \\ e_y \\ e_w \\ e_h \end{matrix} \begin{matrix} \begin{matrix} e_x & e_y & e_w & e_h \end{matrix} \\ \begin{bmatrix} 1016.6 & 112.2 & 637.4 & -210.5 \\ 112.2 & 171.5 & -323.0 & -99.8 \\ 637.4 & -323.0 & 2570.4 & 123.4 \\ -210.5 & -99.8 & 123.4 & 388.8 \end{bmatrix} \end{matrix}$

Table A.3.: **Experiment 3**: Mean vector and covariance matrix of the bounding box error vectors in pixels (px) for Individual Conditions (IoU threshold: 0.5).

Detection condition	Mean vector	Covariance matrix
Night cloudy	$\begin{matrix} e_x \\ e_y \\ e_w \\ e_h \end{matrix}\begin{bmatrix} 2.9 \\ 8.3 \\ -0.2 \\ -5.4 \end{bmatrix}$	$\begin{matrix} \\ e_x \\ e_y \\ e_w \\ e_h \end{matrix}\begin{matrix} \begin{matrix} e_x & e_y & e_w & e_h \end{matrix} \\ \begin{bmatrix} 3558.3 & -1.7 & 1747.2 & 195.2 \\ -1.7 & 464.6 & -212.6 & -258.1 \\ 1747.2 & -212.6 & 9760.5 & 267.5 \\ 195.2 & -258.1 & 267.5 & 878.5 \end{bmatrix} \end{matrix}$
Noon cloudy rain	$\begin{matrix} e_x \\ e_y \\ e_w \\ e_h \end{matrix}\begin{bmatrix} -4.0 \\ 3.5 \\ -3.2 \\ 2.9 \end{bmatrix}$	$\begin{matrix} \\ e_x \\ e_y \\ e_w \\ e_h \end{matrix}\begin{matrix} \begin{matrix} e_x & e_y & e_w & e_h \end{matrix} \\ \begin{bmatrix} 1918.8 & 112.6 & -1997.3 & -165.5 \\ 112.6 & 218.1 & -212.8 & -256.1 \\ -1997.3 & -212.8 & 4137.5 & -93.2 \\ -165.5 & -256.1 & -93.2 & 728.8 \end{bmatrix} \end{matrix}$
Afternoon cloudy rain	$\begin{matrix} e_x \\ e_y \\ e_w \\ e_h \end{matrix}\begin{bmatrix} -1.1 \\ 8.3 \\ -10.6 \\ -1.9 \end{bmatrix}$	$\begin{matrix} \\ e_x \\ e_y \\ e_w \\ e_h \end{matrix}\begin{matrix} \begin{matrix} e_x & e_y & e_w & e_h \end{matrix} \\ \begin{bmatrix} 2990.1 & 272.7 & -1822.6 & -849.6 \\ 272.7 & 476.4 & -468.3 & -378.4 \\ -1822.6 & -468.3 & 6448.5 & 102.2 \\ -849.6 & -378.4 & 102.2 & 1593.4 \end{bmatrix} \end{matrix}$
Evening cloudy rain	$\begin{matrix} e_x \\ e_y \\ e_w \\ e_h \end{matrix}\begin{bmatrix} 2.5 \\ 6.0 \\ -10.0 \\ -2.4 \end{bmatrix}$	$\begin{matrix} \\ e_x \\ e_y \\ e_w \\ e_h \end{matrix}\begin{matrix} \begin{matrix} e_x & e_y & e_w & e_h \end{matrix} \\ \begin{bmatrix} 1052.4 & -41.4 & 232.7 & -62.5 \\ -41.4 & 402.2 & -474.7 & 83.8 \\ 232.7 & -474.7 & 3321.8 & -141.5 \\ -62.5 & 83.8 & -141.5 & 518.0 \end{bmatrix} \end{matrix}$
Foggy clear	$\begin{matrix} e_x \\ e_y \\ e_w \\ e_h \end{matrix}\begin{bmatrix} 5.3 \\ 0.2 \\ 7.8 \\ 2.6 \end{bmatrix}$	$\begin{matrix} \\ e_x \\ e_y \\ e_w \\ e_h \end{matrix}\begin{matrix} \begin{matrix} e_x & e_y & e_w & e_h \end{matrix} \\ \begin{bmatrix} 2276.1 & 137.6 & 2883.3 & -99.1 \\ 137.6 & 171.0 & 90.1 & 55.3 \\ 2883.3 & 90.1 & 4326.0 & 120.8 \\ -99.1 & 55.3 & 120.8 & 557.8 \end{bmatrix} \end{matrix}$
Foggy cloudy	$\begin{matrix} e_x \\ e_y \\ e_w \\ e_h \end{matrix}\begin{bmatrix} 4.0 \\ 8.2 \\ -24.1 \\ -8.9 \end{bmatrix}$	$\begin{matrix} \\ e_x \\ e_y \\ e_w \\ e_h \end{matrix}\begin{matrix} \begin{matrix} e_x & e_y & e_w & e_h \end{matrix} \\ \begin{bmatrix} 5483.1 & -139.8 & 2526.8 & 23.9 \\ -139.8 & 458.6 & -1273.2 & -91.5 \\ 2526.8 & -1273.2 & 7735.0 & 186.0 \\ 23.9 & -91.5 & 186.0 & 2031.7 \end{bmatrix} \end{matrix}$
Rainy clear	$\begin{matrix} e_x \\ e_y \\ e_w \\ e_h \end{matrix}\begin{bmatrix} -5.8 \\ 2.1 \\ -5.4 \\ -0.9 \end{bmatrix}$	$\begin{matrix} \\ e_x \\ e_y \\ e_w \\ e_h \end{matrix}\begin{matrix} \begin{matrix} e_x & e_y & e_w & e_h \end{matrix} \\ \begin{bmatrix} 4070.9 & -77.2 & 2079.4 & -79.3 \\ -77.2 & 313.9 & -527.9 & 40.6 \\ 2079.4 & -527.9 & 4763.7 & -9.1 \\ -79.3 & 40.6 & -9.1 & 401.8 \end{bmatrix} \end{matrix}$
Rainy cloudy	$\begin{matrix} e_x \\ e_y \\ e_w \\ e_h \end{matrix}\begin{bmatrix} -2.5 \\ 5.6 \\ -4.7 \\ 3.2 \end{bmatrix}$	$\begin{matrix} \\ e_x \\ e_y \\ e_w \\ e_h \end{matrix}\begin{matrix} \begin{matrix} e_x & e_y & e_w & e_h \end{matrix} \\ \begin{bmatrix} 2417.8 & -6.5 & 1795.7 & -78.1 \\ -6.5 & 464.9 & -526.5 & -59.2 \\ 1795.7 & -526.5 & 4057.8 & 7.3 \\ -78.1 & -59.2 & 7.3 & 872.3 \end{bmatrix} \end{matrix}$

Table A.4.: **Experiment 3**: Mean vector and covariance matrix of the bounding box error vectors in pixels (IoU threshold: 0.5).

B. Calculation of Extrinsic Parameters for the Camera Setup in Simulation 1

The camera setup in the first simulation consists of one static "simple legacy camera". The camera is placed at a distance, d, so the entire circle with the maximum allowed radius of the circular motion model is captured by the camera, as illustrated in figure B.1. By placing the camera at a distance d from the centre in the positive direction, and parallel to the w_1-axis the only rotation of the camera necessary is a yaw rotation by ψ (rotation about the second unit vector of the CCF $\vec{c}_2$) and pitch (rotation about the first unit vector of the CCF $\vec{c}_1$).

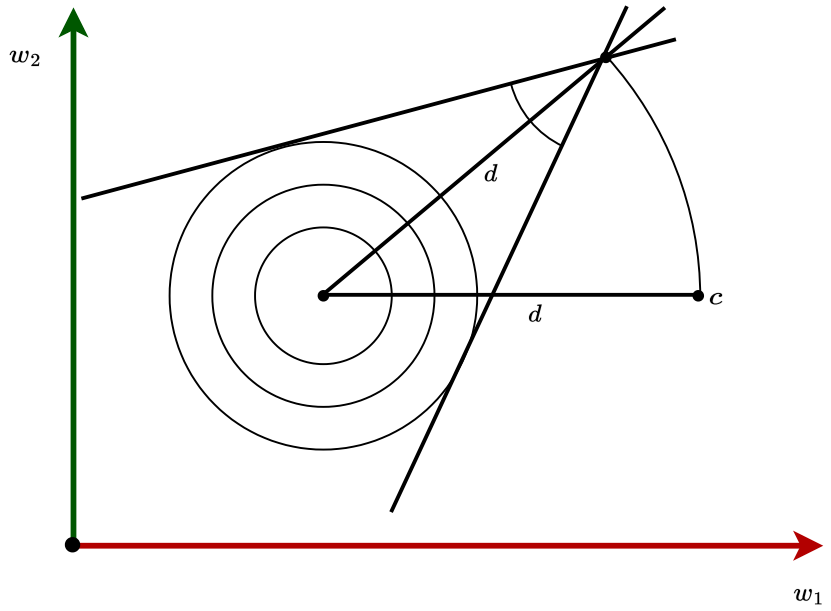


Figure B.1.: Camera placement for a static simple legacy camera in Simulation 1 capturing the entire circle with the maximum allowed radius of the circular motion model. Viewed from above.

To derive d, one first needs to calculate the horizontal angle of view (AOV_h). AOV_h in radians is given by,

$$AOV_h = 2 \arctan\left(\frac{w}{2f}\right), \tag{B.1}$$

where w is the sensor width and f is the focal length. The minimum distance necessary, d, such that the camera captures the entire circle with the largest radius, r_{max}, is then calculated by,

$$d = \frac{r_{max}}{\tan(2AOV_h)} \tag{B.2}$$

Let $p = \begin{bmatrix} p_1 & p_2 & p_3 \end{bmatrix}^T$ denote the camera centre position point in WCF and $c = \begin{bmatrix} c_1 & c_2 & c_3 \end{bmatrix}^T$ denote the centre position of the circles in WCF. When the camera is placed at a distance d from point c in a positive direction, and the camera view direction is parallel to the w_1-axis at a specified height h, the camera centre position point is the following,

$$p = \begin{bmatrix} c_1 + d \\ c_2 \\ h \end{bmatrix}$$

In order to point the camera towards the centre of the circles, the camera must be rotated. A yaw rotation of π is required to account for the fact that the camera is positioned on the positive side of the circle centre ($\psi = \pi$). Additionally, a pitch rotation is necessary to compensate for the position height. Given the height, h, the pitch rotation angle can be calculated using trigonometry,

$$\phi = \arctan\left(\frac{h}{d}\right). \tag{B.3}$$

Figure B.2 shows the resulting triangle.

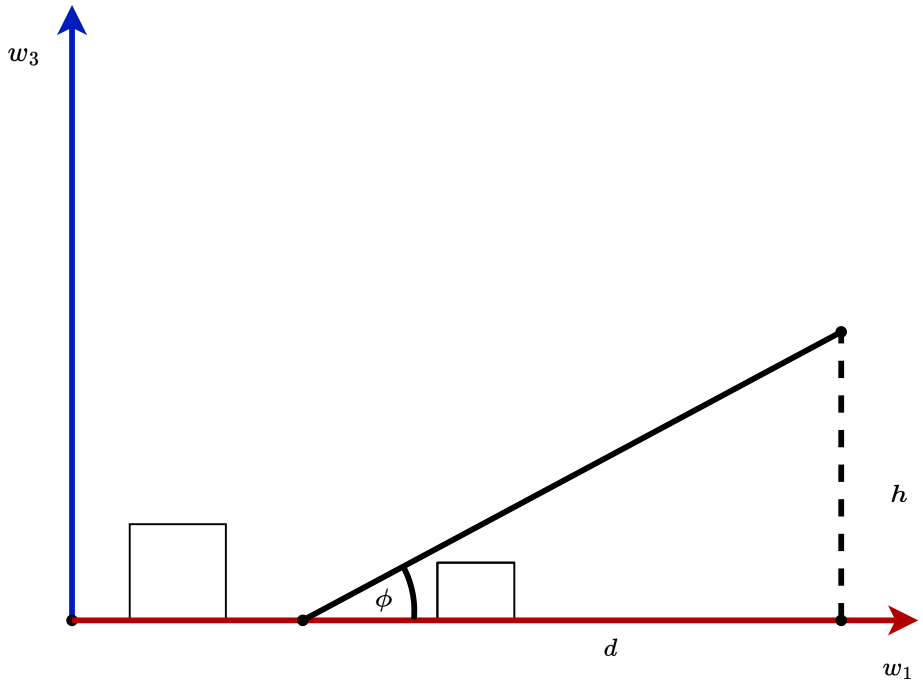
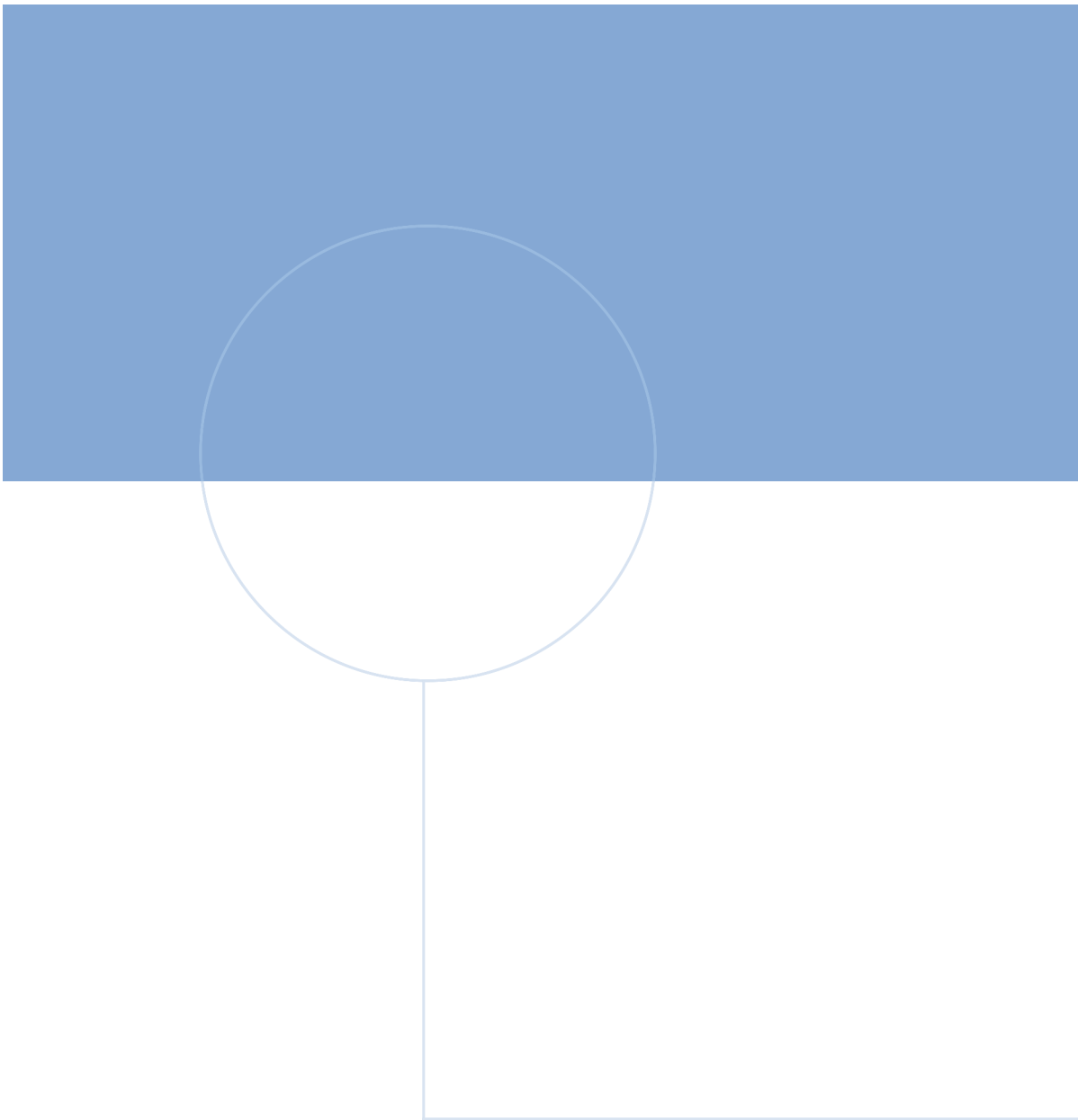


Figure B.2.: The pitch ϕ of the camera placement for a static simple legacy camera in Simulation 1 capturing the entire circle with the maximum allowed radius of the circular motion model.



NTNU

Norwegian University of
Science and Technology