Maritime Dock Detection using Camera

D.J. Scherrenburg

Delft University of Technology

TU Delft

DAMEN

Maritime Dock Detection using Camera

by

D.J. Scherrenburg

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday June 5, 2025 at 03:00 PM.

Student number:	5175151	
Project duration:	April 22, 2024 – June 5, 2025	
Thesis committee:	Dr. ir. H. Caesar,	TU Delft, supervisor
	Dr. ir. J.M. Prendergast,	TU Delft
	Ir. F. Verburg,	Damen
Supervisor Damen:	Ir. D. Kotiadis	

Cover: Veersteiger Boven-Hardinxveld

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

DAMEN

Abstract

Autonomous vessels offer potential benefits in safety, operational efficiency, and environmental impact, but require reliable perception systems to function independently. This thesis investigates the reliability of camera-based detection of maritime docks, a class of static but visually diverse objects that are often difficult to distinguish from their surroundings. While extensive research exists for detecting ships and buoys, docks remain underexplored, with no publicly available datasets containing a significant number of labeled instances.

To address this gap, the Dordrecht Dock Dataset was developed, containing 30,761 frames recorded under real-world maritime conditions across eight distinct docks. A custom interpolation-based annotation tool enabled efficient labeling, achieving annotation speeds of up to 3,000 frames per hour. Two deep learning models—Faster R-CNN and YOLO11n—were trained and evaluated on this dataset using a consistent pipeline. Evaluation followed a leave-one-dock-out cross-validation strategy with fixed test sets and standard performance metrics, including mAP50, mAP50–95, F1 score, and inference speed.

Faster R-CNN outperformed YOLO11n across nearly all accuracy metrics (mAP50 of 0.85 vs. 0.69), particularly at shorter ranges. YOLO11n demonstrated over twice the inference speed (37 FPS vs. 17 FPS), but exhibited weaker generalization and reduced reliability on visually different docks. Both models showed significant performance drops beyond 100–110 meters, emphasizing the need for high-resolution input or focus on short-range detection.

Contents

Abstract i

List of Figures iii

List of Tables iv

Nomenclature v

1 Introduction 1

2 Related Work 3
2.1 Advances in Deep Learning for Object Detection . . . 3
2.2 Automated Docking and Dock Detection Approaches . . . 4

3 Methodology 6
3.1 Dordrecht Dock Dataset . . . 6
3.1.1 Data Acquisition . . . 6
3.1.2 Annotation Process . . . 7
3.1.3 Dataset Composition . . . 7
3.2 DNN Training and Evaluation Frameworks . . . 8
3.2.1 Network Architectures . . . 8
3.2.2 Data Preprocessing . . . 10
3.2.3 Training Pipeline . . . 10
3.2.4 Evaluation Framework . . . 11

4 Experiments & Results 14
4.1 Baseline Comparison . . . 14
4.1.1 Experimental Setup . . . 14
4.1.2 Results . . . 15
4.2 Generalization to Unseen Dock Types . . . 19
4.2.1 Experimental Setup . . . 19
4.2.2 Results . . . 19

5 Discussion 22
5.1 Dataset Strengths and Limitations . . . 22
5.2 Training and Evalutation Methodology . . . 23
5.3 DNN Behavior and Performance . . . 23
5.4 Real-World Deployment . . . 24

6 Conclusion 25

References 26

List of Figures

3.1 Route of the Dordrecht Dock Dataset recording with locations of the docks. 6
3.2 Frame distribution in the Dordrecht Dock Dataset. 9
3.3 Annotation distribution in the Dordrecht Dock Dataset. 9
3.4 Comparison of the original image and the augmentations applied for Faster R-CNN and YOLO. 11

4.1 Distribution of frames across distance intervals for the training, validation, and test sets, shown separately for each of the four data splits in the baseline experiment. While the training and test sets are relatively balanced across all splits, the validation sets show greater variability, particularly in Splits 3 and 4 with spikes at 50-100m and 200-250m. . 15
4.2 Precision-recall curves for YOLO11n and Faster R-CNN on Split 1 of the baseline experiment, evaluated at an IoU threshold of 0.5. The curve for Faster R-CNN encloses a significantly larger area, indicating higher average precision across recall levels compared to YOLO11n. 17
4.3 Mean Average Precision at IoU threshold 0.5 (mAP50) as a function of distance for YOLO11n and Faster R-CNN on all four data splits of the baseline experiment. Showing significantly better performance on Splits 3 and 4, especially on long range (beyond 110 meters). 17
4.4 Close- and long-range predictions for each of the four data splits in the baseline experiment. The values at the top of the bounding boxes represent the confidence scores for those predictions. For generating the images, a confidence threshold of 0.1 is used for the short range and 0.01 for the long range. The white boxes are insets for visibility. . . 18
4.5 Mean Average Precision at IoU threshold 0.5 (mAP50) as a function of distance for YOLO11n and Faster R-CNN on all three test sets of the generalization experiment. Showing better performance of Faster R-CNN, a decreasing performance over distance, and a lousy performance on the Private dock. 20
4.6 Predictions at different ranges for the three test docks of the generalization experiment. The values at the top of the bounding boxes represent the confidence scores for those predictions. A confidence threshold of 0.01 is used to generate the images. The white boxes in the images on the right are insets for visibility. 21

List of Tables

3.1 Overview of dock recordings in the Dordrecht Dock Dataset. 9

4.1 Cross-validation data splits using a minimum label size of 5 pixels for the baseline comparison. All splits use *Waterbus dock Hollandse Biesbosch* as test set. 15
4.2 Performance metrics on the four data splits of the baseline comparison between Faster R-CNN and YOLO11n. Metrics computed at an IoU threshold of 0.5 and an optimal confidence threshold of 0.18 for Faster R-CNN and 0.03 for YOLO11n. 16
4.3 Inference time and FPS averaged over three runs for Faster R-CNN and YOLO11n. . . 16
4.4 Generalization results for YOLO11n and Faster R-CNN across three test datasets, including mean. Metrics computed at an IoU threshold of 0.5 and an optimal confidence threshold of 0.29 for Faster R-CNN and 0.01 for YOLO11n. The mean values are weighted over the total test set. 20

Nomenclature

Abbreviations

Abbreviation	Definition
AIS	Automatic Identification System
AP	Average Precision
CNN	Convolutional Neural Network
DFL	Distribution Focal Loss
DNN	Deep Neural Network
FPS	Frames Per Second
GNSS	Global Navigation Satellite System
HSV	Hue, Saturation, Value
IMU	Inertial Measurement Unit
IoU	Intersection over Union
LiDAR	Light Detection and Ranging
mAP	Mean Average Precision
mAP50	Mean Average Precision at IoU threshold 0.5
mAP50–95	Mean Average Precision averaged over IoU thresholds from 0.5 to 0.95
R-CNN	Region-based Convolutional Neural Network
RPN	Region Proposal Network
SGD	Stochastic Gradient Descent
TN	True Negative
TP	True Positive
FN	False Negative
FP	False Positive
YOLO	You Only Look Once
YOLO11n	Version 11 nano variant of the YOLO object detector
ViT	Vision Transformer

1 Introduction

Water transportation is essential to our global economy. Approximately 80 percent of the world's rapidly growing volume of goods is transported by water [44]. In addition, vessels are necessary for transporting passengers, specialized tasks such as fishing and supplying offshore activities, and the naval and military. Therefore, the advent of autonomous vessels is a significant step in the future of transportation with many advantages. It will increase safety by removing human error, increase cost efficiency by eliminating the need for a crew, which reduces operational costs and allows for more efficient use of space on board, and improve environmental sustainability by adjusting the route and speed to minimize fuel consumption and emissions [50].

A complete understanding of the surroundings is essential for making vessels fully autonomous. Therefore, it is necessary to develop a perception system that collects and processes all the required information from the environment so that the system can base its choices on this. These perception systems consist of only one or a combination of sensors such as LiDAR, Radar, (visible light) camera, thermal camera, and stereo camera. These sensors provide different types of information and, therefore, have their advantages and disadvantages, as concluded in the sensor analysis of Robinette et al. [39]. LiDAR produces 3D detections and performs well at short ranges. However, these sensors are not protected against environmental conditions in the maritime environment, just like other optical devices susceptible to lens hazing, and hydrometeors can degrade signal quality. 3D Radar produces 3D detections like LiDAR, works at longer ranges, is more resistant to hydrometeors, and can determine the relative velocity of detections using Doppler. Disadvantages are slow update rates and poor resolution. Cameras produce 2D detections and will therefore need an additional sensor to determine distances, but they generally perform well due to their rich information output (high resolution and color sensitivity). However, cameras have problems with bright sunlight and poor visibility due to severe weather or low-light conditions. Thermal cameras also produce 2D detections, have a much lower resolution, no color sensitivity, and may have difficulties with hydrometeors. However, they do not suffer from bright sunlight and work well in low-light conditions. Stereo cameras produce 3D detections and provide a rich information output with the advantages of regular cameras. The disadvantages are the same as those of regular cameras, with the additional advantage that they have a small range. This work will investigate the detection of objects with only a regular camera. This sensor is chosen because its effectiveness has been demonstrated many times in numerous industries [21, 32, 10, 24, 2, 31]. In addition, this sensor is the cheapest, easy to install, and intuitive to use. The choice to use a camera automatically means that this work will only focus on detection and not on localization of detections.

This work will specifically address docks as the object of interest. Docks are waterfront structures designed for vessels to berth, load, and unload. Accurate detection of these docks is essential for the complex autonomous docking challenge, which is an important step towards autonomous vessels. Detecting docks can be challenging as docks come in many different types and sizes and are sometimes difficult to distinguish from their surroundings because a quay can transition into a dock without a clear visual distinction. The detection of other common objects in the maritime environment, such as ships and buoys, has been studied extensively [9, 42], but dock detection is hardly discussed in these works.

A difference with other objects, such as ships, is that the location of the docks is static and known. Based on the map data and the GPS on board, the system can approach the dock within a short distance. However, GPS suffers from measurement and interpolation errors [35], which makes it unsuitable to rely solely on during the docking stage. Therefore, to better estimate the dock's position relative to the vessel at a close distance, it is essential to have an onboard perception system specialized in detecting the dock.

Automatic detection of objects in camera images means that the autonomous system must be able to locate objects in an image to subsequently place a bounding box around them. A bounding box is a rectangle that fits precisely around the object so that all desired parts of the object are inside. Nowadays, most object detectors rely on a Deep Neural Network (DNN). A DNN is a multi-layered artificial neural network that learns complex patterns in data using weighted connections, activation functions, and backpropagation. One of the challenges of DNNs is that training them requires a lot of data. This data must be of high quality, and all target objects must be indicated in all images, also known as labels. Generating a good and balanced dataset is expensive and time-consuming. For this reason, many works train the DNN on existing datasets and then fine-tune them to the target conditions. In the maritime sector, several datasets are available, mainly with ships and buoys [45]. Unfortunately, no datasets with a significant number of docks are found available. In this work, a custom dataset, the Dordrecht Dock Dataset, is created and used, an image of which can be seen on the cover.

Therefore, developing reliable dock detection is a critical step toward enabling autonomous docking systems—an area that remains largely underexplored. This leads to the central research question of this work:

To what extent can maritime docks be reliably detected using camera-based systems?

To investigate this, the following contributions are made in this work:

- A custom dataset for maritime dock detection, the Dordrecht Dock Dataset, is created and annotated to support training and evaluation of object detection models.
- A consistent training and evaluation framework is set up to perform a comparative study between Faster R-CNN and YOLO11n.
- An extensive evaluation is conducted to compare performance across dock types using cross-validation, offering insights into model robustness and generalization.

Structure: Chapter 2 reviews the literature on object detection advances and the use of these techniques on dock detection specifically. Chapter 3 provides an overview of the methodology used in this paper, including the dataset and the frameworks for training and evaluating. The experimental results are presented in Chapter 4 and discussed in Chapter 5. Finally, a conclusion with recommendations for future research is given in Chapter 6.

2

Related Work

Detecting maritime docks using camera-based systems is an emerging area within autonomous vessel technology. While object detection has seen significant advancements in recent years, its application to dock detection remains relatively limited. This chapter reviews key deep learning-based object detection developments and examines existing dock detection approaches.

2.1. Advances in Deep Learning for Object Detection

The advent of deep learning has transformed the field of computer vision and object detection [48]. This is particularly due to the development of Convolutional Neural Networks (CNNs). The foundation for CNNs was laid in 1980 by Fukushima [11], but the real breakthrough came with the performance of AlexNet in 2012 [23]. CNNs have three main layers: convolutional, pooling, and fully connected (FC). The convolutional layers form the most significant part of the network. These layers apply a specific filter to the image to extract certain features. These feature maps are larger than the input, so the pooling layer reduces this by averaging features. This reduces the complexity and makes the DNN more robust against noise. A stack of multiple of these two layers together forms the so-called backbone of the DNN. This backbone is followed by the FC layer, which performs the classification task and returns a probability for each defined output class. In this way, CNNs are capable of classifying entire images but not of detecting objects in images.

This is where the Region-Based Convolutional Neural Networks (R-CNNs) [13] gains its significance. This Region-Based CNN adds a region proposal layer to the beginning of the CNN. This layer extracts region proposals (possible objects) from the image, which are then passed through the CNN to determine whether it is an object that belongs to a target class. The disadvantage of this network is that it is inefficient because all region proposals ($\tilde{2}$k) are passed through the CNN separately. Fast R-CNN [12] tackles this issue by changing the order of the architecture. Passing the image through the CNN first and applying the region proposals to this output increases the speed by 9 times and improves accuracy. Faster R-CNN [38] further enhances this using a Region Proposal Network (RPN). This lightweight CNN generates region proposals directly and allows the entire pipeline to be trained end-to-end. This improvement has increased the speed to such an extent that it approaches real-time detection speed with even better accuracy.

Redmon et al. [37] then came up with the revolutionary idea of treating object detection as a single regression problem, allowing real-time detection at relatively high resolution. R-CNN methods are based on two steps, feature proposal and classification, which slows down the process considerably. With YOLO, this is all done in one step by having features extracted by convolutional layers and then having the fully connected layers determine the output probabilities and coordinates of these. With this, YOLO achieved comparable accuracy to Faster R-CNN on many standard object detection benchmarks but with significantly higher detection speeds. This DNN has laid the foundation for a family of real-time object detection DNNs with a similar operating principle. At the time of writing, the most recent version of YOLO is version 12. Each version improved the previous one in terms of accuracy and/or speed

on object detection benchmarks. For example, version 3 [36] changed the backbone to Darknet-53 to improve accuracy. Version 5 [17] streamlined training and deployment by switching to PyTorch. Version 8 [18] introduced a completely redesigned anchor-free architecture to improve object size and shape flexibility significantly. Finally, version 11 [19] improved accuracy with fewer parameters and, therefore, lower computational costs. In addition, this version is suitable for other tasks such as segmentation, classification, pose estimation, and oriented object detection. Many comparisons conclude that YOLO outperforms Faster R-CNN [34, 29] on both accuracy and speed. For this reason, YOLO has received a lot of attention over the past years.

Beyond CNN-based approaches, recent advances in object detection have explored Vision Transformers (ViTs). These models, such as DETR [5], have shown promising results in object detection, but often at the cost of higher computational and dataset demands [8]. Additionally, object tracking methods such as SORT [3] and DeepSORT [53] are frequently used in conjunction with detectors to enable temporal consistency across frames. While such techniques show compelling results, they fall outside the scope of this work. This work focuses on more established and proven methods to develop a baseline for dock detection research.

2.2. Automated Docking and Dock Detection Approaches

Detecting docking structures in camera images is a subject that has been investigated by a few and has not yet shown any consolidated results. The survey on automated docking by Lexau et al. [28] confirms that very little research has been conducted on detecting docks using cameras. This survey provides a comprehensive overview of research on automated docking for marine surface vessels from 1980 to June 2023. Notably, no works in this survey can autonomously detect the actual docking structure with a camera. It does cover some works that utilize specific aids to identify or locate the dock.

Choi et al. [6] propose a method to accurately detect docks in camera images in real time, based on template matching combined with vector code correlation. This method requires the operator to first indicate the desired target in an image, after which the algorithm searches for this target in subsequent video frames. While it shows promise in real-time performance, its dependence on human intervention makes it unsuitable for full autonomy.

Other methods aim to simplify detection through visual aids such as symbols or fiducial markers. Many of these have been developed in the context of the Maritime RobotX Challenge [41], where docks are augmented with colored shapes or markers to assist detection. Stanislas et al. [43] and Lee et al. [26] use LiDAR to locate the dock, followed by color thresholding techniques to identify symbolic features in the image. An exception to this two-stage approach is the work of Agcanas et al. [1], which uses an R-CNN to detect symbols in camera images without LiDAR-based pre-localization. These methods achieve reliable detection but depend heavily on artificially added visual aids, which are unlikely to be present in real port environments.

To overcome this reliance, some works suggest using fiducial markers for localization. Digerud et al. [7] and Volden et al. [51] present methods capable of accurately estimating the camera's relative position to these markers. However, both studies note limitations in adverse conditions, and again, such markers are not standard infrastructure in most ports.

Leite [27] explores detecting docking structures in simulation environments by utilizing color transformations to isolate and identify docks with unique geometries and predefined colors. This method works well in simulation but does not generalize to real-world docks that lack consistent coloring. More recent work by He et al. [15] enhances dock and ship detection using a YOLO-based CNN trained with anchor boxes initialized through K-means clustering. Although they successfully outperform standard YOLO on their dataset, the method lacks detail on dataset composition and is evaluated only under favorable daylight conditions.

Finally, Wang and Luo [52] propose a method that integrates a CNN with gated recurrent units to combine current perception with historical memory for autonomous docking. In this study, it is also unclear what exactly the dataset consists of, except that it is recorded in a simulation environment. Additionally, no performance metrics of this detection are presented, and it is unclear how it would perform in the real world.

In conclusion, current literature on dock detection with camera sensors is limited and primarily reliant on additional infrastructure such as symbols or markers. While deep learning-based methods show potential for enabling autonomous docking without visual aids, these methods remain underexplored and under-validated, particularly in complex, real-world maritime environments. As stated in [28], “truly autonomous surface vessels will need to be able to dock at any port to ensure safety and flexibility.” This highlights the need for further research into adaptable and infrastructure-independent approaches for dock detection using cameras.

3

Methodology

This chapter describes the methodology used in this research to detect maritime docks using cameras. The methodology comprises two main components. First, the creation of a novel dataset, the Dordrecht Dock Dataset, including image data captured from a vessel navigating Dutch waterways. This chapter describes the data acquisition and annotation process and presents the final dataset composition. Second, the design of a training and evaluation framework to compare two DNNs. The network architectures, preprocessing, training pipeline, and evaluation framework will be described.

3.1. Dordrecht Dock Dataset

As described in the introduction, no existing dataset contains a significant amount of labeled docks. One of the contributions of this work is the creation of a dock dataset to facilitate the development of a camera-based dock detection model. This section outlines the data acquisition process, annotation methodology, and composition of the Dordrecht Dock Dataset.

3.1.1. Data Acquisition

The dataset is collected using a sensor stand mounted on the bow of a Damen Schoalbuster. This sensor stand contains multiple sensors: a camera, thermal camera, LiDAR, GNSS, IMU, and AIS. Only the camera images are processed for this work, and the GNSS data is used to evaluate the performance. The camera is oriented in a forward-facing position, aligned with the vessel's direction of motion, with an elevation of approximately 4 meters relative to the water surface. The camera used for recording the data is the LI-AR0233-GW5400-FPDLINKIII-060H [30]. The main specifications of this camera are the resolution of 1920 x 1080, frame rate of 60 Hz, and field of view of 72° diagonally. Due to storage limitations and the redundancy of many similar images, images are captured at an average rate of 15 Hz.

Figure 3.1: Route of the Dordrecht Dock Dataset recording with locations of the docks.

The data acquisition took place on the 18th of September 2024, from 09:00 to 15:00. The weather started very foggy and cloudy, but this cleared up during the day and became sunny. The acquisition took place in the area of Dordrecht, The Netherlands. Figure 3.1 shows the total covered route on this day, with the recorded dock locations marked with a pin. These docks were selected in advance based on their properties. Most of these docks are intended for ferries and are distinct from the background. This forms a good starting point/baseline for the research question as to what extent it is possible to detect docks. This is the best route achievable in a time span of one day, and these are the only docks on this route that meet these requirements.

During data acquisition, the docks are approached from as large a distance as possible, and some are approached multiple times from different angles. Data was only recorded while approaching the docks. The total dataset is presented in Section 3.1.3.

3.1.2. Annotation Process

To train a DNN, it is necessary to provide the model with the desired result. Simply running the images through the network will not result in the model detecting the docks. The model must be provided with the desired outcome to know which predictions are correct or incorrect during training. Therefore, the DNN needs an additional input, namely the ground truths, also called labels. For object detection purposes, labels in the form of bounding boxes are required. A bounding box is a rectangular shape placed around the target object, as seen in green in Figure 3.4b. This bounding box has a class that indicates to which object it belongs. In the case of this work, there is only 1 class, namely a dock.

Assigning these labels, also known as annotating, is a labor-intensive process. Each image in the dataset must be manually provided with a bounding box. Several labeling tools are available online [49, 40, 25], each with advantages and disadvantages. Since the dataset consists of video fragments, using an annotation tool that supports interpolation is desirable. Interpolating labels is a powerful way to speed up this process. In static or nearly linear scenarios, only the first and last frames of the video fragment need annotations using this technique. The intermediate frames can be interpolated. Since no open-source annotation tool meeting the requirements could be found, a custom tool is created for this project. This custom interpolation tool has significantly accelerated the annotation process, even though the maritime environment is not optimally suited for interpolation. Due to the irregular heave of the vessel on the waves, it is not possible to manually annotate only the first and last frames. The used annotation strategy focuses on identifying the peaks and troughs of the vessel's sinusoidal heave motion. These key positions served as anchors for the temporal interpolation of the intermediate frames. Finally, the interpolated frames are reviewed and corrected where necessary. All in all, this custom tool ensured that only 4% of the frames had to be annotated manually, resulting in a peak annotation speed of around 3000 frames per hour.

In each frame, all docks are annotated. Most images contain one dock because the recorded ferry docks are generally isolated. The docks are annotated until it is no longer possible to distinguish the dock from the surroundings due to the low resolution at large distances. Partly occluded docks are also annotated and included in the dataset. Figure 3.4b shows an example of a dock with its label. The bounding boxes are exported as TXT files in the YOLO format. In this format, every label is expressed as [class, x center, y center, width, height], with relative coordinates.

3.1.3. Dataset Composition

The dataset contains a total of 8 docks with varying geometries and backgrounds. The width varies between 10 and 35 meters, and the background varies from dikes and forest to buildings and industry. The recordings of these docks result in a total dataset size of 30761 frames with 28703 labeled docks. The exact numbers per dock are shown in Table 3.1, showing the recording duration, distance range, and approach angles. It is important to note that the dataset is not optimally balanced. Recordings of several docks, mainly Veersteiger Boven Hardinxveld, contain significantly more frames, a longer duration, a larger distance range, and more approach angles. It is good to note that the Veersteiger is recorded at a higher framerate, resulting in more similar frames.

Figure 3.2 shows an extensive frame distribution. In Figure 3.2a, the spatial distribution of all frames can be seen relative to the dock. From this, it can be concluded that the docks are approached from varying angles. However, this plot does show that the majority are captured within the angles of 50

and 80 degrees to the dock. This is the natural approach angle for most docking maneuvers of ferries, and therefore most important. It can also be seen that most frames were captured within a distance of 750 meters to the dock, with a recording of the Waterbus dock Rosmolenweg being the exception. Figure 3.2b shows the frame distribution over the distance to the dock more clearly. Here, it can be seen that a large part of the frames are taken at a distance of 0 to 100 meters, and the number of frames decreases as the distance increases.

As described, all frames have been provided with bounding box labels using the interpolation annotation tool. The annotation distribution is shown in Figure 3.3. In Figure 3.3a, the centers of the bounding boxes are shown, indicating that nearly all frames are located in the upper half of the image, with the majority in the image center. In addition, Figure 3.3b shows the size of the bounding boxes. From this, it can be concluded that the majority of the annotations have a height lower than 0.2 and a width lower than 0.4. Furthermore, this shows that the width is, on average, larger than the height, corresponding to the dock size.

It is good to take away from this dataset composition that it does not show high complexity. It contains only eight different docks, most of which belong to the category Waterbus dock, meaning there is not much variation. The individual frames also show little variation as they are extracted from a video, meaning successive frames appear similar. Finally, the dataset focuses on a single object class, which is less complex than multiple object classes.

3.2. DNN Training and Evaluation Frameworks

In Related Work (Chapter 2), several DNNs used in the literature to detect objects in images are discussed. In this work, Faster R-CNN and YOLO will be compared. Both DNNs have been studied thoroughly and provide established baselines. Faster R-CNN offers reliable performance and a stable baseline. YOLO, on the other hand, continues to develop, maintaining its state-of-the-art position. One of the advantages of Faster R-CNN over YOLO for corporate use is the licensing. The YOLO models from Ultralytics are licensed under the AGPL-3.0 license [14], stating that the software must be made public when modified or distributed. The Faster R-CNN model from PyTorch, on the other hand, is licensed under the BSD 3-Clause license [47], which does not require this disclosure. Vision Transformers are not included in this comparison due to their proven data requirements [8] combined with the low complexity of the current dataset.

Training DNNs involves many variables and processes that can be adjusted to influence performance. To make the comparison between the YOLO and Faster R-CNN models as fair as possible, the frameworks have been kept as similar as possible. Before discussing the training pipelines, we will go into more detail about the two DNNs, and finally, the evaluation framework will be discussed.

3.2.1. Network Architectures

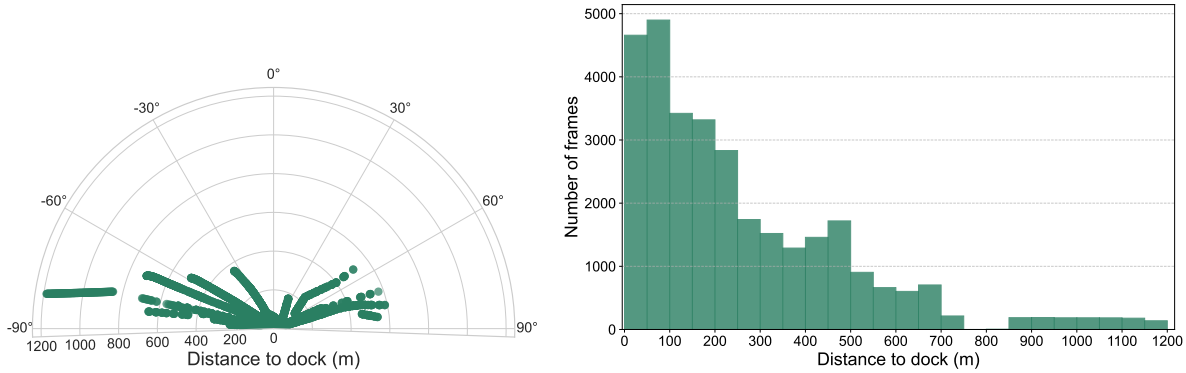
The general network structures of the two DNNs have been discussed in Related Work (Chapter 2). However, both DNNs still offer several architectural choices to be made. For YOLO, there are different versions and network sizes. Faster R-CNN offers the possibility of selecting a different backbone. The backbone of both DNNs can also be (partially) frozen to obtain a more stable training result and prevent overfitting. Given the limited complexity of the dataset, as there is little variation and only a single object class, it is advisable to avoid using overly complex DNNs. DNNs with high complexity are more prone to overfitting, especially with non-complex training data. Overfitting can lead to a DNN that also learns all the noise in the dataset instead of just the meaningful patterns. This will lead to good performance on the training data, but poor generalization to unseen data [54]. For this reason, using a simpler DNN is appropriate in this work. It also has the advantage of being less computationally expensive, resulting in lower training time, and the ability to run on less advanced hardware.

YOLO

In this work, YOLO11n [19] is used. This version is selected because it showed the best state-of-the-art results at the time while reducing complexity compared to previous versions. To further reduce the complexity of the model, the nano (n) variant is used.

Table 3.1: Overview of dock recordings in the Dordrecht Dock Dataset.

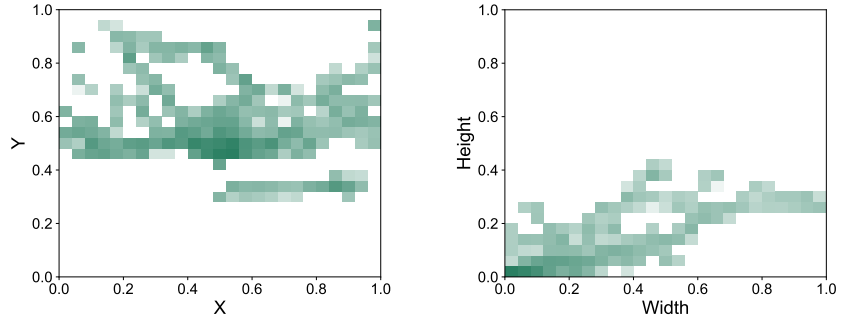
Dock	Labeled docks	Frames	Duration (s)	Distance range (m)	Angles (deg)
Waterbus dock Rosmolenweg	5759	6045	667	0-1200	-75, 80
Bunkerstation Dekker & Stam	6482	3771	253	0-375	-75, 25, 70
Waterbus dock Noordhoek	1510	1510	82	0-300	-75
Waterbus dock Noordeinde	807	1486	127	200-500	50
Waterbus dock Hooikade	657	657	55	50-230	-84
Veersteiger Boven Hardinxveld	13400	13846	415	0-700	-65, -30, 60, 80
Waterbus dock Hollandse Biesbosch	2668	2732	264	0-575	-75, 70
Private dock Middeldiep	420	714	76	40-70	-30
Total	**28703**	**30761**	**1939**	**0-1200**	



(a) Polar plot showing the spatial distribution of camera viewpoints relative to the dock.

(b) Histogram of the number of frames as a function of distance to the dock.

Figure 3.2: Frame distribution in the Dordrecht Dock Dataset.



(a) Heatmap of normalized bounding box center positions.

(b) Heatmap of normalized bounding box width and height.

Figure 3.3: Annotation distribution in the Dordrecht Dock Dataset.

Faster R-CNN

The best-known backbones for Faster R-CNN are those of the ResNet family [16], since these have proven to be easier to optimize and improve accuracy. These backbones range from a depth of 18 to 152 layers. The ResNet50 backbone is used in this work because it offers a good compromise between complexity and speed.

3.2.2. Data Preprocessing

Before the images can be passed through the networks, several transformations are required. In this work, the preprocessing consists of downsizing the images and certain pixel transformations.

Image Downsizing

Training a DNN requires a lot of processing power and memory from a computer. It is, therefore, common to downsize the images to require less computational power from the computer and to enable real-time detection. For this reason, the images in this training pipeline are reduced to a length of the longest side of 640 pixels while maintaining the aspect ratio. For this downsizing, the bilinear interpolation method is used. This method uses a weighted average of the nearest pixels to offer a good balance between speed and visual quality. Because the labels are stored as relative values, they automatically scale with the new resolution. Downsizing images results in less detailed docks, impacting detection at long distances. Since the focus in this work is on short range, this should not result in detection issues.

Pixel Transformations

The input images must meet the DNN-specific expectations. The DNNs are trained on a specific input format and will, therefore, be able to apply their prior knowledge best to this input. A difference between the input requirements of YOLO and Faster R-CNN is that YOLO expects the images in BGR format and Faster R-CNN expects RGB. In addition, most DNNs require the pixel values of the input images to be normalized, and so do YOLO and Faster R-CNN. This is applied by dividing the pixel values by 255. Finally, it is common for Faster R-CNN to standardize the color channels using the mean and standard deviation values of the pretraining dataset. This ensures a consistent input distribution across training samples.

3.2.3. Training Pipeline

To ensure that the results of the two networks can be compared as well as possible, the training pipeline will be as similar as possible. This section discusses this training pipeline, including the augmentation strategy and training configuration.

Augmentation Strategy

Data augmentation involves modifying the input images to enhance the dataset. This technique generally enhances generalization and minimizes the risk of overfitting [22]. The following augmentations are used:

- Mosaic augmentation [4] (to YOLO only)
- Affine transformations
- Random HSV
- Random flip

In Figure 3.4 the augmented images of Faster R-CNN and YOLO are shown next to the original. The idea of mosaic data augmentation is that multiple images are merged into one image. An example of this can be seen in Figure 3.4c, where the input image now consists of 4 frames. For the Faster R-CNN, the Mosaic augmentation is chosen to be left out. The main reason is that Faster R-CNN largely depends on the RPN. When multiple images are merged, the trained regions will deviate significantly from the actual object regions, negatively affecting the network's performance. The affine transformations contain zooming and translating, as shown in Figure 3.4b. This shift towards the upper left does not result in unrealistic object placements compared to mosaic and has been applied to Faster R-CNN. Random HSV and flip can also be seen in this figure. The colors are a bit brighter than in the original, and the image is flipped from left to right.

Each time a batch of images is fed to the DNN in the training phase, these augmentations are applied to the images with a certain probability or random strength. For example, the color value will be modified with a random percentage and each image will be flipped with a probability of 0.5. This ensures that the DNN processes a slightly different training dataset during each epoch, making it less likely to overfit.

(a) Original image **(b)** Faster R-CNN augmented **(c)** YOLO augmented

Figure 3.4: Comparison of the original image and the augmentations applied for Faster R-CNN and YOLO.

Training Configuration

The recommended training configuration for YOLO is retained where possible. Both DNNs are trained for 100 epochs with an early stopping of 50 epochs based on the fitness score. This means the training is automatically terminated if no improvement is detected in the fitness score for 50 consecutive epochs. This fitness score is described in the section on performance metrics (Section 3.2.4). A linear learning rate schedule is used, starting with a learning rate of 1×10^{-3}, decreasing by 1% every epoch. The optimizer used for both DNNs is a Stochastic Gradient Descent (SGD) with momentum (0.9) and weight decay (5×10^{-4}). For both DNNs, a batch size of 8 is used. The default batch size for YOLO is 16, but this is reduced due to memory constraints that Faster R-CNN would otherwise exceed. Due to architectural differences, loss functions are not aligned directly between DNNs: YOLO11 uses a composite loss including bounding box regression (CIoU), classification, and distribution focal loss (DFL); Faster R-CNN uses standard RPN and detection losses.

3.2.4. Evaluation Framework

It is important to think carefully about the evaluation framework for a fair comparison. The evaluation must consist of metrics that assess the DNNs correctly. These metrics will be discussed in this section. In addition, the strategy to perform the evaluation fairly will be discussed.

Performance Metrics

To compare the DNNs, it is necessary to define certain scores, also known as metrics. This work uses commonly used metrics in object detection, including Intersection over Union, precision, recall, and mean Average Precision. This section will explain these metrics.

The object detectors aim to predict bounding boxes that match the ground truth boxes as closely as possible. Every prediction comes with a confidence score, representing the DNN's confidence in this prediction. However, high confidence does not guarantee that the prediction matches the ground truth. To quantify how well the predictions overlap with the ground truths, *Intersection over Union (IoU)* is used. IoU measures, as the name suggests, the ratio between the overlapping area and the combined area of the prediction and ground truth boxes:

$$\text{IoU} = \frac{\text{area of intersection}}{\text{area of union}}. \tag{3.1}$$

When the boxes overlap perfectly, this returns an IoU score of 1; when there is no overlap, a score of 0. This metric can be used on its own by averaging the IoU over all matched predictions, but it also serves as a starting point for the other metrics.

The IoU is very useful in determining whether a predicted bounding box is correct (True Positive (TP)) or incorrect (False Positive (FP)). The DNN can also predict that no object can be found; this is called a True Negative (TN) if correct and a False Negative (FN) if incorrect. These metrics are then used to determine precision and recall. *Precision* measures the ratio of True Positives to the total number of predictions, and *recall* measures the ratio of True Positives to the total number of ground truths:

$$\text{Precision} = \frac{\text{TPs}}{\text{TPs} + \text{FPs}} = \frac{\text{True Positives}}{\text{Total Predictions}}, \tag{3.2}$$

$$\text{Recall} = \frac{\text{TPs}}{\text{TPs} + \text{FNs}} = \frac{\text{True Positives}}{\text{Total Ground Truths}}. \tag{3.3}$$

In practice, precision is most useful in applications where it is essential not to make wrong decisions because a precision of 1 means that all predictions are correct. However, it is possible that the detector missed objects. Recall focuses on these misses as a recall of 1 means that all objects are predicted correctly, but there is a chance that it made too many predictions and is often wrong. A commonly used metric that balances precision and recall is the F1 score, which is the harmonic mean of the two. It is particularly useful when seeking a single value to summarize model performance, especially in cases with imbalanced data:

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{3.4}$$

The F1 score is highest when both precision and recall are high, and it penalizes extreme imbalances between them. In addition, there is the precision-recall curve. This curve plots precision against recall for a varying confidence threshold. By increasing this threshold, the DNN will omit uncertain predictions, which generally results in an increase in precision and a decrease in recall. The goal is to optimize the area under this curve as this means that both precision and recall have a high value for a varying confidence threshold, and, therefore, no trade-off needs to be made.

To give a value to this area, the metric *mean Average Precision (mAP)* has been introduced. This metric is determined by averaging the Average Precision (AP) over the different classes, as can be seen in the following equation:

$$\text{mAP} = \frac{1}{N} \sum_{i=0}^{N} \text{AP}_i \ , \quad with \ \text{AP} = \int_0^1 \text{Precision}(r) \, dr \ , \tag{3.5}$$

with *N* being the number of classes. Of course, this work only has 1 class, which makes mAP equal to AP, but since mAP is the most commonly used metric for object detection, we will refer to it as mAP. This mAP can be further extended by calculating this metric for different IoU thresholds. mAP50 is used for the mAP at an IoU threshold of 0.5, and mAP50-95 represents the average mAP over an IoU threshold of 0.5 to 0.95.

For the results in this work, the confidence threshold will be determined by maximizing the F1 score and for the IoU threshold a value of 0.5 will be used. The fitness score is determined by $0.1 \times mAP50 + 0.9 \times mAP50{-}95$.

In addition to these metrics that assess the quality of the detections, inference speed is also an important property. *Inference speed* is the speed at which the DNN makes a prediction, which is very important for real-time applications, such as this application of detecting docks. Inference speed will be expressed in frames per second (FPS).

Evaluation Strategy

In addition to the metrics for determining performance, it is equally important that these metrics are obtained and compared in a fair manner. First, it is important that as many variables as possible are identical between the DNNs, as discussed in this section. Close attention will therefore be paid to this.

Second, splitting the data into train, validation, and test sets is essential. This ensures that the data used to train the DNN is not used for validation and testing. This makes it possible to measure the DNN's performance fairly on unseen data. In the case of this work, however, the dataset contains very similar frames because they come from a video recording. As a result, a random distribution will still

result in the exposure of similar data in the three sets. A validation and test set will be created to prevent this, consisting of all recordings from one or more docks. This ensures that there is no similar data in the three sets and that the validation and test sets are completely unseen.

However, when the DNNs are trained on a specific data split, it may be the case that one DNN performs above average and the other does not. To prevent this, multiple data splits will be made to determine the average performance over these splits. This will eliminate the chance factor and generate a robust outcome that can be compared fairly with each other. Using multiple data splits is also called cross-validation.

This work will use a leave-one-dock-out cross-validation strategy with a fixed test set. This means that all splits use the same test set consisting of all complete recordings of one or more docks. Then, for each split, all complete recordings of one or more different docks will be taken from the train set to be used as a validation set. All in all, this evaluation framework will ensure an accurate performance evaluation for a fair comparison.

All experiments were conducted on a laptop equipped with the following hardware and software:

- Operating system: Ubuntu 22.04 LTS
- CPU: AMD Ryzen 7 5800H with Radeon Graphics
- Memory: 16 GB DDR4 RAM
- Graphics Card: NVIDIA GeForce RTX 3050

4

Experiments & Results

To answer the question of the extent to which maritime docks can be reliably detected with camera-based systems, experiments and their results will be discussed in this chapter. These experiments aim to achieve the best possible dock detection result and to analyze what influences these results. For this reason, YOLO11n and Faster R-CNN will be thoroughly compared in this chapter. First, a thorough baseline comparison will be made between the performance of these two DNNs. Subsequently, the differences will be further explored.

4.1. Baseline Comparison

The goal of the baseline comparison is to get a good impression of the DNNs' performance, especially the differences between them. First, the setup will be discussed, after which the quantitative and qualitative results will be shown.

4.1.1. Experimental Setup

The frameworks described in Section 3.2 will be used for the aseline comparison. It is described here that for accurate performance evaluation, it is important to apply cross-validation. A leave-one-dock-out cross-validation strategy with a fixed test set is used for this baseline comparison. It is decided to exclude Veersteiger Boven Hardinxveld from the validation sets because of the significant impact on the size of the train set, and to merge the smaller sets for a comparable size. In addition, it is decided to filter the dataset for this comparison. For this baseline, a minimum label size filter is used. This means that all data frames that have a label with a value lower than the minimum pixel value are omitted. A DNN has more difficulty detecting objects that cover few pixels in the image. For example, a recent work shows that the various YOLO versions perform significantly better on objects appearing larger in the image [46]. For this baseline, images are included if both the width and height of all objects have a minimum size of 5 pixels in the downsized images. This results in a maximum distance to the docks of 322 meters, as can be seen in Figure 4.1, since frames captured at longer ranges are excluded. For this work, this is not a problem since it is precisely the short distance that is of interest. Eliminating docks outside this range also ensures that the dataset is better balanced because, as seen in Table 3.1, the docks with the most frames also contain the longest distances. All in all, these filters ensure that the DNNs will have less difficulty training the dataset, which will provide a strong baseline comparison.

The resulting data splits can be seen in Table 4.1. Both DNNs are trained on all data splits for 100 epochs with early stopping after 50 epochs without improvement. Figure 4.1 shows the distribution of the frames over the distance ranges to the docks. This shows that the train set shows a similar distribution to the test set for all data splits. However, more variation can be seen in the validation set. Splits 1 and 2 show a distribution that is also comparable to the train and test sets, but for Splits 3 and 4 this is not the case. Split 3 shows a clear peak at 50-100 meters and for Split 4 this is at 200-250 meters.

Table 4.1: Cross-validation data splits using a minimum label size of 5 pixels for the baseline comparison. All splits use *Waterbus dock Hollandse Biesbosch* as test set.

Split	Train Frames	Val Frames	Val Docks	Test Frames
Split 1	13021	2747	Waterbus dock Rosmolenweg	1652
Split 2	12780	2988	Bunkerstation Dekker & Stam	1652
Split 3	13838	1930	Waterbus dock Noordhoek, Private dock Middeldiep	1652
Split 4	14833	935	Waterbus dock Noordeinde, Waterbus dock Hooikade	1652

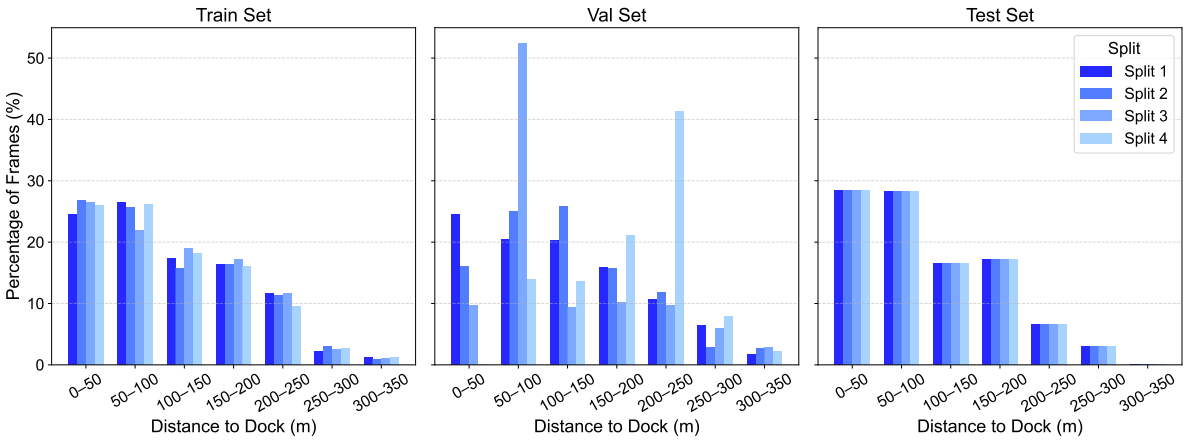


Figure 4.1: Distribution of frames across distance intervals for the training, validation, and test sets, shown separately for each of the four data splits in the baseline experiment. While the training and test sets are relatively balanced across all splits, the validation sets show greater variability, particularly in Splits 3 and 4 with spikes at 50-100m and 200-250m.

4.1.2. Results

The scores of both DNNs on the metrics introduced in Section 3.2.4 are shown in Table 4.2. The results are presented together with the mean and standard deviation (std) taken over the four data splits. It is important to note that all metrics presented in this table operate on a scale of 0 to 1, with 1 representing the best possible value. In addition, the std should be as low as possible, meaning the results are consistent across the different splits.

The precision, recall, and F1 scores depend on the confidence threshold. The optimal threshold for the individual models is used for the results shown. This threshold is determined based on the maximum F1 score, resulting in a confidence threshold of 0.18 and 0.03 for Faster R-CNN and YOLO11n, respectively. It can be seen that all three of these metrics are in favor of Faster R-CNN. The other metrics are independent of the confidence threshold, resulting in a fairer comparison. Both mAP values are in favor of Faster R-CNN, with the most significant difference in mAP50, outperforming YOLO11n by 25%. The difference in mAP50-95 is smaller, suggesting that the predictions of YOLO11n show a higher average IoU score, which is also reflected in the IoU score. The std values of both DNNs are pretty high for most metrics. Looking at the individual split results, it becomes clear that this is due to the lower performance on the first two data splits.

To further explore the difference in mAP50 between the DNNs, the precision-recall curve in Figure 4.2 is valid. This figure confirms that Faster R-CNN performs better, as the area under this graph is much larger. What is striking is that the graph of Faster R-CNN shows a precision of 1 up to a recall of more than 0.5, while for YOLO11n, this is the case up to just over 0.2. In addition, it is striking that the recall value of the YOLO11n graph only reaches just above 0.8. This means that even with an extremely low confidence threshold, the results of YOLO11n will still miss some docks, while Faster R-CNN is capable of detecting nearly every dock using a low confidence threshold. In addition, the plot in Figure 4.3 provides valuable information. In this figure, the mAP50 metric is shown as a function of the distance to the dock for the four data splits. From this plot, it becomes clear that Faster R-CNN shows better performance at most distances for all data splits. It can be seen that both DNNs show a less stable performance at larger distances (above 110 meters). In addition, it is noticeable that for

Table 4.2: Performance metrics on the four data splits of the baseline comparison between Faster R-CNN and YOLO11n. Metrics computed at an IoU threshold of 0.5 and an optimal confidence threshold of 0.18 for Faster R-CNN and 0.03 for YOLO11n.

Model	Split	IoU	Precision	Recall	F1	mAP50	mAP50-95
Faster R-CNN	Split 1	0.730	0.898	0.620	0.734	0.760	0.391
	Split 2	0.701	0.584	0.725	0.647	0.726	0.334
	Split 3	0.753	0.910	0.886	0.898	0.946	0.490
	Split 4	0.765	0.945	0.880	0.912	0.972	0.541
	Mean $\pm$ Std	$\mathbf{0.737 \pm 0.028}$	$\mathbf{0.834 \pm 0.168}$	$\mathbf{0.778 \pm 0.129}$	$\mathbf{0.798 \pm 0.129}$	$\mathbf{0.851 \pm 0.126}$	$\mathbf{0.439 \pm 0.094}$
YOLO11n	Split 1	0.745	0.781	0.450	0.571	0.572	0.338
	Split 2	0.750	0.567	0.398	0.467	0.474	0.302
	Split 3	0.790	0.716	0.886	0.792	0.859	0.514
	Split 4	0.795	0.976	0.674	0.797	0.863	0.541
	Mean $\pm$ Std	$\mathbf{0.770 \pm 0.026}$	$\mathbf{0.760 \pm 0.170}$	$\mathbf{0.602 \pm 0.224}$	$\mathbf{0.657 \pm 0.165}$	$\mathbf{0.692 \pm 0.199}$	$\mathbf{0.424 \pm 0.121}$

both DNNs, the performance on the first two data splits decreases from a lower distance than the last two. Finally, it can be seen that the performance on certain longer distances is suddenly very good. Especially YOLO11n shows a strong peak performance on the distance of 210-220 meters for all data splits.

The same patterns can be seen in the qualitative results in Figure 4.4. This figure shows the ground truth and predictions with confidence scores for both DNNs on all data splits for a close- and long-range frame. Both DNNs show consistent results on all data splits at close range with correct predictions and high confidence scores. Only Split 2 is a small exception for this frame due to the false positive of Faster R-CNN and the low confidence of YOLO11n. However, this false positive will generally be filtered out since the confidence score is low, and the low confidence of YOLO11n will be just higher than the plausible maximum confidence threshold of 0.25. In this case, the result will be the same as the other splits but less stable.

There is more going on in the qualitative results on long range. YOLO11n is only able to correctly detect the dock in Split 3 with a low confidence score. In addition, it shows a false positive in Splits 1 and 2, but these confidence scores of 0.01 will be filtered out. Faster R-CNN shows a lot more detections, except for Split 1. Split 2 shows several true positives, including a correct detection with a confidence of 0.36 that is not visible. In contrast, it shows many false positives on the left and bottom right with confidence scores up to 0.25 on the left side. In Splits 3 and 4, it correctly detects the dock without false positives, but the confidence score for Split 3 is pretty low.

In addition to the discussed metrics that assess the quality of the detections, inference speed is also an important metric. For both DNNs, the inference speed is measured by making the DNN predict 1000 images and measuring the time of each prediction. These times are then used to determine the mean, std, min, max, and FPS. This test is performed three times independently, and the average values of these three runs can be seen in Table 4.3. As expected, YOLO11n shows a significantly higher FPS, namely more than twice as high. However, it is clear from the std values that Faster R-CNN has a more stable inference speed, since the std value is significantly lower.

Table 4.3: Inference time and FPS averaged over three runs for Faster R-CNN and YOLO11n.

Model	Mean (ms)	Std (ms)	Min (ms)	Max (ms)	FPS
Faster R-CNN	59.414	2.969	55.786	107.642	17
YOLO11n	27.307	15.075	11.861	116.661	37

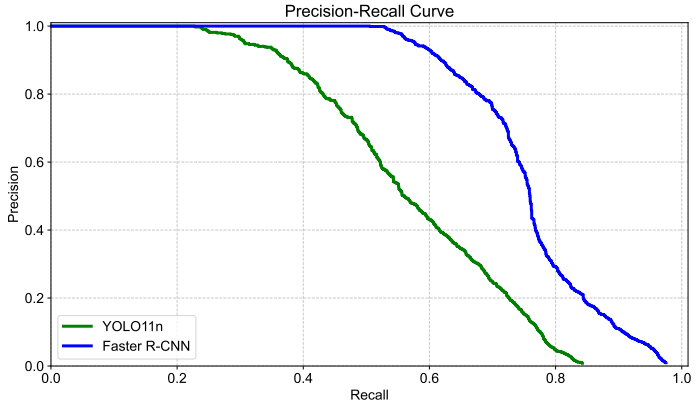
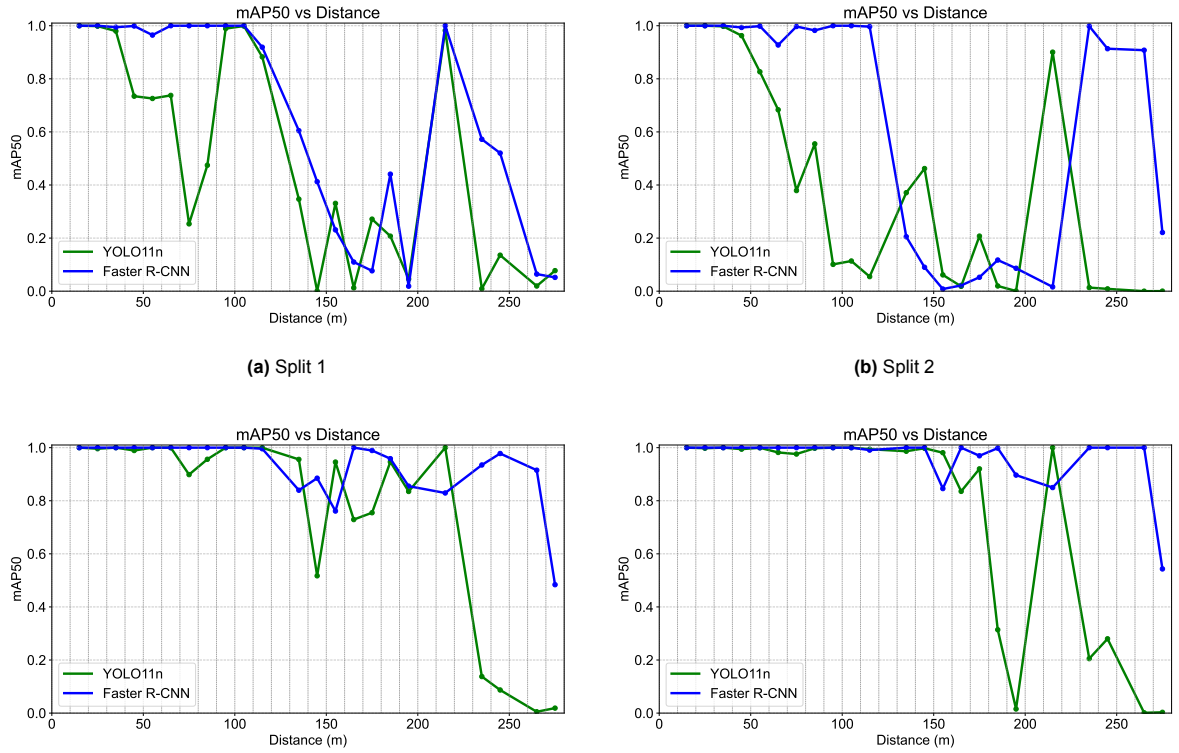


Figure 4.2: Precision-recall curves for YOLO11n and Faster R-CNN on Split 1 of the baseline experiment, evaluated at an IoU threshold of 0.5. The curve for Faster R-CNN encloses a significantly larger area, indicating higher average precision across recall levels compared to YOLO11n.



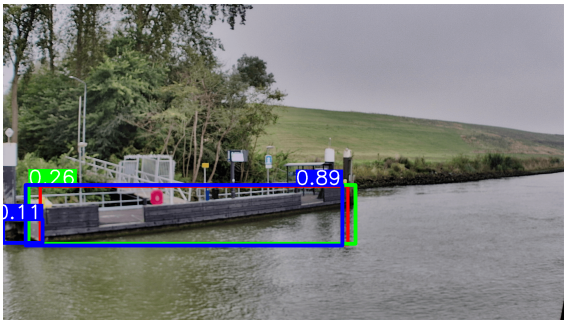
(a) Split 1 **(b)** Split 2

(c) Split 3 **(d)** Split 4

Figure 4.3: Mean Average Precision at IoU threshold 0.5 (mAP50) as a function of distance for YOLO11n and Faster R-CNN on all four data splits of the baseline experiment. Showing significantly better performance on Splits 3 and 4, especially on long range (beyond 110 meters).

(a) Split 1 - Close (30m)

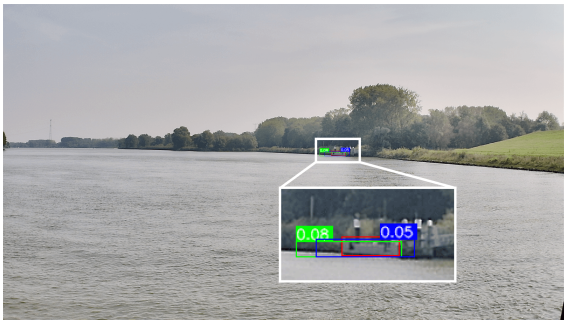
(b) Split 1 - Long (270m)



(c) Split 2 - Close (30m)

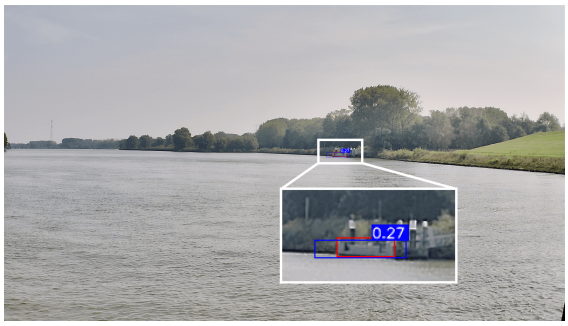
(d) Split 2 - Long (270m)

(e) Split 3 - Close (30m)



(f) Split 3 - Long (270m)

(g) Split 4 - Close (30m)



(h) Split 4 - Long (270m)

Ground Truth YOLO11n Faster R-CNN

Figure 4.4: Close- and long-range predictions for each of the four data splits in the baseline experiment. The values at the top of the bounding boxes represent the confidence scores for those predictions. For generating the images, a confidence threshold of 0.1 is used for the short range and 0.01 for the long range. The white boxes are insets for visibility.

4.2. Generalization to Unseen Dock Types

This experiment aims to investigate the generalization capacity of the DNNs. Due to the limited variation in the dataset, it is interesting to know whether other types of docks could also be detected, without the DNNs being specifically trained on these.

4.2.1. Experimental Setup

For this experiment, the train and validation sets will consist of docks of the Waterbus type only. These docks are very similar in appearance, with similar geometry and colors, but the backgrounds differ. The DNNs will then be tested on the Veersteiger, Bunkerstation, and Private dock sets. The Veersteiger dock has a comparable appearance, but the Bunkerstation and Private dock are significantly different. The minimal side size of five pixels for the label boxes is maintained for this test. For this test, only one dataset is used for a faster result. This dataset uses Waterbus dock Noordhoek for validation and the rest of the Waterbus docks for training.

4.2.2. Results

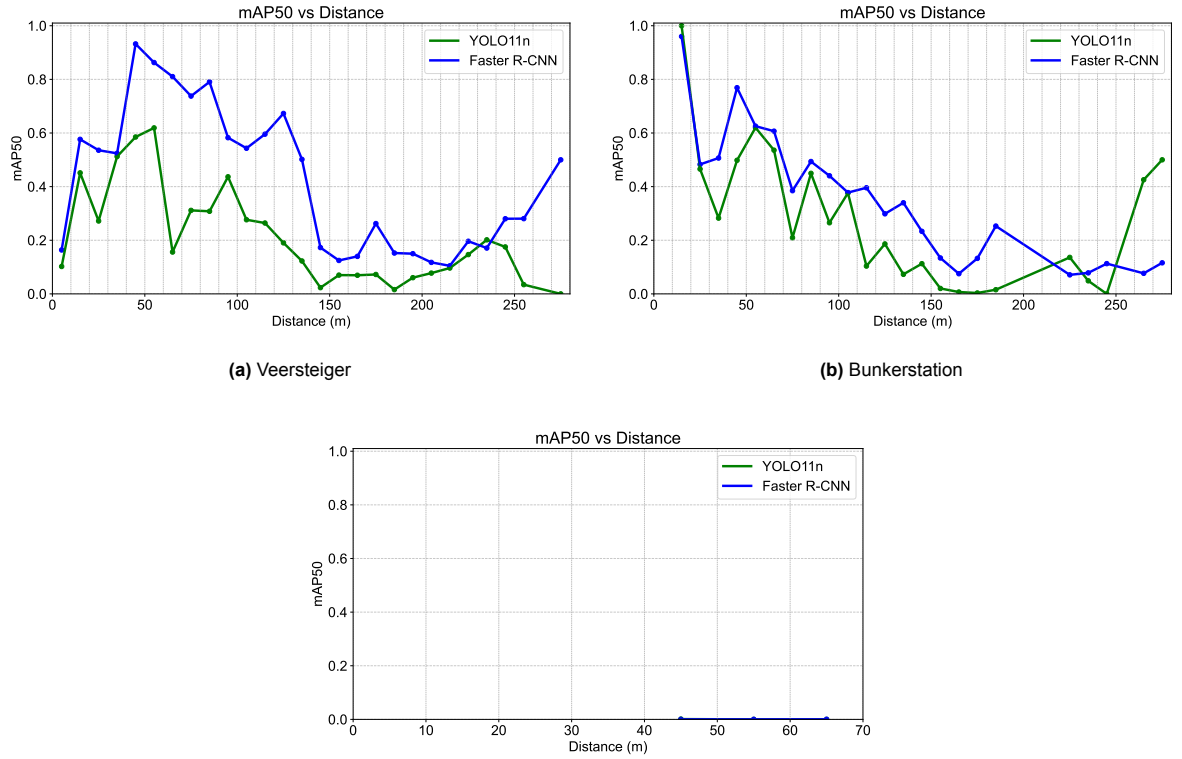
The resulting metric values of this experiment are shown per test dock and as averages in Table 4.4. What is immediately noticeable is that all metric values are much lower than in the baseline experiment. However, similar to the baseline experiment, Faster R-CNN still outperforms YOLO11n overall. Precision, recall, and F1 again show a clear preference for Faster R-CNN when it comes to detection performance. These values are also determined based on the optimal F1 score for both DNNs. The metrics that are independent of the confidence threshold are also clearly in favor of Faster R-CNN. Both mAP50 and mAP50-95 are more than twice as high as those of YOLO11n. It can be seen that the standard values here are in favor of YOLO11n. A closer look at the values on the individual test sets shows that this is caused by Private dock Middeldiep. Both models are almost unable to achieve good detection on this test data, resulting in metric values that are rounded to 0.

From the graphs of the mAP50 as a function of the distance to the dock in Figure 4.5, it can also be concluded that the value of this metric is 0 for all distances to the Private dock. In addition, from the plots of the other test sets it can be concluded that, just like with the baseline experiment, the performance of both DNNs decreases with distance. However, as the table with metrics already indicated, the performance of Faster R-CNN is again better at almost all distances. Finally, Faster R-CNN shows a marked improvement on the Veersteiger data at the long range. YOLO11n demonstrates a similar performance boost on the Bunkerstation data. However, these detections of YOLO11n have a very low confidence at large distances.

In addition to the metrics across the entire datasets, looking at the qualitative results for patterns can be very useful. Figure 4.6 provides two frames for all three of the datasets, including detections and ground truths. For the Veersteiger, it can be seen that the dock is found and detected by both DNNs at close range. Both DNNs show false positives with very low confidence scores. Interestingly, both YOLO11n and Faster R-CNN predict that there are two docks next to each other. This is not only the case in this frame, but it occurs frequently in this dataset. The same phenomenon is visible at a larger range. Here we can see that many of the predictions have a pole on one or two sides, including the false positives on the left side of the dock with a confidence of 0.49 and 0.01. The Bunkerstation data is the only dataset where two docks can be seen in one frame. The qualitative results show that only the right dock is detected. The left dock is hardly detected in any frame. Only once Faster R-CNN detects it with a confidence below 0.05. Faster R-CNN also shows a false positive of more than 0.5. In contrast to the Veersteiger data, the confidence scores here are much lower, resulting in false positives being in the same confidence range as the true positives. The private dock is also visually very different, and as the metrics have shown, the DNNs are not able to make valid detections. Faster R-CNN is somewhat close with detecting the correct quay in all frames. YOLO11n makes hardly any predictions for this dataset, and most of these predictions are incorrect.

Table 4.4: Generalization results for YOLO11n and Faster R-CNN across three test datasets, including mean. Metrics computed at an IoU threshold of 0.5 and an optimal confidence threshold of 0.29 for Faster R-CNN and 0.01 for YOLO11n. The mean values are weighted over the total test set.

Model	Test data	IoU	Precision	Recall	F1	mAP50	mAP50-95
Faster R-CNN	Veersteiger Boven Hardinxveld	0.648	0.409	0.595	0.485	0.402	0.106
	Bunkerstation Dekker & Stam	0.682	0.523	0.313	0.391	0.332	0.133
	Private dock Middeldiep	0.626	0.000	0.000	0.000	0.000	0.000
	Mean	**0.661**	**0.434**	**0.461**	**0.447**	**0.357**	**0.110**
YOLO11n	Veersteiger Boven Hardinxveld	0.622	0.236	0.483	0.317	0.172	0.033
	Bunkerstation Dekker & Stam	0.678	0.266	0.302	0.283	0.187	0.076
	Private dock Middeldiep	0.692	0.000	0.000	0.000	0.000	0.000
	Mean	**0.639**	**0.244**	**0.394**	**0.301**	**0.165**	**0.041**

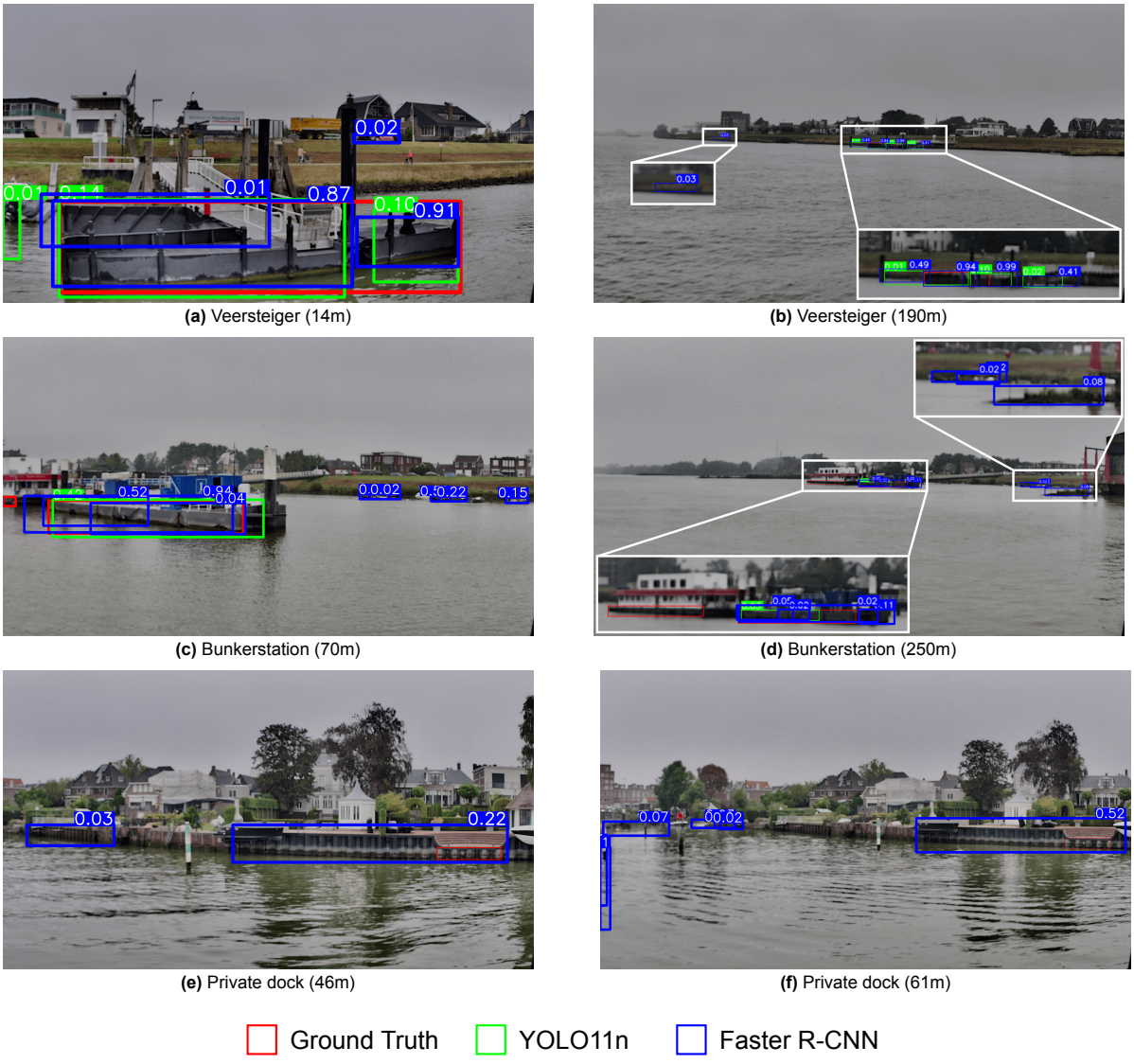


(a) Veersteiger

(b) Bunkerstation

(c) Private dock

Figure 4.5: Mean Average Precision at IoU threshold 0.5 (mAP50) as a function of distance for YOLO11n and Faster R-CNN on all three test sets of the generalization experiment. Showing better performance of Faster R-CNN, a decreasing performance over distance, and a lousy performance on the Private dock.



(a) Veersteiger (14m)

(b) Veersteiger (190m)

(c) Bunkerstation (70m)

(d) Bunkerstation (250m)

(e) Private dock (46m)

(f) Private dock (61m)

Figure 4.6: Predictions at different ranges for the three test docks of the generalization experiment. The values at the top of the bounding boxes represent the confidence scores for those predictions. A confidence threshold of 0.01 is used to generate the images. The white boxes in the images on the right are insets for visibility.

5 Discussion

This chapter reflects on the results by critically evaluating the dataset, methodology, and model performance. The goal is to identify limitations and discuss implications for real-world deployment.

5.1. Dataset Strengths and Limitations

No datasets are found online that focus on dock detection or contain a significant number of docks. The works discussed in Chapter 2 that focus on dock detection using deep learning do not provide any insight into the dataset used and/or only use simulation data. Therefore, a major contribution of this work is creating a high-quality dataset focused on dock detection. This dataset is recorded using an RGB camera from a real vessel and thus contains realistic image data captured from a vessel's point of view. All this data is annotated with a custom annotation tool that allows for interpolation between the frames to speed up this process. This work contains an extensive dataset breakdown, detailing the dataset recording and annotation, and an elaborate overview of the dataset composition.

Creating this dataset has further raised awareness of the value of a rich dataset, especially when dealing with objects that are not easily accessible from the desired point of view. As applied in this work, the most straightforward way is to record the data from a vessel. However, this process is expensive and time-consuming. A vessel can only record a minimal area in a day, leading to limited variation. The limited variation in dock types is therefore one of the most significant limitations of the current dataset. It contains eight different docks, most of which are Waterbus docks. For future work, adding more variety to the dataset is recommended. One way to improve this data collection process could be to use a drone. Drones can be used to simulate a vessel approaching a dock, making the process easier, cheaper, and quicker.

Besides the little variation, the dataset is also not very balanced. The different dock types do not have comparable numbers of frames, distance ranges, and angle ranges. Additionally, the dataset is recorded on a single day and therefore does not contain a balanced range of environmental conditions, such as precipitation and lighting. Imbalance has adverse effects on detection performance [33]. Therefore, a recommendation is to examine the balance in dock types, distances, angles, backgrounds, and environmental conditions when the dataset is expanded in future work.

When expanding the dataset, thinking carefully about the desired detection distance is also advisable. This work has shown that DNNs have more difficulty detecting docks as the distance increases, as Figure 4.3 shows. This is probably partly due to the downscaling step in preprocessing. When it is necessary to detect docks at larger distances reliably, DNNs will have to be tested with larger input images. This has negative consequences for the computational cost and inference speed. When this is not necessary, as in this work, these frames can be omitted from the dataset to save annotation and computation costs.

As mentioned earlier, the custom annotation tool with interpolation has ensured that the annotation process has been significantly accelerated. This tool allows for annotating frames at a speed of 3000

per hour. It is good to realize that this amounts to only 200 seconds of recording with a recording rate of 15 Hz. For future work, it is therefore recommended that this recording rate be reduced rigorously when the number of docks is expanded. Having a few recordings with higher frame rates for visualization is useful.

5.2. Training and Evalutation Methodology

This work compares the DNNs using the same data with almost identical frameworks. This ensures the elimination of most confounding variables. Therefore, differences in performance can be attributed to the difference in network architecture between YOLO11n and Faster R-CNN. The only difference between the training frameworks is the data augmentation. The mosaic augmentation is not used for Faster R-CNN because of the RPN. Future work could investigate the correctness of this by applying this augmentation to Faster R-CNN and comparing the results. For a correct comparison, using similar frameworks is standard practice. The DNNs will eventually need to be further optimized before they can be applied. This could affect the results presented in this work. Future work can therefore benefit from comparing individually optimized DNNs.

The cross-validation used has proven to be a valuable addition, since the results on the different data splits show substantial variations. Averaging the performance over the data splits provides a clear and fair picture of the performance. The choice not to use a fixed confidence threshold for generating the metrics is also correct. Both DNNs have a different way of determining the confidence score and can therefore not be measured on the same scale. For example, the results show that Faster R-CNN produces higher average confidence scores than YOLO11n. The metrics are fairly comparable by optimizing this confidence threshold for both DNNs separately.

5.3. DNN Behavior and Performance

The baseline comparison shows a significantly better detection performance of Faster R-CNN. All metrics presented in Table 4.2 favor Faster R-CNN except for IoU. Additionally, YOLO11n shows a higher inference speed of 37 FPS compared to the 17 FPS of Faster R-CNN.

A deeper dive into the results shows that the performance of both DNNs decreases with increasing distance to the dock. This decrease mainly occurs in the range of over 100 meters, as seen in Figure 4.3. Figure 4.1 shows that the number of frames decreases as the distance to the dock increases. However, the decrease in performance does not directly correspond to the reduction in the number of frames. Splits 1 and 2 in particular show poor performance between 150 and 200 meters, while the dataset still contains more than enough frames here. Subsequently, both models show a performance spike beyond 200 meters, while the dataset contains less data in this range. All in all, it is difficult to conclude from these results whether the DNNs perform less well on docks that occupy a small number of pixels in the image or whether this is caused by the smaller amount of data. If the DNNs actually perform worse on these small appearing docks, it could be tested whether this improves with a larger input resolution. However, this results in a higher computational cost. These results do show that Faster R-CNN performs better at almost all distances. At short range up to 100-150 meters, the performance is excellent, but also beyond 200 meters, the performance is significantly better than YOLO11n.

A striking difference is observable in the different data splits in the baseline comparison. The first two splits perform significantly worse than the last two for both DNNs. Because all training and evaluation settings are the same, the main difference can be found in the dataset. The test set is the same for all splits, so the difference here is in the train and especially the validation set. A different performance of Split 2 can be due to its validation set, since Bunkerstation Dekker & Stam deviates in appearance from the other docks. The poor performance on Split 1 is more unexpected since the common Waterbus dock is used for the validation. There are some similar differences between the first and last two data splits. First, the first two splits use only one dock for validation. Second, the validation sets of the first two splits are larger and the train sets are therefore slightly smaller. Third, Figure 4.1 shows that the validation sets of the first two splits show a similar distribution as the train and test splits. The validation sets of the last two splits show peaks at 50-100 and 200-250 meters. So there are multiple differences between the first and last two data splits that could cause this performance difference, but concise reasoning can not be given from these results.

This difference between the data splits raises the question of whether the DNNs are robust to changes and whether they can generalize. The latter is investigated in the second experiment. This experiment shows that both DNNs have trouble detecting unseen dock types. The performance of both DNNs is significantly lower than that of the baseline comparison. However, Faster R-CNN again performs substantially better than YOLO11n on all metrics.

The relatively good performance on Veersteiger (Table 4.4) is due to the appearance that is similar to that of the Waterbus docks in terms of geometry and colors. Qualitative results reveal that both DNNs frequently predict two adjacent dock instances when, in reality, only one continuous dock is present. The image shows that the predictions partially overlap the correct region but should ideally be merged into a single detection. In almost all these cases, the separation between these detections is precisely at the pole's location in the middle of the dock. A closer look at the dataset shows that all Waterbus docks have two poles at both ends, as seen in Figure 4.4a. Since the entire training set here consists of Waterbus docks, it is plausible that the DNNs have learned this relationship. However, not all dock types have a pole on both sides, which makes the models less generalizable. This could also explain why the results on Bunkerstation in this experiment are much better than those on the Private dock, despite both visually deviating from the Waterbus docks. As shown in Figure 4.6c, this dock has the same poles on both sides. This again underlines the importance of a balanced and varied dataset for future work, so that no specific patterns are learned that only occur for some object types.

5.4. Real-World Deployment

For real-world deployment, both models have their applications. Based on the presented results, YOLO11n is preferred for its detection speed. For applications where it is crucial to generate real-time detections for cameras with a frame rate of over 30 Hz, YOLO11n should be chosen. For the use case presented in this work, Faster R-CNN will be the best option based on the presented results. This DNN shows significantly better detection results. In addition, the speeds in the maritime environment are relatively low in general, making a detection speed of 17 FPS sufficient. This detection speed can be further increased by using a more dedicated computer. It is important to note here that further improvements and optimizations can further improve the performance of both DNNs.

As indicated several times, the Dordrecht Dock Dataset is not sufficient for training a general dock detector. This would require several adjustments to be made. First, it needs more variety. Additional dock types and environmental conditions should be added to ensure the ability to detect different types in various conditions. Second, it is important that this dataset is better balanced. A specific type of dock or environmental condition taking up a large part of the dataset can lead to less generalization power.

In addition, the DNNs themselves could be further optimized for this application. Both of these DNNs offer adjustment options to the architecture. For example, other versions and other complexities can be tested with YOLO. For Faster R-CNN, other types of backbones or a different backbone complexity can be considered. Furthermore, numerous adjustments can be applied to the training procedure. The training cycle also offers a lot of parameters to vary, such as the number of epochs, the optimizer, and the learning rate. Analyzing all these adjustments is very time-consuming, but is recommended for future work.

Future research could investigate different DNNs or specific improvements. For example, the state-of-the-art Vision Transformer [8] shows excellent performance. In addition, the application of tracking [20] could be examined. At present, the object is detected individually in each frame. The detection performance might be improved by using the position of the objects in the previous frames.

It is possible that even these suggested improvements will not lead to a dock detector that can detect all types of docks in different conditions. In this case, a different solution will have to be found. A possible solution is to create region-based dock detectors. These detectors will be trained on the docks in a specific region and/or type, as has been done in this work. This work shows that without an extensive dataset and optimization, the DNNs can detect the Waterbus docks reliably. The autonomous system should then switch between the region-based detectors based on GPS data for optimal performance.

6

Conclusion

This work aimed to identify the extent to which maritime docks can be reliably detected using a camera. The Dordrecht Dock Dataset was created using a custom interpolation annotator to analyze this. Subsequently, a consistent training and evaluation framework was set up to perform a comparative study between Faster R-CNN and YOLO11n. The main finding is that Faster R-CNN shows the best detection quality despite its lower FPS. However, both DNNs show difficulty with larger distances and generalization. From this, it can be concluded that it is possible to reliably detect docks within a short range (up to 100 meters) on docks that are comparable to the dataset used.

The limitations are mainly in the dataset and the non-optimized DNNs. The dataset includes an insufficient number of distinct docks, lacking both diversity and balance. In addition, the analyzed DNNs are not fully optimized because the emphasis was on an equal comparison. Recommendations for future research are to improve both of these limitations. A dataset with more balance and variation in dock types and environmental conditions will presumably show better generalization to unseen docks. In addition, the performances can probably be further improved by optimizing the DNNs for this application. This can be done by fine-tuning both the architecture and the training procedure. In addition to these improvements, it can also be valuable to look at newer techniques such as tracking and Vision Transformers.

Overall, this work demonstrates the potential of camera-based systems for autonomous docking, though further progress is needed in developing a comprehensive dataset and refining the detection algorithms.

References

[1] Peter Joseph Agcanas et al. “University of Hawai’i at Manoa: Team Kanaloa”. In: (2018).

[2] Seyed Majid Azimi et al. “Towards Multi-class Object Detection in Unconstrained Remote Sensing Imagery”. In: *Computer Vision – ACCV 2018*. 2019, pp. 150–165. DOI: 10.1007/978-3-030-20893-6_10.

[3] Alex Bewley et al. “Simple Online and Realtime Tracking”. In: *2016 IEEE International Conference on Image Processing (ICIP)*. Sept. 2016, pp. 3464–3468. DOI: 10.1109/ICIP.2016.7533003.

[4] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. Apr. 23, 2020. DOI: 10.48550/arXiv.2004.10934.

[5] Nicolas Carion et al. *End-to-End Object Detection with Transformers*. May 28, 2020. DOI: 10.48550/arXiv.2005.12872.

[6] Yong-Woon Choi et al. “Real-time Detection Technique of the Target in a Berth for Automatic Ship Berthing”. In: *Journal of Control, Automation and Systems Engineering* 12.5 (May 1, 2006), pp. 431–437. ISSN: 1225-9845. DOI: 10.5302/J.ICROS.2006.12.5.431.

[7] Lars Digerud et al. “Vision-based positioning of Unmanned Surface Vehicles using Fiducial Markers for automatic docking”. In: *IFAC-PapersOnLine*. 14th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles CAMS 2022 55.31 (Jan. 1, 2022), pp. 78–84. ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2022.10.412.

[8] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. June 3, 2021. DOI: 10.48550/arXiv.2010.11929.

[9] Meng Joo Er et al. “Ship detection with deep learning: a survey”. In: *Artificial Intelligence Review* 56.10 (Oct. 1, 2023), pp. 11825–11865. ISSN: 1573-7462. DOI: 10.1007/s10462-023-10455-x.

[10] Alvaro Fuentes et al. “A Robust Deep-Learning-Based Detector for Real-Time Tomato Plant Diseases and Pests Recognition”. In: *Sensors* 17.9 (2017), p. 2022. DOI: 10.3390/s17092022.

[11] Kunihiko Fukushima. “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position”. In: *Biological Cybernetics* 36.4 (Apr. 1, 1980), pp. 193–202. ISSN: 1432-0770. DOI: 10.1007/BF00344251.

[12] Ross Girshick. *Fast R-CNN*. Sept. 27, 2015. DOI: 10.48550/arXiv.1504.08083.

[13] Ross Girshick et al. *Rich feature hierarchies for accurate object detection and semantic segmentation*. Oct. 22, 2014. DOI: 10.48550/arXiv.1311.2524.

[14] *GNU Affero General Public License - GNU-project - Free Software Foundation*. URL: https://www.gnu.org/licenses/agpl-3.0.html (visited on 05/02/2025).

[15] Guowen He et al. “An Improved YOLO v4 Algorithm-based Object Detection Method for Maritime Vessels”. In: *International Journal of Science and Engineering Applications* 11 (Apr. 1, 2022), pp. 50–55. DOI: 10.7753/IJSEA1104.1001.

[16] Kaiming He et al. *Deep Residual Learning for Image Recognition*. Dec. 10, 2015. DOI: 10.48550/arXiv.1512.03385.

[17] Glenn Jocher. *YOLOv5 by Ultralytics*. Version 7.0. May 29, 2020.

[18] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. *Ultralytics YOLOv8*. Version 8.0.0. 2023.

[19] Glenn Jocher and Jing Qiu. *Ultralytics YOLO11*. Version 11.0.0. 2024.

[20] Pushkar Kadam, Gu Fang, and Ju Jia Zou. “Object Tracking Using Computer Vision: A Review”. In: *Computers* 13.6 (June 2024). Number: 6 Publisher: Multidisciplinary Digital Publishing Institute, p. 136. ISSN: 2073-431X. DOI: 10.3390/computers13060136.

[21] V Kastrinaki, M Zervakis, and K Kalaitzakis. “A survey of video processing techniques for traffic applications”. In: *Image and Vision Computing* 21.4 (2003), pp. 359–381. DOI: 10.1016/s0262-8856(03)00004-0.

[22] Parvinder Kaur, Baljit Singh Khehra, and Er. Bhupinder Singh Mavi. “Data Augmentation for Object Detection: A Review”. In: 2021 IEEE International Midwest Symposium on Circuits and Systems (MWSCAS). ISSN: 1558-3899. Aug. 2021, pp. 537–543. DOI: 10.1109/MWSCAS47672.2021.9531849.

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *ResearchGate* (Jan. 2012). DOI: 10.1145/3065386.

[24] Jason Ku, Alex D. Pon, and Steven L. Waslander. “Monocular 3D Object Detection Leveraging Accurate Proposals and Shape Reconstruction”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 11859–11868. DOI: 10.1109/cvpr.2019.01214.

[25] *LabelMe*. Version 3.0. URL: http://labelme.csail.mit.edu/Release3.0/.

[26] Jooho Lee, Joohyun Woo, and Nakwan Kim. “Vision and 2D LiDAR based autonomous surface vehicle docking for identify symbols and dock task in 2016 Maritime RobotX Challenge”. In: 2017 IEEE Underwater Technology (UT). Feb. 2017, pp. 1–5. DOI: 10.1109/UT.2017.7890273.

[27] Pedro Nuno Barbosa Leite. “A Self-Guided Docking Architecture for Autonomous Surface Vehicles”. In: (July 17, 2019).

[28] Simon J. N. Lexau, Morten Breivik, and Anastasios M. Lekkas. “Automated Docking for Marine Surface Vessels—A Survey”. In: 2023.

[29] Min Li et al. “Agricultural Greenhouses Detection in High-Resolution Satellite Images Based on Convolutional Neural Networks: Comparison of Faster R-CNN, YOLO v3 and SSD”. In: *Sensors* 20.17 (Jan. 2020), p. 4938. ISSN: 1424-8220. DOI: 10.3390/s20174938.

[30] *LI-AR0233-GW5400-FPDLINKIII-060H*. Leopard Imaging Inc. URL: https://leopardimaging.com/product/automotive-cameras/cameras-by-interface/ti-ftplink-iii-cameras-cameras-by-interface/li-ar0233-gw5400-fpdlinkiii/li-ar0233-gw5400-fpdlinkiii-060h/ (visited on 03/17/2025).

[31] Haitong Lou et al. “DC-YOLOv8: Small-Size Object Detection Algorithm Based on Camera Sensor”. In: *Electronics* 12.10 (2023), p. 2323. DOI: 10.3390/electronics12102323.

[32] Jacinto C. Nascimento and Jorge S. Marques. “Performance Evaluation of Object Detection Algorithms for Video Surveillance”. In: *IEEE Transactions on Multimedia* 8.4 (2006), pp. 761–774. DOI: 10.1109/tmm.2006.876287.

[33] Kemal Oksuz et al. *Imbalance Problems in Object Detection: A Review*. Mar. 19, 2020. URL: https://ieeexplore-ieee-org.tudelft.idm.oclc.org/document/9042296 (visited on 05/07/2025).

[34] Noran S. Ouf. “Leguminous seeds detection based on convolutional neural networks: Comparison of Faster R-CNN and YOLOv4 on a small custom dataset”. In: *Artificial Intelligence in Agriculture* 8 (June 1, 2023), pp. 30–45. ISSN: 2589-7217. DOI: 10.1016/j.aiia.2023.03.002.

[35] Peter Ranacher et al. *Why GPS makes distances bigger than they are*. July 14, 2015. DOI: 10.48550/arXiv.1504.04504. arXiv: 1504.04504[physics].

[36] Joseph Redmon and Ali Farhadi. *YOLOv3: An Incremental Improvement*. arXiv.org. Apr. 8, 2018. URL: https://arxiv.org/abs/1804.02767v1 (visited on 05/14/2025).

[37] Joseph Redmon et al. *You Only Look Once: Unified, Real-Time Object Detection*. May 9, 2016. DOI: 10.48550/arXiv.1506.02640. arXiv: 1506.02640[cs].

[38] Shaoqing Ren et al. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. Jan. 6, 2016. DOI: 10.48550/arXiv.1506.01497.

[39] Paul Robinette et al. “Sensor Evaluation for Autonomous Surface Vehicles in Inland Waterways”. In: OCEANS 2019 - Marseille. June 2019, pp. 1–8. DOI: 10.1109/OCEANSE.2019.8867468.

[40] *Roboflow Annotate*. URL: https://roboflow.com/annotate. 2025.

[41] *RobotX*. RobotX. URL: https://robotx.org/ (visited on 09/14/2024).

[42] Safa Mohammed Sali et al. “A Review on Object Detection Algorithms for Ship Detection”. In: 2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS). Vol. 1. ISSN: 2575-7288. Mar. 2021, pp. 1–5. DOI: 10.1109/ICACCS51430.2021.9441801.

[43] Leo Stanislas and Matthew Dunbabin. “Multimodal Sensor Fusion for Robust Obstacle Detection and Classification in the Maritime RobotX Challenge”. In: *IEEE Journal of Oceanic Engineering* 44.2 (Apr. 2019), pp. 343–351. ISSN: 1558-1691. DOI: 10.1109/JOE.2018.2868488.

[44] Statista. *Frachtvolumen weltweit nach Verkehrsträgern*. Statista. 2015. URL: https://de-statista-com.tudelft.idm.oclc.org/statistik/daten/studie/482955/umfrage/frachtvolumen-weltweit-nach-verkehrstraegern/ (visited on 04/10/2025).

[45] Li Su et al. “A survey of maritime vision datasets”. In: *Multimedia Tools and Applications* 82.19 (Aug. 1, 2023). Number: 19, pp. 28873–28893. ISSN: 1573-7721. DOI: 10.1007/s11042-023-14756-9.

[46] Muhammad Fasih Tariq and Muhammad Azeem Javed. *Small Object Detection with YOLO: A Performance Analysis Across Model Versions and Hardware*. Apr. 14, 2025. URL: https://arxiv.org/html/2504.09900v1?utm_source=chatgpt.com (visited on 04/25/2025).

[47] *The 3-Clause BSD License*. Open Source Initiative. URL: https://opensource.org/license/bsd-3-clause (visited on 05/02/2025).

[48] Paschalis Tsirtsakis et al. “Deep learning for object recognition: A comprehensive review of models and algorithms”. In: *International Journal of Cognitive Computing in Engineering* 6 (Dec. 1, 2025), pp. 298–312. ISSN: 2666-3074. DOI: 10.1016/j.ijcce.2025.01.004.

[49] Tzutalin. *LabelImg*. URL: https://github.com/HumanSignal/labelImg. 2015.

[50] Unox. *Autonomous Ships: The Future of Maritime Transport*. Oct. 2024. URL: https://www.unoks.com/en/blog/autonomous-ships-the-future-of-maritime-transport (visited on 04/10/2025).

[51] Øystein Volden, Annette Stahl, and Thor I. Fossen. “Vision-based positioning system for auto-docking of unmanned surface vehicles (USVs)”. In: *International Journal of Intelligent Robotics and Applications* 6.1 (Mar. 1, 2022), pp. 86–103. ISSN: 2366-598X. DOI: 10.1007/s41315-021-00193-0.

[52] Wei Wang and Xiangfeng Luo. “Autonomous Docking of the USV using Deep Reinforcement Learning Combine with Observation Enhanced”. In: 2021 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA). Aug. 2021, pp. 992–996. DOI: 10.1109/AEECA52519.2021.9574371.

[53] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. *Simple Online and Realtime Tracking with a Deep Association Metric*. Mar. 21, 2017. DOI: 10.48550/arXiv.1703.07402.

[54] Xue Ying. “An Overview of Overfitting and its Solutions”. In: *Journal of Physics: Conference Series* 1168.2 (Feb. 2019). Publisher: IOP Publishing, p. 022022. ISSN: 1742-6596. DOI: 10.1088/1742-6596/1168/2/022022.