Image Analysis using Machine Learning Methods for Electronic Outlook on board Ships

Eduardo Vázquez Salvador (s151464)

Master of Science in Engineering

2018

DTU Electrical Engineering
Department of Electrical Engineering

Image Analysis using Machine Learning Methods for Electronic Outlook on board Ships, Master thesis

Report written by:
Eduardo Vázquez Salvador (s151464)

Advisor(s):
Mogens Blanke, Professor at the Department of Electrical Engineering, DTU
Søren Hansen, Associate Professor at the Department of Electrical Engineering, DTU
Jonathan Dyssel Stets, Postdoc at the Department of Applied Mathematics and Computer Science, DTU

DTU Electrical Engineering
Technical University of Denmark
2800 Kgs. Lyngby
Denmark

elektro@elektro.dtu.dk

Project period: 22. January- 13. July

ECTS: 30

Education: M.Sc.

Field: Electrical Engineering

Class: Public

Edition: 1. edition

Remarks: This report is submitted as partial fulfillment of the requirements for graduation in the above education at the Technical University of Denmark.

Abstract

The project presented in this paper develops four different types of object detection methods. It is part of a larger project currently under development at the DTU Department of Electrical Engineering for developing autonomous situation awareness at maritime vessels. An object detector localizes and classifies objects on digital images. The four object detectors are implemented through various steps: first, they are trained; then, they are applied to a series of image datasets. Detection performances of the methods implemented are evaluated and compared to one another through results obtained from the application of the detectors to the aforementioned image sets.

Preface

This Master thesis was prepared at the department of Electrical Engineering at the Technical University of Denmark in fulfillment of the requirements for acquiring a Master degree in Electrical Engineering. It took place between January 2018 and July 2018, and represented a workload of 30 ECTS.

I would like to thank my supervisors Mogens Blanke, Søren Hansen and Jonathan Dyssel for their supervision and guidance. Their help has proven very useful during the whole development of the project. I would also like to thank Juan Molla, a fellow student whose thesis was closely related to this one, for his collaboration and contribution to the project.

I want to thank the great friends I was lucky enough to meet during my master studies in Denmark, who have made this period an enjoyable and enriching experience. Last but not least, I want to thank my parents and my brother, for their kindness, their empathy and support. It has been a long journey for all of us, and I wouldn't be writing these lines if it wasn't for them.

Contents

List of Figures

List of Tables

Acronyms

4IR Fourth Industrial Revolution.

ACF Aggregate Channel Features.

AdaBoost Adaptative Boosting.

aDT average Detection Time.

AIS Automatic Identification System.

ANN Artificial Neural Network.

AP Average Precision.

CCD Charge Coupled Device.

CMOS Complementary Metal Oxide On Silicon.

CNN Convolutional Neural Network.

COLREGs International Regulations for Preventing Collisions at Sea.

DoG Difference of Gaussian.

ENC Electronic Nautical Chart.

FN False Negative.

FP False Positive.

GMM Gaussian Mixture Model.

GPS Global Positioning System.

GT Ground Truth.

HOG Histogram of Oriented Gradients.

HSV Hue Saturation Value.

ILSVRC ImageNet Large Scale Visual Recognition Challenge.

IMO International Maritime Organization.

IMU Inertial Measurement Unit.

IoT Internet of Things.

IoU Intersection over Union.

JPEG Joint Photographic Experts Group.

LoG Laplacian of Gaussian.

mAP mean Average Precision.

MPC Model Predictive Control.

MSER Maximally Stable Extremal Regions.

NIR Near Infra-Red.

NN Neural Network.

PNG Portable Network Graphics.

RCNN Region-based Convolutional Neural Network.

ReLU Rectified Linear Unit.

RGB Red, Green, Blue.

ROI Region of Interest.

RPN Region Proposal Network.

SIFT Scale Invariant Feature Transform.

SURF Speeded Up Robust Features.

SVM Support Vector Machine.

TIFF Tagged Image File Format.

TP True Positive.

UAV Unmanned Aerial Vehicle.

USV Unmanned Surface Vehicle.

VGG Visual Geometry Group.

Glossary

Aggregate Channel Features Supervised learning model that extracts features from different channels of images for detection purposes.

Classification Determination of the category of an object.

Classifier-based detector Detector composed of a ROI finder and a classifier.

Computer vision Field of computer science that focuses on computer systems obtaining a visual understanding of the physical world.

Convolutional Neural Network Machine learning task where a function is inferred from just input data (no corresponding output data).

Detection Localization and classification of an object.

Digital image processing Group of tools that can be applied to digital images in order to obtain modified images that possess enhanced qualities, such that image appearance is improved or the extraction of information is facilitated.

False Negative A ground truth object is not detected.

False Postive A detected object does not correspond to any ground truth object.

Ground Truth Proper information to which compare detection results, obtained by direct observation.

Image analysis Extraction of meaningful information from images.

Localization Determination of the position of an object in a given reference frame.

Machine learning Field of computer science that aims at computer systems to progressively acquire knowledge from data, without that knowledge being explicitly programmed.

Precision Fraction of detected objects that correspond to Ground Truth information.

Pure detector Detector where detection takes place, for implementation purposes, in one step.

Recall fraction of Ground Truth objects correctly detected.

Region-based Convolutional Neural Network CNN applied to object detection.

Regions Of Interest Areas of an image whose color and/or texture differs from that of their background.

Supervised learning Machine learning task where a function is shaped by inference from data in the form of input-output pairs.

True Positive A ground truth object is correcly detected.

Unsupervised learning Machine learning task where a function is inferred from just input data (no corresponding output data).

CHAPTER 1

Introduction

1.1 Motivation

Automation is reshaping the way in which many industries operate. Its impact even goes beyond mere economic aspects, since it is already affecting the way advanced societies perceive technology and themselves. Furthermore, current trends in the adoption of automated solutions are expected to grow in the near future, presenting challenges and opportunities that today are difficult to foresee. Automation is part of a wave of technological advancements, that includes renewable energies, nano and biotechnology, quantum computing, the Internet of Things (IoT) and some others, and have been labeled as the Fourth Industrial Revolution (4IR).

Automation can be defined as a technology that allows processes to be performed without human intervention. It is currently mostly used in certain limited industrial activities, that tend to be composed of heavy and repetitive tasks. A classical example would be some steps of vehicle manufacturing. The motivation for the adoption of automation in such scenarios is the improved productivity (automated operations are generally faster and more accurate than those performed by workers), cost reduction (lower operational costs generated by a smaller labor force) and safety improvements for workers (specially in certain industries where heavy machinery or dangerous chemical products are used). The potential benefits of automation for many industrial sectors where its implementation is either not existent or still very limited are producing considerable investment and research in new methods for adapting it to the specific requirements of such industries. One of those industries is transport.

Transportation of people and goods constitute one of the most vital economic activities in the world today. For people, the appearance within the last decades of affordable transoceanic flights, together with high speed railway systems and mass transit systems facilitated the reallocation of many students and workers, as well as an immense growth in tourism. Regarding goods, their production and consumption has become global. That requires specific transport systems, that are able to carry large amounts of goods in a cheap and reliable manner. As of today, cargo ships comply with those requirements, and are responsible for carrying most traded goods in the world.

Autonomous vehicles are means of transport where navigation (understood as the pro-

cess of monitoring and controlling the movement of a vehicle) takes place autonomously, that is, without human assistance. Precursors of autonomous vehicles are the autopilot systems used in aircrafts and automobiles, that assist, rather than substitute, pilots and drivers, respectively. Examples of fully automated vehicles are autonomous cars, automated metro systems (such as Copenhagen metro), Unmanned Aerial Vehicle (UAV) and Unmanned Surface Vehicle (USV). Main advantages of adopting this type of technology are cost reduction (no workers needed), fuel consumption efficiency, enhanced passenger comfort and especially, improved safety (an autonomous vehicle, unlike a human, is infallibly monitoring its environment and taking measures to avoid collisions).

This thesis is part of a project where an autopilot system for assisting the pilot of a vessel, or surface vehicle, is being developed. Eventually, an autonomous system should be able to replace the pilot (thus effectively becoming an USV). That stage is, at this moment, not within reach, and the efforts of the members of the project, including this student, focus on providing a useful, reliable, intuitive and real time navigation aid for the pilot controlling the vessel. The autopilot system would gather data from the vessel and its surroundings and propose the most efficient route and speed to the pilot, that would either let the system take control of the vehicle or contrast the information with his own visual information and knowledge of maritime navigation and perform the navigation tasks himself.

The advantages of implementing this system are in line with the ones presented above for autonomous vehicles in general, and include energy efficiency (as the optimal route and speed would be chosen) and enhanced safety (the autopilot algorithm allows for obstacle avoidance and can reduce the amount of information the pilot needs to focus on while navigating). The USV system would perform a safe navigation, using certified Electronic Nautical Chart (ENC) and complying with international navigation rules determined by the International Maritime Organization (IMO), in particular the International Regulations for Preventing Collisions at Sea (COLREGs).

From the description of its functionality, it is clear that the USV system needs to know location, orientation and velocity of the vessel, as well as position and velocity of other floating objects (sea marks or other vessels) in its surroundings (all that information is encompassed by the concept of situation awareness), in order to determine the optimal trajectory and speed while avoiding collision. Sensors such as Inertial Measurement Unit (IMU) devices, Global Positioning System (GPS) devices, digital cameras and radar systems, combined with algorithms that extract and fuse information from the data collected by those sensors are used to obtain that crucial information. This thesis focuses on the detection of objects in digital images. The detection process allows to determine the location of objects around the vessel (in pixel coordinates) and to categorize them. Object detection therefore provides very relevant information regarding situation awareness, but it is necessary to complement it with information from other

sensors.

1.2 Scope of the Project

The aim of the project developed by the student is to detect certain types of objects on digital images. Object detection implies first localizing all the objects present on an image and then classifying them within a certain set of categories. Four different detection methods have been applied to the same images, using various technologies, such that it is possible to compare their performances. For every detection method, an initial stage of literature review was carried out, followed by the implementation of the method in MatLab. Finally, the methods were tested and detection quality results obtained for each of them and compared.

It is beyond the scope of the project to fuse the information extracted from digital images with information obtained from processing data from other sources (namely, a radar). That was not initially the case, since the project was outlined at the beginning of the project as consisting on a sensor fusion algorithm, that would include the object detection part (that is finally the object of the thesis), coordinate transformation between image and radar coordinates, and radar information extraction. Due to delays in the availability of radar data, the sensor fusion algorithm was not implemented. Instead, it is treated theoretically in section 3.3.

It is also beyond the scope of the project the implementation of a real time object detector, that could be applied in actual time to the images being taken by cameras during experimentation to extract object information from them. The approach taken instead is more related to research than actual application, and consists on gathering images previously taken, applying different detectors to them, extracting results and comparing them. The software used, MatLab, would not be suitable for a real time implementation, but is very convenient for research and purely academic purposes.

Part of the implementation of two of the detection methods has been carried out by Juan Molla, a fellow student at DTU. At the beginning of the section that covers the algorithm developed by Juan, it is clearly stated that such part of the project was not developed by this student but by him.

1.3 Outline of the Thesis

This document reflects the knowledge acquired by the student during the development of the thesis, as well as implementation aspects and results derived from that implementation. The document is organized in 6 chapters, including introduction and conclusion.

The chapters are numbered 1-6. Each chapter (except introduction and conclusion) starts with a brief description of the content of the chapter and an outline of its sections, and ends with a summary of its most relevant information. The following is a short description of the content of each of the remaining chapters:

- Chapter 2 introduces the basic physical and digital components necessary for gathering data and analyzing it within the context of the thesis.
- Chapter 3 covers, from a theoretical approach, the perception module that encompasses sensor fusion and object detection.
- Chapter 4 extensively treats the detection methods employed in the thesis, from their theoretical foundations to the results obtained from their application in tests.
- Chapter 5 presents the experimental test performed using images taken by the cameras that are part of the equipment used in the project, and the results obtained from those tests.
- Chapter 6 concludes the document. The most relevant findings of the project are summarized, and paths for future work on the topic are outlined.

CHAPTER 2

System Setup

This chapter introduces the physical and digital tools used for perception purposes within the framework of the thesis. The physical equipment is composed of different types of sensors. The software employed to perform the object detection task is MatLab. The detectors implemented using that software were trained with and applied to certain digital image sets that are also covered in this chapter. One of those datasets is composed of images taken by the cameras that are part of the equipment. The chapter is structured in the following sections:

- Section 2.1 covers the physical components of the perception system.
- Section 2.2 covers the software used.
- Section 2.3 lists the various image sets employed for the development of the thesis.

2.1 Physical Equipment

The only physical components considered here are sensors, that are directly involved in capturing data. All the other physical components of the USV system, such as the vessel itself, the computers and actuators, are not treated.

The perceptual system implemented by the DTU team responsible for the development of the USV project consists of the following sensors: visible (color and monochrome) cameras, near infrared cameras, a radar and a motion tracking device that contains, among other sensors, an IMU and a GPS receiver. Data collected by cameras and radar is combined by means of a sensor fusion algorithm (section 3.3) to identify and locate objects in the surroundings of the vessel. The motion tracking device is used to determine position, orientation and velocity of the vessel itself. All the sensors are mounted on top of a metal structure located by the gunwale of the vessel. The exact setup can be seen in picture 2.1.

Note that image 2.1 does not reflect the final version of the sensor system setup, that would be used at the USV, but is rather a provisional version with which tests are

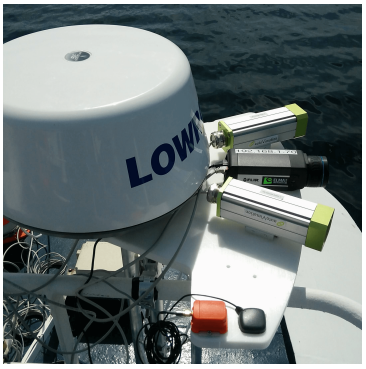
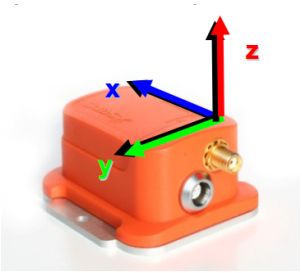


Figure 2.1: Equipment setup

performed. In the final setup, more cameras would be used, placed such that all angles would be covered.

2.1.1 Motion tracking device

In order to determine position, attitude and velocity of the vessel, a single motion tracking device is employed. In particular, the MTi-G-710 model of Xsens. This device includes different sensors, some exteroceptive, such as a GPS receiver, but mostly proprioceptive, like an IMU, accelerometers and gyroscopes. The device combines the data gathered by some of its various sensors through a sensor fusion algorithm and includes a Kalman filter ([Xse]).



Source: [Xse]

Figure 2.2: MTi-G device

2.1.2 Radar

One of the exteroceptive sensors used for gathering data of the surroundings of the vessel is a radar. The term radar is derived from the expression ”radio detection and ranging”, and consists of a device that estimates the relative distance of the objects it detects by means of radio signals. Every object detected by the radar is represented

by range (the distance to the radar) and bearing (the angle to the x axis of the radar). The specific radar model used for in the project is the Lowrance Broadband 3G radar. This radar transmits a frequency modulated continuous wave, that is, a wave with a frequency that changes linearly over time. At a certain moment, the radar emits a signal with a given frequency, that might be reflected on an object, generating an echo that, with that same frequency, is detected by the radar. The difference in frequency between the signal received and the signal being sent by the radar at that moment allows to accurately compute the distance of the object to the radar ([rad]). The Lowrance Broadband 3G radar is particularly accurate when it comes to short-range measures, that is, for detecting objects near the vessel. However, it is also capable of detecting objects as far away as 24 nautical miles ([Low]).

2.1.3 Digital Cameras

Together with the radar, the sensors responsible for detecting objects in the vicinity of the vessel are digital cameras. These type of camera produces images that can be subject to image analysis processes, by which it is possible to extract information out of the data collected. In the context of the project, the information to be obtained from digital images consists of detected objects, that is, objects characterized by their location in image frame and their class. This information can then be combined with that extracted form radar data by means of a sensor fusion algorithm (section 3.3). The specific types of cameras used in the project are now presented. The visible color cameras used are JAI Go-5000C-PGE with Kowa LM12HC lens, with a vertical field of view close to 55 degrees, a horizontal resolution 2560 pixels and a polarizing filter applied. The visible monochrome camera employed is a JAI GO-5000M-PGE with Kowa LM12HC lens and no filter. No data from near infrared camera had been gathered at sea by the time the student handed in this report. The theoretical bases of the images obtained by each of these types of cameras are treated in section 4.1.

2.2 Software

The software used for the implementation of the object detection task is MatLab. The use of this software is very convenient for performing tests, evaluating performance and obtaining plotted results, but is not useful when it comes to real time implementation. Since such implementation is out of the scope of the thesis, and the focus is on the comparison of the performance of various object detection methods, MatLab was a reasonable choice. Furthermore, through a set of MatLab toolboxes, it was possible to apply built-in functions and already developed m-files for training, applying and evaluating all the detection methods employed along the thesis. The specific version of MatLab used is $R2018a$. Earlier versions don't support some of the functions and the Image Labeler app used during the development of the code. MatLab toolboxes used

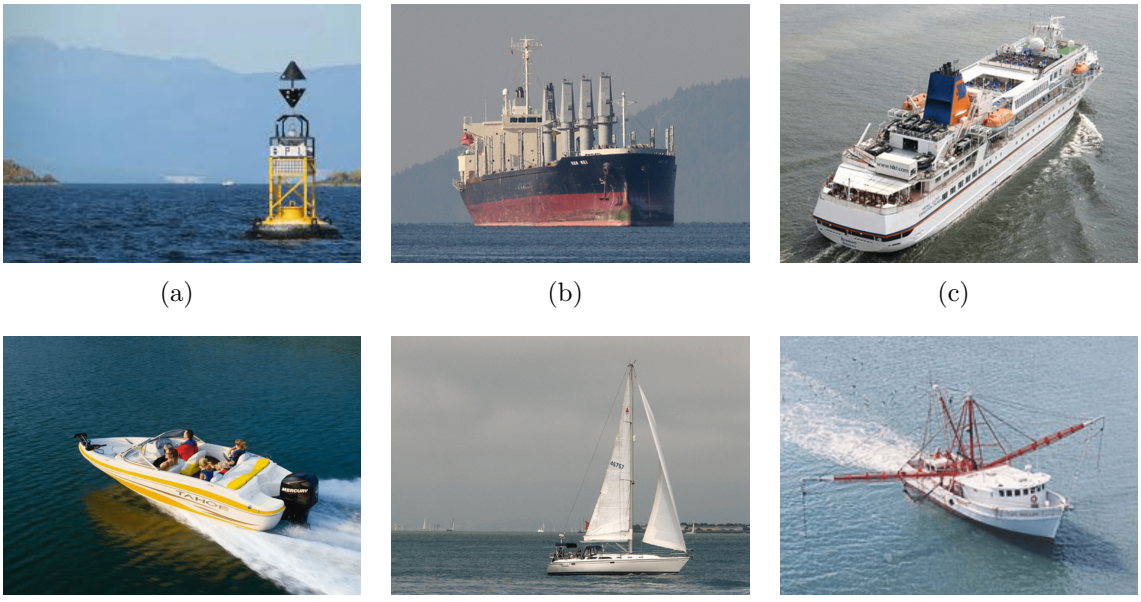
are: Computer Vision System Toolbox, Neural Network Toolbox, Parallel Computing Toolbox and Statistics and Machine Learning Toolbox.

2.3 Digital Image Datasets

The data required for the implementation of object detection consists of digital images. Digital image datasets are collections of images that share certain traits and are normally presented in the same format. Along this thesis, different image sets have been used for different purposes. This section presents them, stating their source, color and file format and application within the scope of the thesis.

2.3.1 ImgaNet and other Internet sources

ImageNet is a publicly avialable large image database composed of digital images that have been collected and categorized in order to provide object detection projects, such as this one, with enough data to perform training and testing. It is the largest image set available, and contains over 14 million URLs of images and over 20 thousand categories. It can be accessed at: www.image-net.org/index.



(a) (b) (c)

(d) (e) (f)

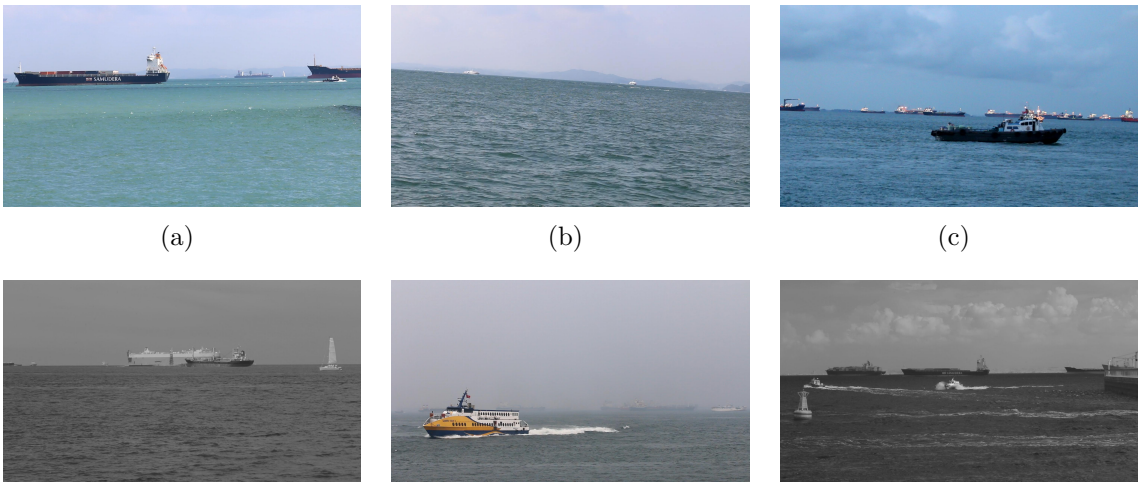
Source: www.image-net.org/index

Figure 2.3: Sample images from ImageNet ((b) to (f)) and other Internet sources (a)

The images used in the thesis coming from ImageNet are 6002 images from different vessel categories, grouped by the student in 10 categories, with the number of images per category ranging from 177 to 941. Since there is no category for buoys in ImageNet, but it is necessary to have buoy images within the project (buoys are sea markers that is vital to be able to detect), the student gathered 94 buoy images from various sources on the Internet. Overall, the 6096 image set gathered by the student is composed of 11 categories containing visible color images in JPEG format. This image set was used for training the different detectors and classifiers used in the project (section 4.4).

2.3.2 Singapore Maritime Dataset

The Singapore Maritime Dataset is a publicly available database containing videos taken at Singapore waters. There are both on board and on shore videos. The videos can also be categorized as visible and near infrared. Each video is composed of a sequence of frames taken consecutively at the same scenario, while the scenario changes from one video to another. The videos are acquired in high definition, and stored in AVI format. It is possible to dowload the videos at: sites.google.com/site/dilipprasad/home/singapore-maritime-dataset.



(a) (b) (c)

(d) (e) (f)

Source: sites.google.com/site/dilipprasad/home/singapore-maritime-dataset

Figure 2.4: Sample images from the Singapore Maritime Dataset: ((a) to (c) visible, (d) to (f) near infrared)

The student combined some of the visible videos into one Singapore Maritime Dataset visible video, and used one infrared video for the Singapore Maritime Dataset NIR video. These videos were used for evaluating the performance of the different detection methods developed during the project.

2.3.3 Helsingør Images

Some of the images taken by the DTU team using the equipment described above have been gathered by the student into the Helsingør image dataset. The name stems from the fact that the images were taken at one of the ferries that connects Helsingør, Denmark with Helsingborg, Sweden. In particular, the student combined various image sequences from two color cameras into the Helsingør color image set and various sequences from one monochrome camera into the Helsingør monochrome image set. The format of the images is TIF. The images were used to perform the detection final experiment.

(a) (b) (c)

Figure 2.5: Sample images from Helsingor dataset ((a) and (b) color, (c) monochrome)

2.3.4 Hundested Images

The student combined some images taken by Professor Mogens Blanke with cameras that are not part of the equipment of the project into a single image dataset. The dataset was named Hundested since most of the images were taken in waters close to Hundested. The images are visible color images, in JPEG format. Since the equipment used for obtaining the images does not belong the the one officially used at the project, this image set was used for evaluation.

Figure 2.6: Sample images from Hundested dataset

2.4 Summary

This chapter has introduced both the physical and the digital components relevant for the development of the project developed during the thesis. The physical equipment is formed by various sensors. The software used to implement object detection is MatLab, and a number of digital image datasets were employed at different stages of the development of the object detection system.

CHAPTER 3

Object Detection and Tracking System

The goal of this chapter is to offer some insight into the broader project of developing an USV, such that the topic of the thesis, that is, object detection from digital images, is properly put into context. The focus of this chapter is in the detection and tracking system that encompasses the object detection module. Sections within this chapter gradually move from the broader project to the more specific one through some intermediate stages that correspond to the detection and tracking scheme. The modular arrangement of the parts composing every level of the project allows for that distribution of the chapter. The content of the chapter is now outlined:

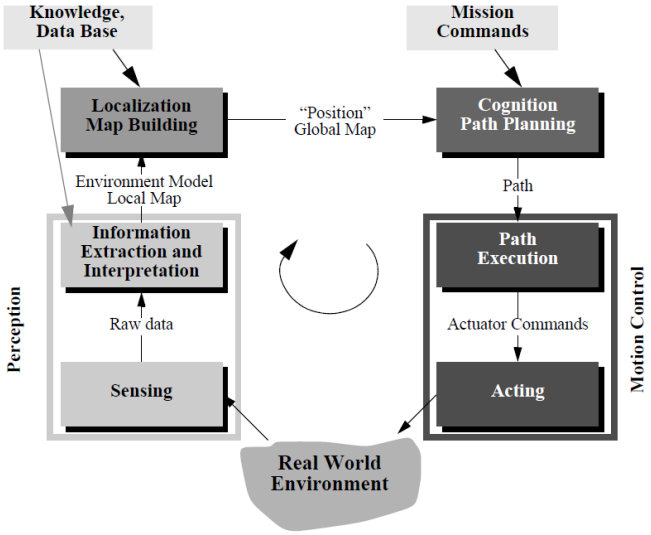
- Section 3.1 gives an overview of the general USV system.
- Section 3.2 covers the object detection and tracking system that makes up the perception module.
- Section 3.3 describes the sensor fusion algorithm within the object detection and tracking scheme.

3.1 Unmanned Surface Vehicle

The project documented in this report is part of a DTU larger project for designing and implementing an USV, that is currently being developed by the Automation and Control group within the Electrical Engineering department. Danish maritime authorities, as well as actors from the maritime industry, collaborate in the project. A brief description of that broader project is given in this section to contextualize the object detection and tracking system, and the object detection system within it.

An USV consists of a vessel that is equipped with a series of sensors, actuators and electronic devices for data processing and computation, arranged in such manner that the vessel performs a correct navigation, complying with maritime rules and avoiding collisions, without the need of human supervision. The framework used in this chapter

is inspired by the one introduced in [Sca11]:



Source: [Sca11]

Figure 3.1: General structure for an autonomous mobile robot

A crucial, and very useful, feature of the scheme shown in figure 3.1, is its modularity: the system is composed of self-contained compartments responsible each of them for specific tasks. This approach is very common in robotics and other technological fields, since it allows to break down a complex problem into smaller problems that are feasible to tackle individually, and it also gives then whole structure a large degree of flexibility.

That loop control scheme for a mobile robot can be directly applied to an USV, since, at that level of abstraction, the fundamental structure of the system is the same for a small autonomous robot and for an autonomous vessel. In the context of this thesis, the structure has been simplified into the following modules: perception, guidance, controller and the vessel itself:

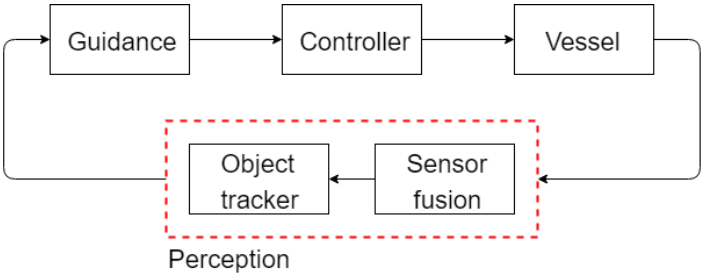


Figure 3.2: General structure for the USV system

The vessel is a watercraft, that is, a water-bourne vehicle. In an USV, the vessel constitutes the controlled object. A series of sensors are placed on the vessel, some for measuring the location, orientation and speed of the vessel itself (proprioceptive sensors, such as IMU and exteroceptive, such as GPS devices) and some sensing its surroundings (exteroceptive sensors, such as cameras and radar). Data collected by those sensors works as input for the perception module, that is responsible for extracting and combining information from that data in order to build a representation of the environment of the vessel.

That representation, that is basically composed of position and velocity of the vessel and of a series of tracked objects, is used by the guidance system, together with the destination reference of the vessel and the hydrographical and topographical characterization of its vicinity (provided by ENCs) to determine the trajectory and velocity the vessel should follow. The determined trajectory, together with the series of velocities along it, if properly computed by the guidance (or path planning) module, allows for the vessel to navigate efficiently and safely, following the shortest route without colliding with any other vessel or object at sea. The trajectory and velocity of the vessel have to be constantly updated based on the information provided by the perception module. The output of the guidance system is next employed to determine the values for the control signals that would actuate the vessel in the desired way. That task is performed by the controller.

The next section narrows down its scope from the general project in figure 3.2 to just the perception module (red dotted square). That module is partly composed of an object detection and tracking system, that, from data gathered by extereoceptive sensors, obtains information regarding the relative position and velocity of objects in the vicinity of the vessel, that is later used by the guidance module.

3.2 Object Detection and Tracking

The perception module in figure 3.2 includes the localization of the vessel and the tracking of objects in its vicinity. The vessel localization is composed of its position, orientation and velocity, and is determined by processing signals from an IMU+GPS device. This section leaves vessel localization aside, and focuses only on the detection and tracking of objects in the surroundings of the vessel. The detection and tracking system determines the relative position and relative velocity of objects on the sea surface with respect to the vessels, as well as the category of those objects (among a finite range of classes that includes buoys, cargo ships, sailboats, etc.). In order to do so, it would be very convenient to employ AIS signals. Automatic Identification System (AIS) is an automatic tracking system used for vessel traffic purposes. The reason such powerful source of information has been dismissed in this thesis is that most small vessels, such as fishing boats, small sailboats and alike, don't use it, so relying on AIS information would

result in the non detection of many vessels. Instead, digital cameras and a radar device are used. The scheme of the object detection and tracking system, shown in figure 3.3, is inspired by [Bla15].

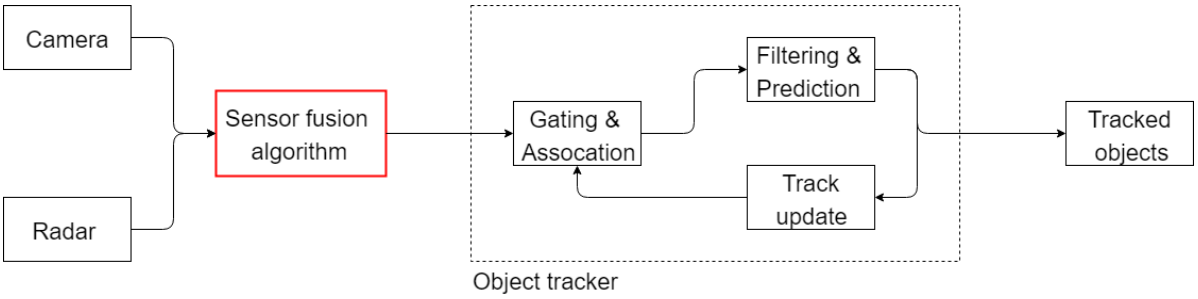


Figure 3.3: Scheme for the object detection and tracking system

The detection and tracking system takes digital images and radar scans as inputs. A sensor fusion module extracts and correlates information from those inputs, and outputs information of detected objects, including their location relative to the vessel and their categories. That information constitutes the input of the object tracker module, that uses it to either update already tracked objects position and velocity or to start new tracks if the detected objects don't correspond to any previously tracked object. The output of the object detection and tracking system is a series of tracked objects, including their positions and velocities relative to the vessel, and their classes.

The sensor fusion algorithm (red square in figure 3.3) is treated in more detail in section 3.3, so just a brief description of its functionality is given here. The sensor fusion algorithm first detects objects on digital images, thus classifying them. Then, it searches for the detected objects on the corresponding radar angles and determines their location in the vessel coordinates.

The object tracker receives the classified and located objects as inputs and, in the gating and association module, compares their classes and location with those of already tracked objects. If there is a correspondence both in category and position between a detected and a tracked object, they are matched. Otherwise, a new track is initialized with the information coming form the sensor fusion module.

The next step in the object tracker, that takes place at the filtering and prediction module, consists on updating position and velocity values for tracked objects by applying the Kalman filter measurement update step, where the information from the sensor fusion algorithm is used as measurements. Then, the prediction update for position and velocity takes place, using a constant velocity model.

Then, the predicted objects are subject, in the track update module, to a process by which certain tracks are confirmed (if they were recently initialized), some others are maintained and others are deleted. This way, new tracks are incorporated to the series of tracks sent to the guidance module, and old tracks of which there are no new measurements are discarded.

Having presented the fundamental functionality of every of the modules that compose the object tracker, now for each of them there is a more detailed treatment. The development of the following subsections is based on [Sca11] and [Bla15].

Let's first adopt a Cartesian coordinate system (O, X, Y, Z) that has its origin (O) at the intersection between the sea surface (considering it a plane) and a line perpendicular to it that passes through the center of gravity of the vessel. The X axis belongs to the sea surface plane, passes through the origin and is parallel to its stern-bow axis, with positive values corresponding to the ahead direction. The Y axis belongs to the sea surface plane, passes through the origin and is parallel to the port-starboard axis, with positive values pointing towards the port side. The Z axis is perpendicular to the sea surface plane, passes through the origin and positive values are assigned such that the right hand rule applies.

Tracked objects are characterized in the vessel coordinate frame by their position, velocity and a signature (the signature is an integer value that is equivalent to a certain object category) :

$$X_{obj} = \begin{bmatrix} x_{obj} \\ y_{obj} \\ v_{x_{obj}} \\ v_{y_{obj}} \\ s_{obj} \end{bmatrix} \tag{3.1}$$

Measurements outputted by the sensor fusion module consist of position and signature:

$$Z_{obj} = \begin{bmatrix} x_{obj} \\ y_{obj} \\ s_{obj} \end{bmatrix} \tag{3.2}$$

Objects are modelled assuming a constant velocity. The motion model does not include the signature. The expression for the motion model, in continuous time, is:

$$\dot{X}_{obj}(t) = \begin{bmatrix} \dot{x}_{obj}(t) \\ \dot{y}_{obj}(t) \\ \dot{v}_{x_{obj}}(t) \\ \dot{v}_{y_{obj}(t)} \end{bmatrix} = \begin{bmatrix} v_{x_{obj}}(t) \\ v_{y_{obj}}(t) \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_{obj}(t) \\ y_{obj}(t) \\ v_{x_{obj}}(t) \\ v_{y_{obj}}(t) \end{bmatrix} = A_c \, X_{obj}(t) \tag{3.3}$$

, where A_c denotes the system matrix in continuous form.

The expression that correlates object state and measurement (again, not considering the signature), in continuous time, is:

$$Z_{obj}(t) = \begin{bmatrix} x_{obj}(t) \\ y_{obj}(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_{obj}(t) \\ y_{obj}(t) \\ v_{x_{obj}}(t) \\ v_{y_{obj}}(t) \end{bmatrix} = C_c \, X_{obj}(t) \tag{3.4}$$

,where C_c corresponds to the measurement matrix in continuous form.

The linear continuous state space model is, therefore:

$$\begin{cases} \dot{X}_{obj}(t) = A_c \, X_{obj}(t) \\ Z_{obj}(t) = C_c \, X_{obj}(t) \end{cases} \tag{3.5}$$

The nature of the computations performed at the object tracking module and the fact that measurements are available only at certain moments makes it necessary to use a discrete model. The continuous model at 3.5 is thus discretize to:

$$\begin{cases} X_{k+1} = A \, X_k \\ Z_k = C \, X_k \end{cases} \tag{3.6}$$

,where a sample time T has been chosen to perform the discretization, X_{k+1} represents the object state at instant $T(k+1)$, X_k represents the object state at instant Tk, Z_k represents the measurement at instant Tk.

A corresponds to the discrete system matrix, obtained by discretizing A_c, and C corresponds to the discrete measurement matrix (equal to C_c):

$$A = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{3.7}$$

So the discrete model in 3.6 can be written as:

$$X_{k+1} = \begin{bmatrix} x_{k+1} \\ y_{k+1} \\ v_{x_{k+1}} \\ v_{y_{k+1}} \end{bmatrix} = \begin{bmatrix} x_k + Tv_{x_k} \\ y_k + Tv_{y_k} \\ v_{x_k} \\ v_{y_k} \end{bmatrix}, \quad z_k = \begin{bmatrix} x_k \\ y_k \end{bmatrix} \tag{3.8}$$

Now the concept of uncertainty, that is the key idea behind the application of algorithms such as the Kalman filter, is introduced. The continuous and discrete models presented in equations 3.1 to 3.8 are deterministic, that is, there is an exact relationship between input and output values, expressed by the system and measurement matrices. However, in a real scenario, it is not realistic to assume that a model completely captures the relationship between variables. That is, there is always an error in the model. If it is assumed that no bias error has been made while developing the model, there is a random model error. In the same fashion, values from measured variables also differ from the real ones, due not only to quantization errors (that, with enough memory space, can be considered negligible) but also, and specially, to measurement noise, that is, a random disturbance in the value of the magnitude measured introduced by the measurement equipment. Again, it is assumed that no bias is introduced while performing the measurement. Both model and measurement errors produce uncertainty, since they don't allow to determine the exact value of a given variable.

The object tracking module is designed based on the existence of modelling errors and measurement noise, in such a way that they compensate each other to a certain extent. That is achieved my applying the Kalman filter algorithm (at the filtering and prediction stage, subsection 3.2.2). Since it is not possible to know with absolute certainty the value of a variable (such as the position or velocity of a vessel), the tracking module relies on predicted and estimated values. Predicted values are obtained by applying the state space model to previous values, and are denoted by a caret (such as $\hat{x}$). Estimated values incorporate observed values to predicted ones (estimated or observed variable example: x).

It is assumed that both model errors and measurement noise can be considered white noise. To express the uncertainty related to the value of a given variable, that variable is characterized by the statistical properties of mean and variance. The mean is employed as the value of the variable, that together with the variance forms a pair that offers information about the expected value of the variable and the possible deviation from it.

For the estimated object, mean vector and covariance matrix at a given time $t = Tk$ are X_k and P_k, whereas for the predicted object they are $\hat{X}_t$ and $\hat{P}_k$. For observed measurements (those obtained by measurement devices), mean vector and covariance matrix are Z_k and R_k, whereas for predicted measurements (obtained by applying the

lower equation in 3.6), mean vector is $\hat{Z}_k$. Variance values are determined experimentally.

3.2.1 Gating & Association

The aim of the gating and association module is to assign observed measurements to tracked objects. In order to do so, the first step is to compare one by one each observed object to each tracked one. Signatures are compared first. For all the observed-tracked pair of objects for which there is a signature match, positions are compared using the Mahalanobis distance. In order to do so, the difference between the observed and the predicted measurements, known as innovation, is computed:

$$V_k^{ij} = Z_k^i - \hat{Z_k^j} \tag{3.9}$$

, for every pair of objects i and j. The innovation covariance matrix is computed as:

$$\Sigma_k^{ij} = C\hat{P_k^j}C^T + R_k^i \tag{3.10}$$

At equation 3.10 it can be seen that the covariance for the innovation is computed as the addition of the covariance of the predicted measurement to the covariance of the observed one. Finally, the Mahalanobis distance is calculated and compared to a gating threshold:

$$(V_k^{ij})^T(\Sigma_k^{ij})^{-1}V_k^{ij} \leq c \tag{3.11}$$

The gating threshold value should be set experimentally. Once every observed measurement has been compared to every predicted one with which there is a signature match, a series of matched measurements have been obtained, with the possibility that more than one observation has been matched to a given prediction and viceversa. At this point, the final association can be easily performed, by assigning to every tracked object the closest observed measurement, obtaining a one-to-one match. If by the end of that association process there are unmatched observed measurements, they initialize new tracks.

3.2.2 Filtering & Prediction

The filtering and prediction module is used to estimate tracked objects position and velocity as accurately as possible, by combining predicted values to observed measurements that correspond to the same objects (that correspondence has been guaranteed at the gating and association module). This module uses the Kalman filter algorithm. First, the measurement update takes place, using the matched observed information from the sensor fusion algorithm as measurement, to obtain the estimated values for position and

velocity. Then, the prediction update is applied.

In order to perform the measurement update, the Kalman gain is computed for every matched object:

$$K_k = \hat{P}_k C_k^T (C\hat{P}_k C^T + R_k)^{-1} = \hat{P}_k C_k^T (\Sigma_k)^{-1} \tag{3.12}$$

The Kalman gain optimally determines to what degree the predicted position and velocity of a tracked object are corrected by its corresponding observed measurement. That depends on their relative uncertainties (expresed by their covariances). How the Kalman gain affects both the mean and the covariance of an estimated object can be understood by looking at the prediction update expressions:

$$\begin{cases} X_k = \hat{X}_k + K_k(Z_K - \hat{Z}_k) = \hat{X}_k + K_k V_k \\ P_k = \hat{P}_k - K_k(C\hat{P}_k C^T + R_k)K_k^T = \hat{P}_k - K_k(\Sigma_k)K_k^T \end{cases} \tag{3.13}$$

The upper equation at 3.13 expresses how the predicted position and velocity values are corrected by the observed measurements, where the correction term depends on the Kalman gain (larger values of the Kalman gain give larger corrections) and the innovation. The lower equation expresses the reduction in the predicted states uncertainty, that is also larger the larger the Kalman gain is. A very important implication of that interpretation of the lower equation, which is what makes the Kalman filter such a powerful tool, is the fact that by combining uncertain information from different sources (a model and some measurements) it is possible to reduce the overall uncertainty.

At this point of the object tracking algorithm, estimated states of confirmed tracks are send to the guidance module. The series of past and present tracked objects allows the guidance system to predict future trajectories of those tracked objects, and thus take the necessary measures (by applying a method such as Model Predictive Control (MPC).

After the measurement update has been performed, from the resulting estimations of all the tracked objects it is possible to predict the state values of mean and covariance at the next iteration $(k+1)$ by applying the model equation:

$$\begin{cases} \hat{X}_k = AX_k \\ \hat{P}_k = AP_kA^T \end{cases} \tag{3.14}$$

For tracks for which there has been no match at the gating and association step, measurement update is not applied, but prediction update is. This is equivalent to relying solely on the model.

3.2.3 Track Update

The next step within the object tracker algorithm is the track update. Every track, at this part of the algorithm, falls into one of the next four categories:

1. **Case 1**: Confirmed tracks for which measurement update has been applied at the current iteration.
2. **Case 2**: Confirmed tracks for which measurement update has **not** been applied at the current iteration.
3. **Case 3**: Recently initialized tracks (still not confirmed) for which measurement update has been applied at the current iteration.
4. **Case 4**: Recently initialized tracks for which measurement update has **not** been applied at the current iteration.

Tracks that fall under case 1 maintain there status. For tracks in case 2, it is checked for how many iterations in a row the measurement update has not been applied (that is equivalent to check for how long the tracked object has not been observed by sensors). When that number exceeds a certain threshold, the track is deleted. Tracks in case 3 are granted the confirmed status after a certain number of consecutive iterations where measurement update has been applied (which means that they are being consistently observed by sensors). Tracks in case 4 are immediately deleted.

This step constitutes a robust way of managing tracks, such that objects no longer observed by the sensors stop being tracked and newly observed objects are incorporated. Predicted measurements from all non deleted tracks are compared with observed measurements in the gating and association step of the next iteration.

3.3 Sensor Fusion

The purpose of the sensor fusion module is to extract and combine information from data gathered by different sensors, namely cameras and radar. That information is then sent to the tracking module, that uses it to update the objects being tracked. The relevant information to gather is the location and type of the objects in the vicinity of the vessel. The location is relevant since it directly informs about the distance of objects to the vessel, and since it allows to determine velocity (by means of the object tracking module) and from there it is possible to estimate future trajectories, which is crucial for collision avoidance. The type, or class, of objects also constitutes relevant information for navigation, since the way the vessel should react to a possible collision with, for example, a sailboat and a motorboat is not the same.

Cameras facilitate the task of identifying objects, since images offer very rich information that can be used to determine the category of an object. However, they are not very useful when it comes to measuring distance, and therefore cannot be used for determining the relative position of the objects detected with respects to the vessel. A radar, on the other hand, does not offer detailed features from which categorize an object, but it does offer range and beam, from where locating a given object in the vessel coordinates (vessel coordinate frame description in section 3.2) is straightforward. The approach chosen consists in combining information extracted from data gathered from both types of sensors, such that cameras classify objects and a radar locates them, and is based on [Nun06]. The algorithm for such information extraction and fusion is represented as a block diagram in figure 3.4.

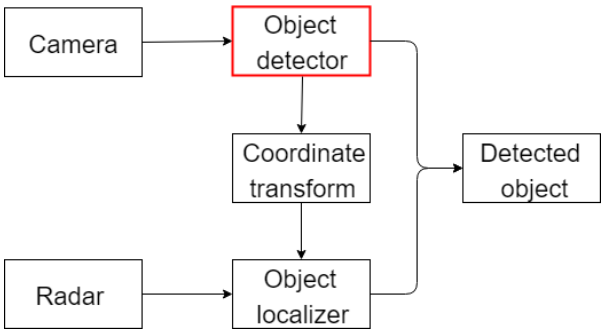


Figure 3.4: Scheme for the sensor fusion module

The sensor fusion algorithm, shown in figure 3.4, works as follows. First, objects are detected on images taken by digital cameras. Detection implies both localization of objects in image frame (pixel coordinates) and classification of objects. Next, detected objects are associated with the radar beam that corresponds to their respective image coordinates. The correspondence between image and radar coordinates depends on the relative orientation of the camera with respects to the radar, and is performed by means of a coordinate transformation. Once the beam has been determined, the range of the object can be obtained from radar data. Finally, radar coordinates are transformed to vessel coordinates and the position information added to the category information to compose the observed measurement to be sent to the object tracking module.

A more detailed description of every step within the sensor fusion module is now given, starting with the object detection step. The object detector module (red square in figure 3.4) is responsible for locating (in image frame) and categorizing objects in digital images. There are different methods that can be applied in order to achieve that. They are treated in detail in chapter 4. Object detection follows, regardless of the method used for its implementation, a certain structure, shown in the diagram of figure 3.5.

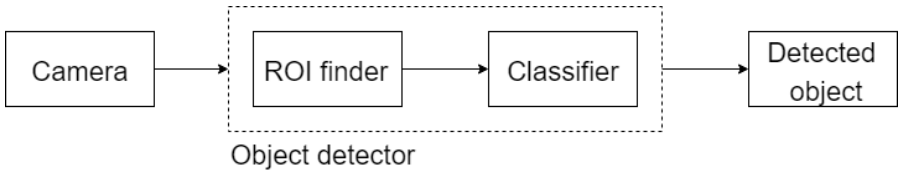


Figure 3.5: Scheme for the object detector module

The object detection module receives a digital image as input. It first locates Region of Interest (ROI), that are regions of the image where objects are located. Optical flow could be used to reduce the amount of false detections by comparing current ROIs with ROIs of images from previous iterations. ROIs have a rectangular shape, and their perimeter is known as bounding box. A bounding box is represented by the pixel coordinates (u, v) of the upper left corner and the height and width of the box. The next step consists of the classification of the objects contained inside bounding boxes into a finite set of categories. The classification process yields, for every ROI, a certain category and a score, that represents the probability that the object inside the bounding box belongs to the given category. The output of the object detector is a vector of objects, each of them composed of a bounding box, a score and a signature for the category (that is, an integer value that has a correspondence to a certain category):

$$Z_{det} = \begin{bmatrix} b\_box \\ score \\ signature \end{bmatrix} \tag{3.15}$$

The next step consists on selecting, for every detected object, the radar angle that corresponds to the localization of that object in pixel coordinates. For that purpose, the u coordinate (bare in mind the unit of the image coordinate system is the pixel) of the center of the bounding box is used. The origin of the image coordinate frame is the upper left corner of the image. Taking the orientation of the camera relative to the radar into account, there is an unambiguous relationship between a u pixel coordinate of the image and the angle of the radar.

Consider the pixel coordinates of the camera optical center (that corresponds to the center of the image) are represented by (u_0, v_0) in the image frame and the focal length of the camera by f. Consider also that ϕ is the horizontal angle between the optical axis and the line that passes through the center of projection and $(u, 0)$. Then, the relationship between the u coordinate of a point in the image and ϕ is:

$$\phi = atan(\frac{u - u_0}{k_p f}) \tag{3.16}$$

,where k_p is a constant that allows for the conversion of distances in meters into pixels.

Assume the camera and radar vertical axes are aligned, and there is a certain horizontal angle θ between the radar x axis and the camera optical axis. Then, the relationship between the u coordinate of a given detected object in the image frame and the radar angle for that object ψ is:

$$\psi = \phi + \theta = atan(\frac{u - u_0}{k_p f}) + \theta \tag{3.17}$$

Once the radar angle ψ has been determined, the range r for that angle is retrieved from the radar data. Then, the coordinates of the object in the vessel frame can be obtained. If it is assumed that the x axes of the radar is collinear with the vessel frame x axis, the object coordinates can be computed by just applying:

$$\begin{bmatrix} x_{obj} \\ y_{obj} \end{bmatrix} = \begin{bmatrix} r \cdot cos(\psi) \\ r \cdot sin(\psi) \end{bmatrix} \tag{3.18}$$

Finally, the observed measurement is formed by adding, for every detected object, the signature that represents the category of the object to its position in the vessel frame:

$$Z_{obj} = \begin{bmatrix} x_{obj} \\ y_{obj} \\ signature \end{bmatrix} \tag{3.19}$$

3.4 Summary

Chapter 3 offers a context for the detection methodologies developed along the thesis. The content of the chapter is structured from the broadest system, that is, the autonomous vessel scheme, to the more specific one, that is, the detection module itself. For every level, basic concepts and descriptions were outlined, although not into great detail, since this section is a theoretical exercise and there has been no implementation related to it.

First, the general framework of the USV system was treated. Within it, perception, guidance and controller modules were briefly described. Next, as part of the perception module, the detection and tracking system was covered. It is composed of the sensor fusion and the object tracker systems. The latter one was treated in more detail first, giving a mathematical description of its inner components (gating and association, filtering and prediction, and track update modules). Finally, the sensor fusion algorithm is covered in some detail, and the object detector is presented as part of it.

CHAPTER 4

Image Analysis for Object Detection

The most crucial information, from a semantic perspective, gathered for sensor fusion purposes within the scope of this project, is extracted from data collected by cameras, that is, from images. The specific process of information extraction that is relevant for the development of this chapter is known as object detection. Detection of a certain type of object on an image taken by a camera, either daylight or infrared, consists on determining whether a member of that given category is present on the image. Such process of object detection is applicable to images thanks to the rich information on color, texture and shape captured on them.

In order to allow for an adequate contextualization of object detection, it is necessary to introduce some basic definitions:

- **Computer vision:** Field of computer science that focuses on computer systems obtaining a visual understanding of the physical world.
- **Image analysis:** Extraction of meaningful information from images.
- **Machine learning:** Field of computer science that aims at computer systems to progressively acquire knowledge from data, without that knowledge being explicitly programmed.
- **Supervised learning:** Machine learning task where a function is shaped by inference from data in the form of input-output pairs.
- **Unsupervised learning:** Machine learning task where a function is inferred from just input data (no corresponding output data).

In this chapter, a basic theoretical background related to digital images and image processing is first given. A theoretical introduction to convolutional neural networks is also given. Then, different methods employed for object detection are treated. Finally, tests performed for evaluation of the performance of the detection methods are presented, as well as the results obtained:

- Section 4.1 covers the theoretical basis of digital images.
- Section 4.2 covers the theoretical basis of image processing.
- Section 4.3 covers the basic concepts and principles related to convolutional neural networks.
- Section 4.4 treats in detail the different detection methods applied in the thesis.
- Section 4.5 covers the evaluation of the performance of the previously treated detection methods.

4.1 Digital Images

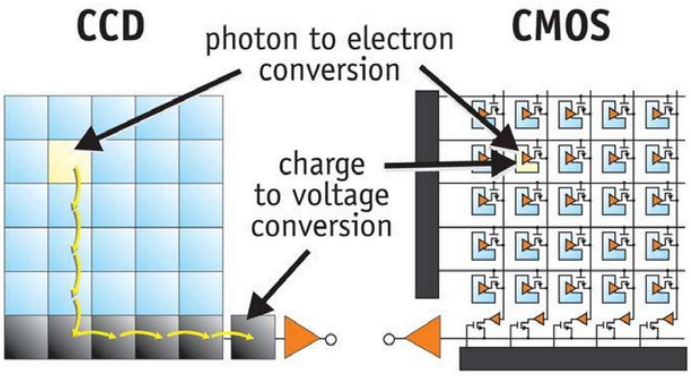
Images constitute the object of the detection methods presented in this chapter. More specifically, digital images, since the cameras used in the project are digital cameras and the aforementioned methods are therefore applied to this type of images. For a proper understanding of such methods, it is thus important to start by introducing digital images. The information exposed in this section has as its primary source [Sca11].

Images are bi-dimensional visual representations of three dimensional physical objects present in the world. A digital image, in particular, is an image with a finite resolution (resolution is a magnitude that characterizes the level of detail on a given image), and is formed on a chip composed of a matrix of light-sensitive picture elements, called pixels. The number of pixels of an image, that can range from the order of thousands to the order of millions, determines its resolution. Pixels can be thought of as capacitors that accumulate charge based on the amount of photons that struck them during a given period of time. In other words, digital images measure total light intensity.

The electro-optical device that is responsible for the formation of digital images is known as digital camera. The working principle of a digital camera is as follows: light, originated from either a natural source (the Sun) or an artificial one, is reflected in surfaces of objects in the world, and finally reaches the camera, where it passes through a number of lenses before reaching the imaging sensor, that is, the chip composed of pixels presented above.

Based on the technology used to collect the charge of every pixel, it is possible to distinguish between Charge Coupled Device (CCD) and Complementary Metal Oxide On Silicon (CMOS) digital cameras. In CCD cameras, the process of reading the charge of every pixel takes place at a corner of the chip, which implies that all the charges have to be transported from their pixels to that corner. It is therefore critical for CCD cameras that every single charge is preserved during transportation. In CMOS chips there is a specific circuitry attached to every pixel, such that the charge is measured

and amplified without the need to transport it. This difference in the working principle of CCD and CMOS chips results in the latter being considerably cheaper and much less power consuming. CMOS cameras are more widely use, and their low power consumption gives them a clear advantage for robotics applications. On the other hand, CCD cameras outperform CMOS in quality sensitive application.



Source: meroli.web.cern.ch/lecture_cmos_vs_ccd_pixel_sensor.html

Figure 4.1: CCD vs CMOS technologies

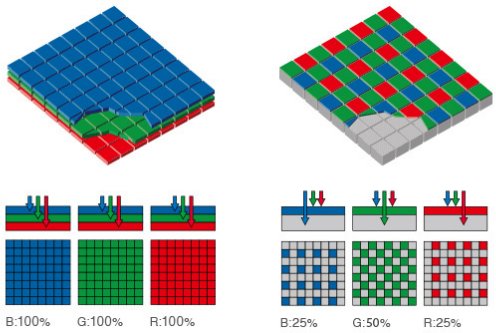
Regardless of the technology used to capture pixel charge, there are different types of digital cameras based on the type of images they are able to generate. Those images can, in turn, be stored in several digital formats. It is thus possible to categorize digital images according to the following criteria (among several other): images based on color and spectrum range; and images based on digital format.

4.1.1 Digital Images Based on Color and Spectrum Range

A **grey scale** image is a digital image where total light intensity within the visible light spectrum (that is, radiation within the following wavelength range: [390 700] nm) is measured, with no color distinction.

A **color** digital image, that corresponds to the type of images generated by the daylight cameras used in the project, differs from the grey scale image treated above in that, in color images, a specific color filter is applied to each pixel, such that it only measures the light intensity of that specific color (that can be red, green or blue). There are two configurations for color images. One of them consists on alternating pixels with different color filters on the same chip following a certain pattern by which there are twice as many green pixels as blue or red (to adapt digital images to the vision properties of the human eye). This configuration is known as the Bayer filter. The other configuration

consists on three separate chips, each of them of a single color (that is, one red, one green and one blue).



Source: www.ephotozine.com/article/digital-camera-image-sensor-technology-guide-16808

Figure 4.2: Three color chip (l) vs Bayer filter (r)

An **infrared** digital image works under the same principles of a grey scale digital image, but it is sensitive to near-infrared light (wavelength range: [750 1400] nm), instead of visible light. Near-infrared images have been used in different types of applications, such as night surveillance activities; in this project, the use of infrared cameras makes it possible to obtain visual information of the surroundings of the vessel at night. Near-infrared cameras use an infrared light source (called illuminator) in low light conditions in order to be able to obtain information of the surroundings without the human eye being affected by the light (since it is not sensible to infrared light).

4.1.2 Digital Images Based on Digital Format

There are many storage formats for digital images; the ones presented in this subsection correspond to those used during the experimental phase of the project.

Joint Photographic Experts Group (JPEG) is a lossy compression method for digital images. The term lossy refers to the fact that, in the compression process, some original information is lost and cannot be restored. This results in reduced image fidelity. The JPEG standard is based on the discrete cosine transform (DCT), by which spatial information is converted to frequency information. In the case of JPEG, the compression takes place by discarding high frequency information. This explains the smooth variations of tone and color typical of JPEG images. The JPEG format is among the most commonly saved in digital cameras, as well as one of the most common format for storing and transmitting images on the Internet.

Source: thisisnotaprint.blogspot.com

Figure 4.3: Near-infrared image

Portable Network Graphics (PNG) is a graphics file format that allows to store digital images with lossless compression. Such compression method is composed of two stages: the first one consists of a pre-compression (a filtering process that makes the image more efficiently compressible), using the second one a lossless compression algorithm known as "DEFLATE". The PNG format is widely used for transferring images on the Internet. PNG compression is preferred to JPEG for images with sharp contrast, such as those containing text, while JPEG allows for smaller file sizes for images with smoother transitions.

TIF, or Tagged Image File Format (TIFF) is a file format for storing digital images. TIF is a flexible format, and can be adapted to contain different types of compressed, both lossy and lossless, and uncompressed images. TIF format, as well as the other two presented, supports both grey scale and Red, Green, Blue (RGB) images.

4.2 Image Processing

The aim of this section is to introduce some image processing techniques that are relevant for the theoretical treatment of the detection methods employed in the thesis. The main source of information for this section is [Sca11].

Digital image processing can be define as a group of tools that can be applied to digital images in order to obtain modified images that possess enhanced qualities, such that image appearance is improved or the extraction of information is facilitated. Image processing, in the context of the project, is a fundamental step in the image analysis process, since it allows to filter out most of the data contained in digital images (data

that is irrelevant for the purpose of object detection) and to keep relevant data. Such data is further treated by a detection method, eventually obtaining useful information.

The amount of image processing techniques if vast, and it ranges from frequency filters to remove certain components of images, such as noise, to feature extraction used to identify objects. In the following subsections, only the relevant techniques for the application of the detection methods used in the thesis are covered.

4.2.1 Basic Image Processing Operations

One of the most basic image processing operations, that is widely used in many techniques, is filtering. Filtering consists on removing some of the original data of an image, and can be performed in the frequency or in the spatial domain. The latter type of filters is treated here.

Let's introduced some notation first. A given (grey scale, for simplicity) digital image I can be mathematically described by the intensity value (that ranges from 0 to 255) at every pixel. Pixels are represented as a pair of integer numbers, (x, y), that correspond to height and width, in pixel units, from the upper left corner of the image. Then, the intensity value of the image at a given pixel is represented as $I(x, y)$.

A spatial filter (also known as mask, or kernel) consists of a small region S_{xy} (normally a rectangle) around a given pixel (x, y) and an operation T that is applied to the pixels of that region:

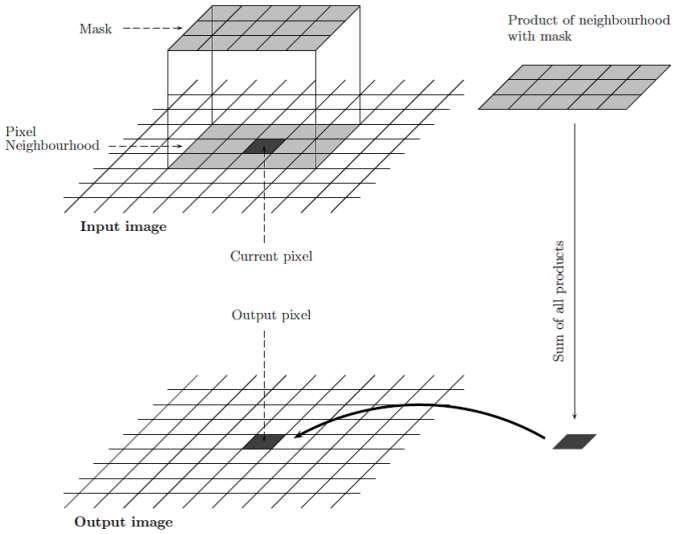
$$I'(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} \omega(s,t) I(x-s, y-t) \tag{4.1}$$

,where $I'(x, y)$ is the filtered image for the pixel, ω is the filter value corresponding to a specific pixel of the kernel spatial distribution and $I(x - s, y - t)$ is the original image for that pixel (that belongs to S_{xy}).Such operation is called convolution, and can be written in a more compact way:

$$I'(x,y) = \omega(x,y) * I(x,y) \tag{4.2}$$

,where $*$ is the convolution operator.

A filter is applied to an image by using the previous formula for every pixel of the image. A smoothing filter is used to blur images and to reduce noise. This noise reduction is of interest as a prior step when applying derivatives to the image intensity,



Source: dsp.stackexchange.com/questions/29490/what-is-an-arithmetic-mean-filter

Figure 4.4: Spatial filter

since derivatives are very sensitive to noise. A commonly used smoothing filter is the Gaussian filter, where the continuous operator is:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{4.3}$$

,where $G(x,y)$ is the value of the filter for a given pixel and σ^2 is the variance of the Gaussian distribution. The discrete operator G is obtained by sampling the continuous operator from its maxima vertically and horizontally in a uxv pixel matrix, with the maxima at the center, and then normalizing.

After the image has been smoothed with the Gaussian filter, the derivative operation can be applied. Thanks to a property of convolution, it is possible to combine a Gaussian filter and the derivative operation into a single kernel. For the simplest case of first order derivative:

$$(G * I)' = G' * I \tag{4.4}$$

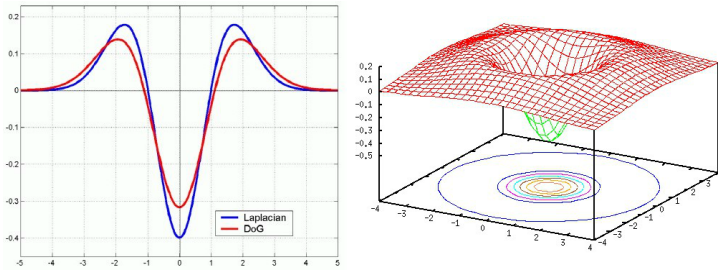
The Laplacian of Gaussian (LoG) is a more complex operator, but follows the same structure as shown in 4.4. It is regarded as a good operator when it comes to identify sharp changes in an image intensity. Its continuous expression is:

$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{4.5}$$

Sources: forums.ni.com/t5/Machine-Vision/gaussian-filter/td-p/2441104/page/3 www.csie.ntu.edu.tw/ r93944019/cv/hw10/DOG.jpg

Figure 4.5: Sequence of images: original (l), Gaussian (c), DoG (r)

A similar method to the LoG is the Difference of Gaussian (DoG), that yields almost identical results to those of LoG but reduces computation costs. DoG is achieved by applying a Gaussian to an image twice, each time with a different value in width (σ), and then taking the difference of the resulting images.



Sources: sensblogs.wordpress.com/2011/08/22/quick-reviews-on-descriptors-and-detectors-by-fei-fei-li/ fourier.eng.hmc.edu/e161/lectures/gradient/node9.html

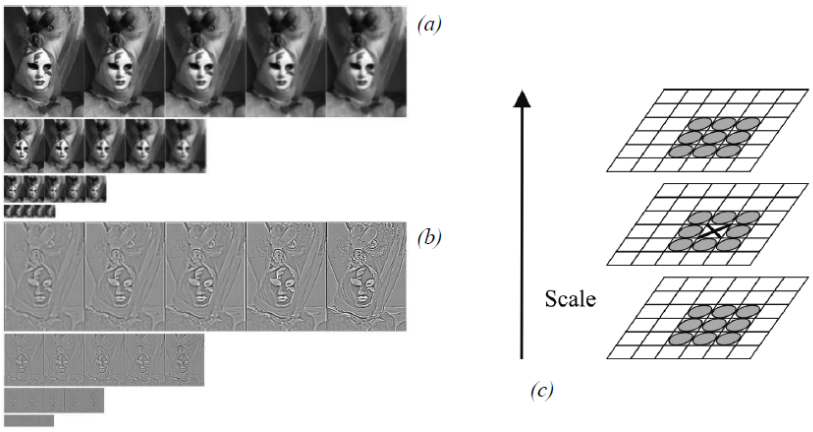
Figure 4.6: LoG vs DoG (l), bidimensional DoG (r)

4.2.2 Features

Within the context of image processing, features are patterns in images that differ from their surroundings in color, intensity and texture. There are different tools for identifying blobs on digital images. The ones covered in this subsection are relevant for the treatment of the detection methods used.

Probably one of the most robust feature extractors is **Scale Invariant Feature Transform (SIFT)**. The features extracted by SIFT are characterized by being very distinctive, and invariant to changes in illumination, rotation, viewpoint and scale. This has made this feature extractor widely use in image analysis applications.

The algorithm works as follows. First, DoG is applied to the image where features are to be detected. This is done by blurring the original image at various scales with a series of Gaussian filter with different σ and then computing the difference between successive images in the same scale. SIFT features are then identified as local maxima or minima of the DoG images across scales, and candidate keypoints in low contrast areas or along edges are discarded. Finally, a descriptor for every feature is formed. Such descriptor is composed of a normalized vector of orientation histograms corresponding to the neighboring region of the keypoint.



Source: [Sca11]

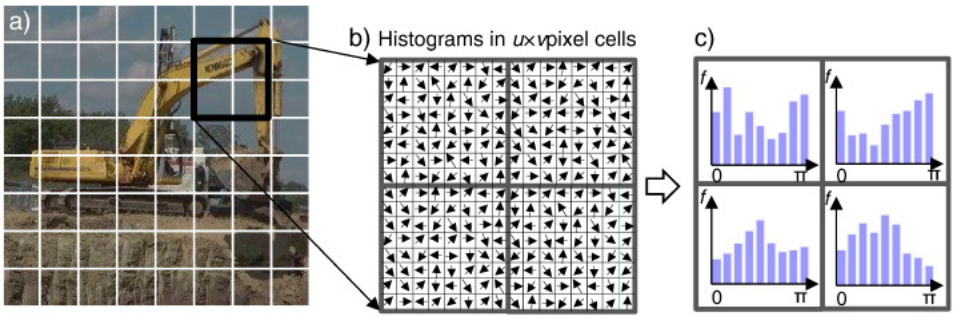
Figure 4.7: SIFT algorithm: a) Blurred images using Gaussian filters with different σ at various scales, b) DoG on those images, c) Detection of local maxima or minima

The **Speeded Up Robust Features (SURF)** detector is based on SIFT, but considerably more efficient, although is also less robust. There are two main differences with SIFT. One of them is the fact that DoG filters are approximated by Haar wavelets. Haar wavelets are square-shaped functions that together form a wave-like oscillation. The other difference lies in the way the sum of elements within the convolution operation is performed. In the case of the SURF algorithm, integral images are used. Integral images, also known as summed-area tables, consist on an algorithm for calculating the sum of values of a matrix in a fast and efficient way.

Another popular feature detector is **Maximally Stable Extremal Regions (MSER)**. A maximally stable extremal region is a connected set of pixels that has either higher or lower intensity than all the pixels on its outer boundaries. These regions constitute features, and are selected using intensity thresholding.

The last feature detector covered in this subsection is **Histogram of Oriented Gradients (HOG)**. This technique is based on computing local descriptors for small, connected, regularly distributed regions in a given image called cells. These descriptors

are local normalized distributions (histograms) of directions of gradients.



Source: www.sciencedirect.com/science/article/pii/S0926580512002403

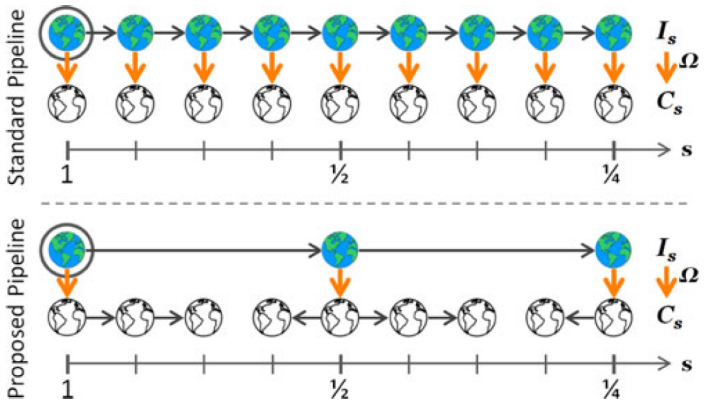
Figure 4.8: HOG algorithm: a) Digital image divided in cells, b) Oriented gradients per cell, c) Histograms of oriented gradients

4.2.3 Fast Feature Pyramids

As shown in the previous subsection, feature detectors such as SIFT or SURF require the computation of filters on images at different scales (a multi-scale representation of an image, such as the one used for computing SIFT features, is known as feature pyramid). Such multi-scale filtering process is computationally costly, and constitutes a bottleneck for many object classification and detection algorithms. That limitation prompted the development of an image processing tool known as fast feature pyramids. This tool is used in one of the object detection methods applied. The information presented in this subsection has been extracted from [Per14].

Fast feature pyramids is a technique that relies on the fact that it is possible to approximate a filtered image at a certain scale by extrapolation from a filtered image at a nearby scale. That way, it is not necessary to explicitly compute every scaled image and then apply the subsequent filter. Instead, just a small number of scales are chosen to perform the up or down scaling of the original image, and then the relevant filter is applied. Finally, the rest of the filtered images along the ”pyramid” of scales are obtained by prediction from the ones available.

The principle that makes such prediction operation accurate enough to successfully apply this tool is power law. Power law can be formulated as a relationship between two numerical entities (in this case, digital filtered images) that depends only on the ratio between those entities, not on their absolute value. That can be expressed mathematically, applied to filtered images, as:



Source: [Per14]

Figure 4.9: Fast feature pyramid working principle compared to standard feature pyramid computation

$$T(I_{s_1})/T(I_{s_2}) = (s_1/s_2)^{-\lambda_T} + \varepsilon \tag{4.6}$$

,where T is a kernel, s_1 and s_2 are two different scales, I_{s_1} and I_{s_2} correspond to the original image scaled to those values, λ_T is the ratio power for that kernel (to be found empirically) and ε is a certain deviation from the power law (of relative small magnitude).

Here, the way in which power law can be applied to the formation of feature pyramids is presented by comparison to the conventional way. The standard approach for obtaining a scaled filtered image consists on first rescaling the image and then applying the kernel operator:

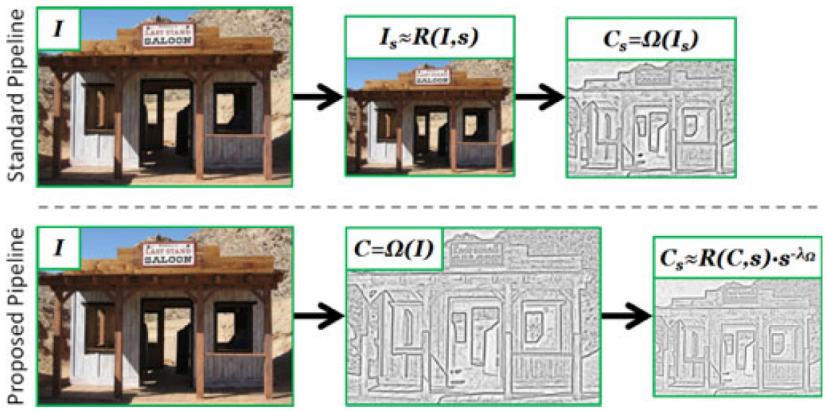
$$I'_s = T(R(I, s)) \tag{4.7}$$

,where I'_s is the filtered image at scale s and R is a rescaling operator.

By applying power law , it is possible to apply first the kernel operator and then scale the image to the desired size:

$$I'_s \approx R(I', s)s^{-\lambda_T} \tag{4.8}$$

When using fast feature pyramids, the filter operation is performed only at certain scales of the pyramid, as opposed to the standard method, where the filtering operator is applied at every scale. Then, at intermediate scales, expresion 4.8 is used to find the scaled filtered images from the filtered image at the nearest scale.



Source: [Per14]

Figure 4.10: Power law applied to filtered images compared to the standard way of scaling and filtering images

4.3 Convolutional Neural Networks

This section is aimed at introducing Convolutional Neural Network (CNN) before treating its application to object classification and object detection within the scope of the methods employed in the thesis. CNN can be broadly described as a machine learning technique for multiclass image classification, using neural networks. In this section, neural networks are first treated in general, and then CNN in particular. The primary source of information for this section devoted to CNN has been a Master thesis recently developed at NTNU ([Tan17])

4.3.1 Introduction to Neural Networks

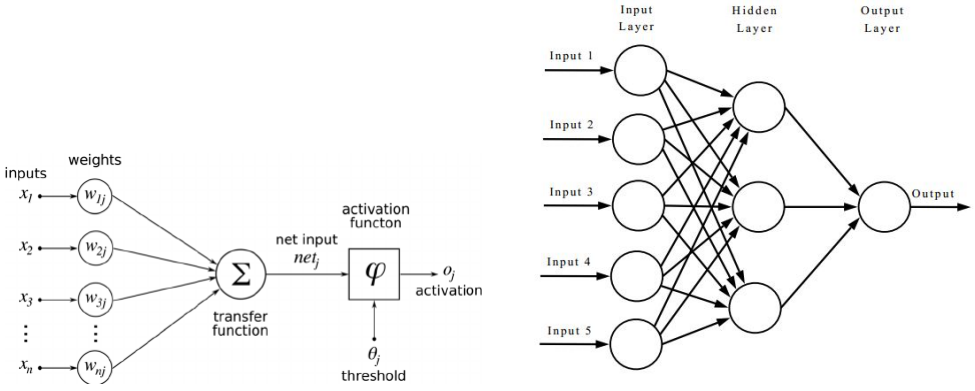
Neural Network (NN), also known as Artificial Neural Network (ANN) are computing systems composed of interconnected nodes (artificial neurons) organized in consecutive layers. Each neuron from a given layer receives a particular combination of weighted outputs from the nodes of the previous layer as inputs. The, it sums them and adds a bias parameter. Next, it applies a non-linear function to the sum, called activation function, without which the operation inside the neuron would be linear. Finally, it outputs the result to the neurons of the next layer. That series of operations can be expressed, for a single neuron, as:

$$a_j^l = \sigma(z_j^l) = \sigma\left(\sum_k \left(w_{jk}^l a_k^{l-1}\right) + b_j^l\right) \tag{4.9}$$

,where a_j^l is the output of the neuron j from layer l, σ is the activation function of the

neuron, z_j^l is the sum of inputs, w_{jk}^l is the weight of input from neuron k of the previous layer to neuron j, a_k^{l-1} is the output of neuron k and b^l is a bias introduced by the neuron.

As stated above, neurons are organized in layers. All the neurons belonging to the same layer receive their inputs from the same set of neurons (in different combinations based on the particular input weights of every neuron) and send their outputs to another set of neurons, that correspond to the previous and next layer, respectively. The first layer of a neural network is known as input layer, the last as output layer, and all the intermediate layers are known as hidden layers. This nomenclature suggests that a neural network can be thought of in practice as a black box where for a given input there is a corresponding output and the internal operations are not visible to the user.



(a) (b)

Sources: (a)www.kdnuggets.com/2016/10/artificial-intelligence-deep-learning-neural-networks-explained.html, (b)tex.stackexchange.com/questions/132444/diagram-of-an-artificial-neural-network

Figure 4.11: ANN structure at different levels: (a) Internal structure of an artificial neuron; (b) Structure of a simple neural network

By a process of supervised learning, it is possible to train a NN for a specific task. In the training, weights applied to neuron outputs and bias parameters are tuned through an iterative process known as gradient descent to obtain the optimal values for the specific task, that in turn uses backpropagation for determining the values of gradients. The training is supervised because the objects composing the training set are made up of inputs to the NN and their corresponding outputs.

The training algorithm is based on the minimization of a cost function, that typically consist of the mean square error between the desired output of the neural network given a specific input and the actual output for that same input:

$$C = \frac{1}{2N} \sum_{x} \| \, y(x) - a^L(x) \, \|^2 \tag{4.10}$$

,where C is to cost function to be minimized, N is the number of objects that make up the training set, x represents the input of one of the objects, $y(x)$ is the correct output for that input and $a^L(x)$ is the output obtained by applying the NN to be trained with the current parameter values.

An important assumption for NN training is that the total cost function is the average of the cost functions for every object of the training set:

$$C = \frac{1}{N} \sum_{x} C_x \tag{4.11}$$

,which means that, for a specific object of the training set, the cost function is:

$$C_x = \frac{1}{2} \| \, y - a^L \, \|^2 = \frac{1}{2} \sum_{j} \left(y_j - a_j^L \right)^2 \tag{4.12}$$

,where j represents every individual output element.

In order to minimize C, weights and biases are tuned until a local minima is reached in the cost function. The method used for obtaining the parameter values for that minima is known as gradient descent, and is based on iteratively adjusting weights and biases in the decreasing direction of the cost function. It is possible to express the cost function increment (for a training sample) based on those of the parameters as follows:

$$\Delta C \approx \frac{\partial C}{\partial w^l} \Delta w^l + \frac{\partial C}{\partial b^l} \Delta b^l \tag{4.13}$$

,where ΔC, Δw^l and Δb^l are the cost, weight and bias increments, respectively, and $\left(\frac{\partial C}{\partial w^l}, \frac{\partial C}{\partial b^l} \right)^T$ form the gradient of C, also denoted as ∇C. If $(\Delta w^l, \Delta b^l)$ are grouped into Δv, it is possible to obtain a short notation for expression 4.13:

$$\Delta C \approx \nabla C \Delta v \tag{4.14}$$

Δv can be manipulated, and it is chosen to guarantee that the cost function decrements its value (that is, the increment of the cost function is always negative) by setting it to:

$$\Delta v = -\eta \nabla C \tag{4.15}$$

,where η is a small, positive parameter called learning rate, that defines how large are the decrements in δv. By applying 4.15 to 4.14, the decrement in C can be expressed as:

$$\Delta C \approx -\eta \parallel \nabla C \parallel^2 \tag{4.16}$$

,from which it is obvious that ΔC is always negative.

Gradient descent consists on updating the values of the weights and bias for every neuron in the NN iteratively, such that the overall cost function C is minimized:

$$v \rightarrow v' = v - \eta \nabla C \tag{4.17}$$

Finally, separating by parameter:

$$w^l \rightarrow w^{l\prime} = w^l - \eta \frac{\partial C}{\partial w^l} \tag{4.18a}$$

$$b^l \rightarrow b^{l\prime} = b^l - \eta \frac{\partial C}{\partial b^l} \tag{4.18b}$$

From equations 4.18 it is clear that the calculation of gradient descend depends on the prior knowledge of the values of ∇C. This values are obtained, for every object of the training set and every iteration of gradient descent, by means of the backpropagation technique.

Backpropagation is based on propagating perturbations on the activation functions of neurons from all the layers to the output layer of the neural network. The process starts at that last layer and moves backwards to every hidden layer and ultimately to the input layer. At every layer, partial derivatives of weights and biases at every neuron are calculated based on how the perturbations from those specific neurons affect the cost function.

Let's first define the perturbation. For a given neuron, the activation function goes from $\sigma(z_j^l)$ to $\sigma(z_j^l + \Delta z_j^l)$. The perturbation propagates until it reaches the output layer, and it produces a change in the cost function of $\frac{\partial C_x}{\partial z_j^l} \Delta z_j^l$. The propagation error for that neuron is then defined as:

$$\delta_j^{x,l} = \frac{\partial C_x}{\partial z_j^l} \tag{4.19}$$

That error can be computed for the output layer (characterized as L) with the following equation:

$$\delta^{x,L} = \nabla_a C_x \; o \; \sigma'(z^L) \tag{4.20}$$

,where $\nabla_a C_x$ corresponds to the gradient of the cost function with respects to the output of the NN and $\sigma'(z^L)$ expresses the derivative of the activation functions of the last layer with respects to the perturbation. Those are values that can be computed. It is possible to calculate the propagation error of any given layer taking the error of the next layer as starting point:

$$\delta^{x,l} = ((w^{l+1})^T \delta^{x,l+1}) \; o \; \sigma'(z^l) \tag{4.21}$$

,where w^{l+1} are the weights of the inputs for the next layer. This equation allows to understand why this algorithm is called backpropagation. The last step of the algorithm consists on relating the values of those errors to the partial derivatives of the cost function with respects to weights and biases, which is straight forward:

$$\frac{\partial C_x}{\partial w^l} = \delta^{x,l}(a^{l-1})^T \tag{4.22a}$$

$$\frac{\partial C_x}{\partial b^l} = \delta^{x,l} \tag{4.22b}$$

Now that the expressions for computing the values of ∇C for every sample of the training set have been given, it is possible to incorporate them into the gradient descent computation (equation 4.18):

$$w^l \rightarrow w^{l\prime} = w^l - \eta \sum_x \delta^{x,l}(a^{l-1})^T \tag{4.23a}$$

$$b^l \rightarrow b^{l\prime} = b^l - \eta \sum_x \delta^{x,l} \tag{4.23b}$$

At this point, the most important aspects regarding the structure and training of neural networks have been covered. The remainder of this subsection deals with a particular type of NN.

4.3.2 Introduction to CNN

A neural network tailored for object classification from image inputs is known as convolutional neural network (CNN). CNN is currently considered as a state-of-the-art technique in the field of computer vision, and it outperforms other methods, both when applied just to classification or embedded in a detector. CNN building requires supervised training with several images from relevant classes.

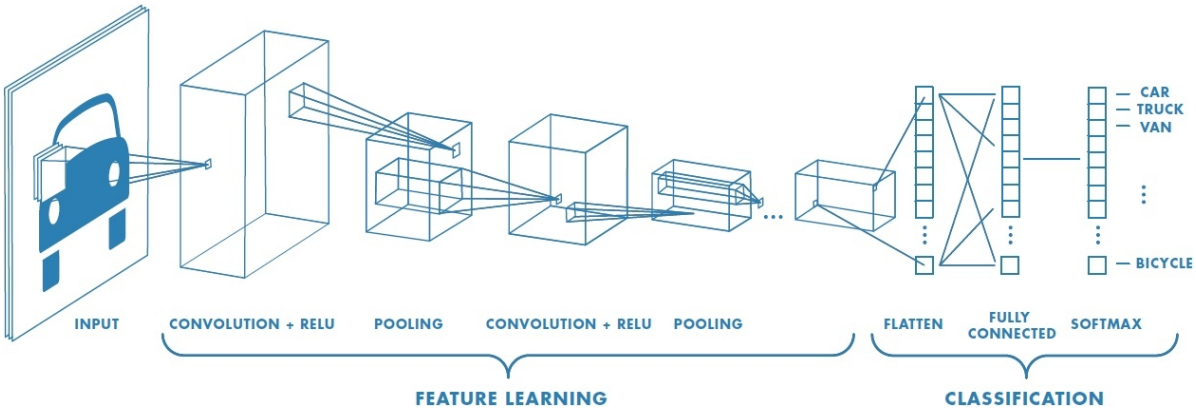
CNN smallest building blocks are neurons, that are organized in layers, as in any type of neural network. CNN are characterized by being composed of four types of layers: convolutional layers, Rectified Linear Unit (ReLU), pooling layers and fully connected layers. Each of these types has its own particularities:

- **Convolutional layer:** Considered as the core of a CNN, this type of layer consists of a number of filters applied to small regions of a given image (in an operation called convolution, as seen in section 4.2) and generating a series of output images known as activation maps. These activation maps constitute the input of the next layer. A group of convolutional layers work in an equivalent, although not the same, way to feature extractors. A peculiarity of convolutional layers is the existence of hyperparameters related to any such layer. These are parameters not subject to optimization, and among them there is the number of filters (denoted as K) and their spatial extent (F).
- **Rectified linear unit:** A ReLU is typically used after a convolutional layer in a CNN. The activation function of a ReLU is used to threshold the output of a convolutional layer, making sure it doesn't take negative values: $\sigma(z^l) = max(0, z^l)$. ReLu are specifically optimized to allow for efficient computation, scale invariance and outputs sparce activation, which makes them suitable for combination with convolutional layers.
- **Pooling layer:** A downsampling layer with the aim of reducing number of parameters and therefore computations through the selection of certain activation maps from the previous convolutional layer. The downsampling process can be performed by means of different types of techniques, such as max-pooling, that takes the largest activation value from every neighbourhood of activation maps.
- **Fully connected layer:** A layer where all neurons are connected to all the activations from the previous layer. It has the disadvantage of being very computationally expensive. A type of fully connected layer, typically placed as the final layer at most CNN structures, is the softmax layer. For every image input to the CNN, this layer returns a set of scores that add up to 1, one for every category in the specific classification problem. This implies the normalization of all activations from the previous layer through an activation called normalized softmax function.

Different CNN models have been develop within the computer vision community. None of them is optimal for every possible application; the convenience of using one model or another depends on the specific problem. Although each model has its own distinctive architecture, the general structure, regarding the distribution of the different types of layers described above, follows some basic principles:

- The bulk of a CNN is composed of alternating convolutional layers and ReLU.

- Pooling layers are inserted at certain points along the CNN structure, reducing the amount of activations.
- The final layer corresponds to a fully connected layer with a normalized softmax activation function.



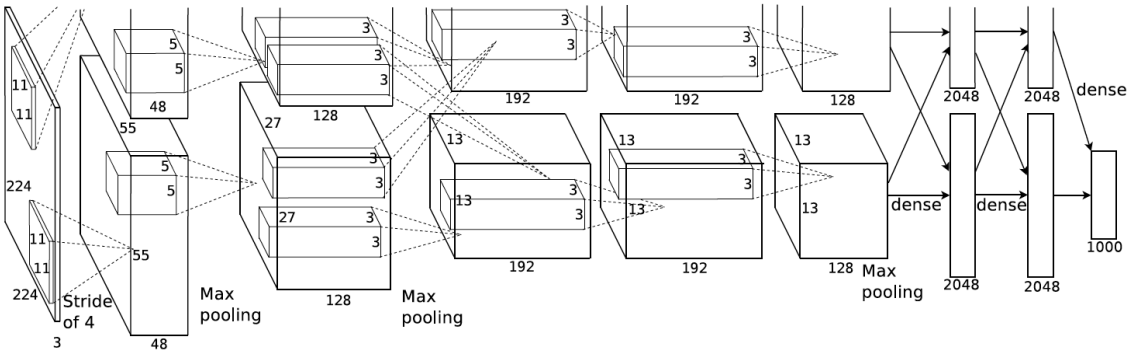
Source: hackernoon.com/what-is-a-capsnet-or-capsule-network-2bfbe48769cc

Figure 4.12: CNN structure

Two different CNN models are now presented. One of the is AlexNet, a model with a relatively simple structure that nonetheless has a good performance. The other is VGG16 -Visual Geometry Group (VGG)-, that possesses a more complex architecture. This two models are applied to the CNN-based detection methods employed in this project, and have been chosen in order to assess how the level of complexity of the CNN structure affects detection results.

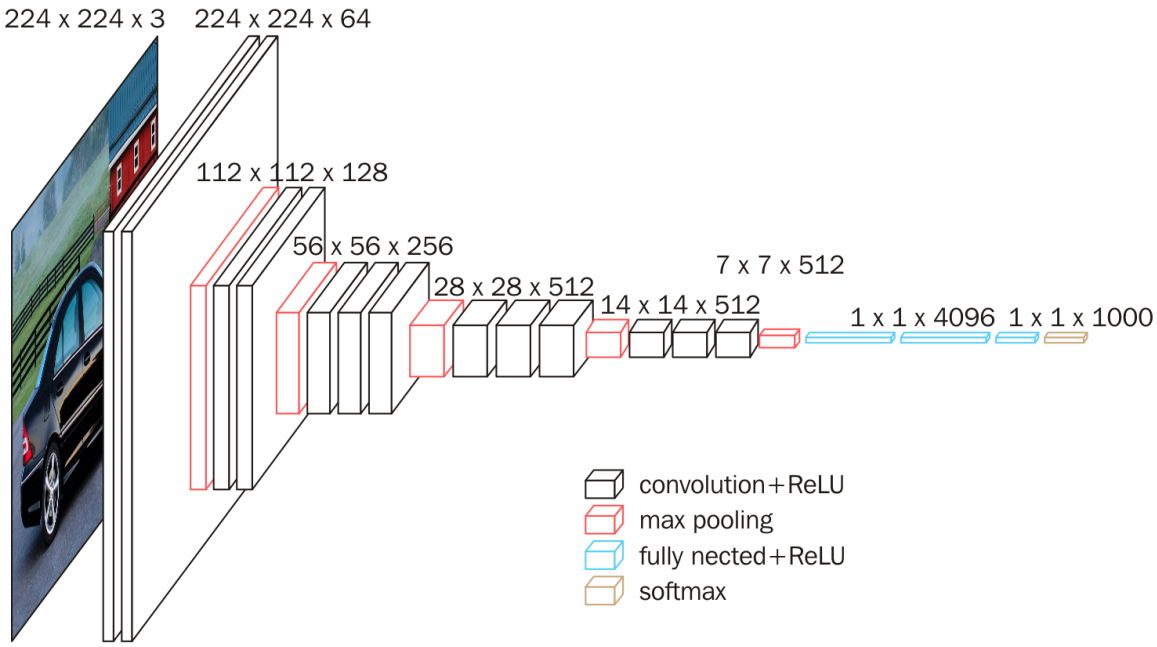
AlexNet is a CNN model that won the 2012 edition of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) ([Das17]), an annual sotfware contest based on detection and classification of images. It is composed of 5 consecutive convolutional layers, located at the beginning of the network, and 3 consecutive fully connected layers, at the end of it (being the last one softmax). ReLu is applied after every convolutional and fully connected layer. There are 3 pooling layers at different points along the convolutional layer part of the structure, that use max-pooling. A characteristic of this model is that all the layers except for the fully connected ones are split into two pipelines.

VGG 16 is a very deep CNN model, that ended up in second place at the 2014 edition of ILSVRC ([Das17]). It presents a very uniform architecture, with 13 convolutional layers (with their respective ReLu layers) alternating with 5 pooling layers (max-pool), followed by three fully connected layers and a final softmax layer. In total, the VGG16 model contains 138 million parameters.



Source: medium.com/@smallfishbigsea/a-walk-through-of-alexnet-6cbd137a5637

Figure 4.13: AlexNet archtitecture



Source: blog.datawow.io/cnn-models-ef356bc11032

Figure 4.14: VGG16 archtitecture

4.4 Detection Methods

Once the theoretical basis regarding image processing and convolutional neural networks have been covered, it is possible to treat the detection methods based on those image analysis techniques. That is the goal of this section. There are different approaches to object detection. In this project, both state-of-the-art technology (involving the appli-

cation of CNN) and more conventional approaches have been used. This allows for a comparison in detection performance (precision, time of detection, etc.) and the choice of the most suitable detector.

The detection methods employed can be grouped, regardless of whether they employ neural networks or classical techniques, in two categories: on one hand, pure detectors; on the other, a combination of a region of interest (ROI) finder and classifiers. Table 4.1 presents those methods classified by the techniques used and the categories.

		Technique	
		Conventional	CNN
Method	ROI & Classifier	ROI algorithm & SVM classifier	ROI algorithm & CNN classifier
	Pure Detector	ACF detector	Faster RCNN detector

Table 4.1: Methods for object detection

List of acronyms from table 4.1:

ROI	Regions Of Interest	areas of an image whose color and/or texture differs from that of their background
SVM	support vector machine	supervised learning model used for classification of data
ACF	Aggregate Channel Features	supervised learning model that extracts features from different channels of images for detection purposes
CNN	Convolutional Neural Network	specific type of neural networks used for object classification
RCNN	Region-based Convolutional Neural Network	CNN applied to object detection

Detectors from the upper row in table 4.1 are structured in two steps: first, regions of interest are detected on the image. These regions contain potential objects to be categorized. Then, each region of interest is classified within a given set of object categories. This second step has been applied by two different types of classifiers: SVM and CNN classifiers.

Detectors from the lower row in table 4.1 are categorized as pure detectors since they directly detect (that is, locate and classify) objects from raw data (images). In the case

of pure detectors, the two steps in which the detectors from the upper row are divided take place internally. Two pure detectors have been employed: a conventional one (that uses aggregate channel features) and the state-of-the-are Faster RCNN detector.

4.4.1 Classifier-based Detectors

Classifier-based detectors, composed of a ROI finder and a classifier, are treated in this subsection. Within the context of this thesis, such detectors are referred to as classifier-based detectors due to the fact that the detection is performed explicitly in two steps, the last one of them being classification.

The application of a classifier-based detector requires the prior training of the classifier to be used. The training is a process by which the classifier is specifically shaped to classify images within a certain set of categories that are relevant for the overall detection procedure. The particular way in which the training takes place depends on the technique used for classification, but is a process that always takes place offline, that is, before the detector is applied to a given scenario. Classification techniques used in this thesis (SVM and CNN) feature a type of training can be labeled as supervised learning, since it uses labeled images (being the input an image and the corresponding output its label).

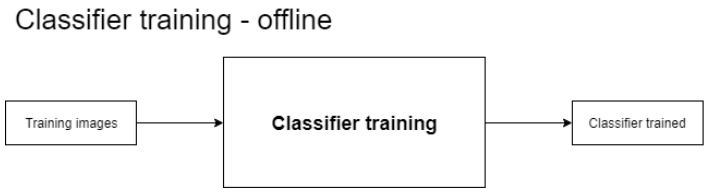


Figure 4.15: Classifier training diagram

Once the classifier has been trained, it is possible to apply it to the regions determined by the ROI finder, and thus integrate it into the detection process. Since this two-step process is employed to actually locate certain types of objects in images, which constitutes the object of detection, it is said to happen online.

The ROI finder algorithm is treated next, followed by the two classification techniques: SVM classification and CNN classification.

ROI Finder

The ROI finder algorithm has been completely developed by Juan Molla, a fellow student at DTU. The reason there is a subsection devoted to it is its relevance, since it is

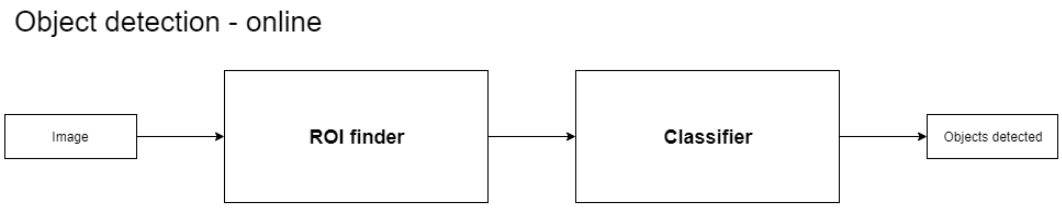


Figure 4.16: Classifier-based detection diagram

part of the classifier-based detectors applied in this thesis. As the development of this algorithm was not carried out by this student, its technical aspects are not treated in detail. Instead, important implementation aspects are discussed.

The sole purpose of the ROI finder is to localize regions on digital images that correspond to objects. This task is possible because objects usually have clearly distinguishable color and texture from those of its background, particularly when that background is made up of sea and sky, such as in most images in this project. Two different types of ROI finder have been developed: one is specially tailored for localizing buoys, the other aims at localizing vessels.

The algorithm for both types of ROI finder is the same, although the tools applied to some of its steps differ. The ROI finder algorithm consists of a number of consecutive steps:

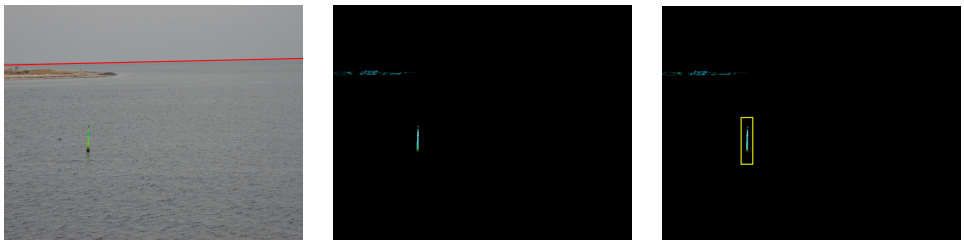
1. Horizon line detection
2. Background subtraction
3. Foreground segmentation

The first step consists on finding the horizon line. In open sea, the horizon line corresponds to the line that separates sea from sky. The technique used for horizon line detection, which is the same for both types of ROI finders, is known as orientation projection. This first step is useful in the context of maritime navigation for two reasons. One of them, that applies to the buoy ROI finder, is that many objects that should be detected appear normally under the horizon line on images taken on board the vessel, namely buoys. Thus, for the buoy ROI finder, once the horizon line has been detected the upper part of the image is removed. Notice that the usefulness of this step greatly decreases when it comes to detecting vessels, that usually appear partly above the horizon line. For that type of ROI finder, the usefulness of detecting the horizon line stems from the fact that the sea and the sky, being both part of the background, differ considerably in color and texture. Therefore, it is necessary to distinguish them in order to perform their respective background subtraction in the next step. It

is also worth mentioning that the horizon line detection becomes considerably more difficult when there is land on the picture (such as when leaving or approaching a harbour).

The next step is the background subtraction. At this point, it is necessary to differentiate the way this step is performed for buoys and for vessels. In the case of the buoy ROI finder, background subtraction consists on removing the sea and leaving only the objects on the foreground. This is achieved by means of a mask based on the Hue Saturation Value (HSV) color space. Since sea conditions vary greatly, there is no single mask that can be universally applied, and it is necessary to try different masks, each for certain sea conditions, and finally apply the one that best removes the background. This is done manually by the user, which implies that, for buoy detection, the process is not completely automatic, which constitutes a great limitation. For vessel localization, background subtraction is performed for sea and sky separately. The technique used is Gaussian Mixture Model (GMM), that can be adapted to recognize the color and texture patterns of sea or sky in a specific image. This, unlike the technique applied for background subtraction for buoy localization, allows for the automation of this step for vessel localization.

In the next (and final) step, the objects left are grouped into larger units and bounding boxes are drawn around them, in a process known as foreground segmentation, that is based on morphological operations. This technique is the same for both types of ROI finders. The main difference in this step lies in the fact that the bounding boxes for the buoy ROI finder have certain dimension restrictions. The regions of images demarcated by bounding boxes are known as regions of interest, and work as inputs to a classifier that will categorize them. That classification constitutes the final step of the detection process.



(a) (b) (c)

Figure 4.17: ROI algorithm for buoy localization: (a) Horizon line detection; (b) Background subtraction; (c) Region of Interest

SVM Classifier

Support Vector Machine (SVM) classification is a method for categorizing images based on feature spaces where categories are separated by hyperplanes, being those features arranged as vectorized histograms. Documentation for this section includes [Bra04] and [Sca11].

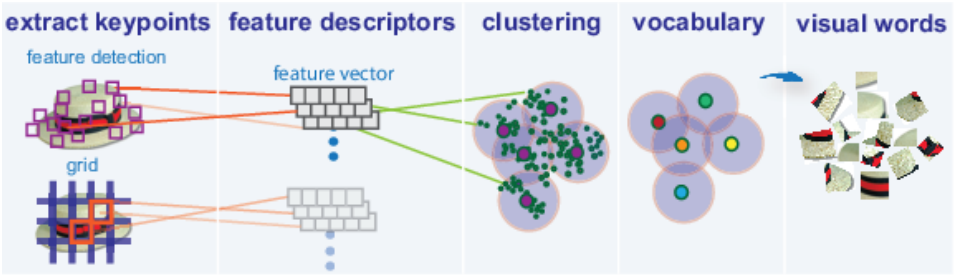
This method was implemented in MatLab by using various m-files and built-in functions provided by MathWorks as part of the Computer Vision System Toolbox. By applying those functions, that are based on [Bra04], it is possible to generate an SVM classifier as a multiclass image classifier, that assigns a given image to a certain category within a finite range of categories. The process for building and applying such type of image classifier is composed of a number of steps, that can be broadly grouped into: building a vocabulary, training the classifier and applying the classifier to images.

The first step, performed off-line before actually training the classifier is building a visual vocabulary from the training images, using the MatLab function *bagOfFeatures*, that receives a training set composed of images of categorized in different classes as input and outputs a visual vocabulary. A visual vocabulary is a collection of visual words, that is, mutually exclusive groups, each of them composed of similar features extracted from images. Those features are SURF, that present a good balance between robustness and speed. The function also allows to use MSER features, but tests carried out by the student yielded better classification results when SURF features were used.

Once features of a certain type have been extracted from a number of images, those features are grouped by means of a clustering process that after several iterations delivers compact and distinctive groups of similar features. Each of these groups corresponds to a visual word. The clustering process used is a simple partitioning method: k-means clustering, that iteratively assigns features to their closest cluster centers and recalculates those centers. Eventually, after enough iterations, a stable partition of features among the different clusters in achieved, and the visual vocabulary is formed.

The next step is the classifier training, for which the function *trainImageCategoryClassifier* is used.The training of an SVM classifier can be considered as supervised learning, and takes the training image set and its associated visual vocabulary as inputs, producing a multiclass classifier specifically tailored to categorize images of the classes used for the training. Every image used for training is encoded into a feature histogram, where every feature extracted from the image increases the count for the word assigned to that feature (that corresponds to its closest cluster center). That procedure is employed to obtain feature histograms (also known as feature vector) of all the images in the training set.

(a)



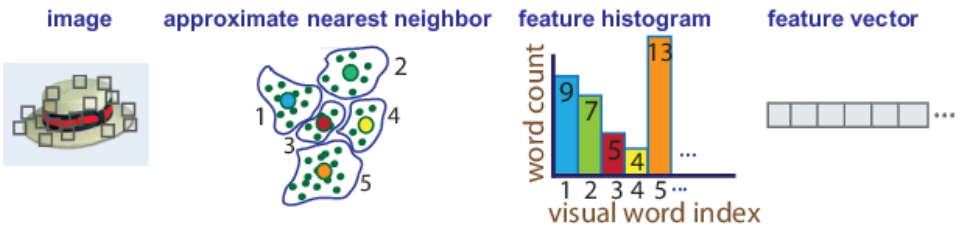
(b)

Sources: (a)cs.stackexchange.com/questions/73773/why-are-sift-features-significantly-different-after-simply-re-sizing-image, (b)se.mathworks.com/help/vision/ug/image-classification-with-bag-of-visual-words.html

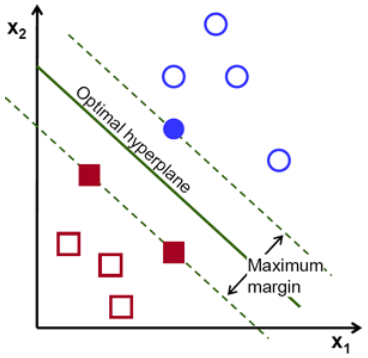
Figure 4.18: (a) SURF features; (b) Bag of visual words

Then the application of the SVM training algorithm to the set of histograms allows to distinguish classes by generating optimal hiperplanes. For every category, a distinct SVM training is applied. It creates a hyperplane in the visual world space (that is, the multidiminsional space in which every visual word corresponds to a dimension) that separates the given category from the rest (one-against-all approach) with maximal margin (where the margin is the distance between the hyperplane and the closest histogram vector point). The histogram vectors closest to the hyperplane are known as support vectors, hence the name support vector machine. The result of the training is the set of hyperplanes that separate object categories from one another, output by the training function as a *imageCategoryClassifier* object.

Once the SVM classifier has been trained with a sufficiently large and representative set of images from relevant categories, it can be applied to new images and it assigns them to one of those categories. This is implemented by calling the *predict* method for the classifier object. The method takes the *imageCategoryClassifier* object and the image to be classified as inputs and outputs the class the image belongs to. In order to achieve that, first, the features of the image are extracted and the feature histogram of



(a)



(b)

Sources: (a)se.mathworks.com/help/vision/ug/image-classification-with-bag-of-visual-words.html, (b)docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html

Figure 4.19: SVM classifier training: (a) Feature vector encoding from training images; (b) Classes delimitation with optimal hyperplanes

the image is formed. Then, the image is assigned to the category to which its histogram belongs (based on the division drawn by planes in the histogram space).

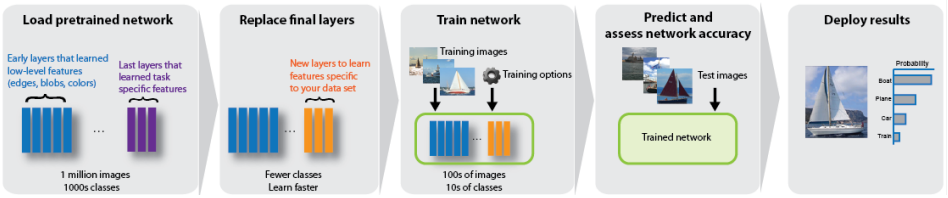
CNN Classifier

The use of CNN (Convolutional Neural Network, section 4.3) for multiclass classification is straightforward, since that is precisely its function. A CNN is first trained to classify images among a defined set of categories, and then applied to new images to perform the classification task on them.

The implementation of CNN classification in MatLab requires the use of m-files and built-in functions provided by MathWorks via the Neural Network Toolbox and the Statistics and Machine Learning Toolbox. Pretrained CNN models provided by MathWorks have been used; in particular, AlexNet and VGG16. This section covers the implementation of CNN training (including a relevant learning tool called transfer learning) and classification applying a trained CNN.

Training a CNN consists on determining the optimal values of the model parameters (weights and biases). In MatLab, this is achieved by calling the function *trainNetwork*, where the inputs are the training image set and the network model, and the output is the trained CNN. It is a supervised learning method, since instances for every class are provided through labeled images. For every image belonging to the training set, the model parameters are adjusted to increase the probability of the model performing a correct classification. This process takes place for every image of every class. The more images used in training, the more robust the classifier. The increased probability of accurate classification is reflected on the cost function, that is minimized by gradient descent using backpropagation (subsection 4.3.1).

A particular method within machine learning very relevant for training purposes is transfer learning, where an already trained neural network works as the starting point for the development of a new model, involving classes different to those of the original training. The feature extraction layers from the pretrained network are not retrained; only the last layers, those specifically trained for the original set of classes, are retrained (from zero, not from the values of the pretrained network) in order to be adapted to the new set of classes. This allows for a much faster training process, and is performed in MatLab by just substituting the last layers of the pretrained network by new layers (that is achieved by selecting all the layers from the previous network except the last three, and combining them with three new layers of types *fullyconnectedLayer*, *softmaxLayer* and *classificationLayer*, in that order), and then passing that network to the training function *trainNetwork*. Note that every time the network is trained, these last three layers are trained from scratch, not from their values of the previous training.



Source: se.mathworks.com/help/nnet/examples/transfer-learning-using-alexnet.html

Figure 4.20: Transfer learning scheme

Once a CNN has been trained, it can be used to classify new images. It is possible to do that in MatLab by just using the CNN object method *classify*, where the inputs are the trained CNN object and the image to be classified. For every input image, this method outputs an array of scores relative to the classes, that represent the probability that the image belongs to them. The image is assigned to the class with the highest score.

4.4.2 Pure Detectors

Pure detector are treated in this subsection, where the detection takes place, for implementation purposes, in one step. As was the case for classifier-based detectors, a training process needs to happen before applying the detector. For both types of detectors used in the project (ACF and Faster RCNN), the training is supervised learning, where a training set composed of labeled images is used to tune certain detector parameters in order for the detector to accurately perform the task for the classes of objects labeled in the training set.

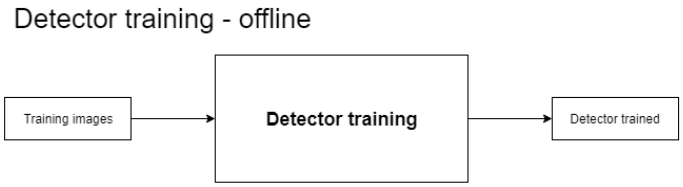


Figure 4.21: Detector training diagram

The term pure detector is used in the context of the thesis to emphasize the fact that detectors regarded as such perform the localization and classification of regions of interest internally, as opposed to classifier-based detectors. Therefore, they can be viewed as black boxes where the input is a given image and the output is a series of labeled and localized objects present in the image.

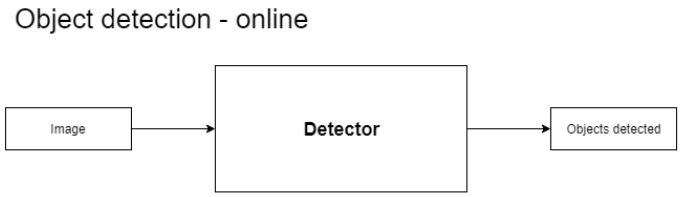


Figure 4.22: Pure detection diagram

ACF detector is covered first. Finally, Faster RCNN detector is treated.

ACF Detector

Aggregate Channel Features (ACF) detection is a method for object detection based on the computation of various filtered images from a given digital image in order to extract features. Those filtered images are known as channels, and they are combined to extract the features used for detection, hence the name, aggregate channel features.

The implementation of this detection method in MatLab made use of certain MathWorks provided m-files and built-in functions. These functions apply the methodology treated in [Per14], so they use fast feature pyramids (covered in subsection 4.2.3) for an efficient computation of features. The implementation of the ACF detector takes place, as stated at the introduction to the subsection, in two stages: training of the detector and application to images. Training is covered first.

The training is performed by applying the m-function *trainACFObjectDetector* to a training set composed of annotated images (obtained using Matlab's Image Labeler App). Whe using that function, the ACF detector is a single class detector that is trained, from positive and negative instances of the given class, through supervised learning to identify objects of that class on images. For that reason, one detector for every class has been trained. The training stage (that takes place internally when applying the training function) can be divided into two steps: first, the channel features computation; second, the learning process.

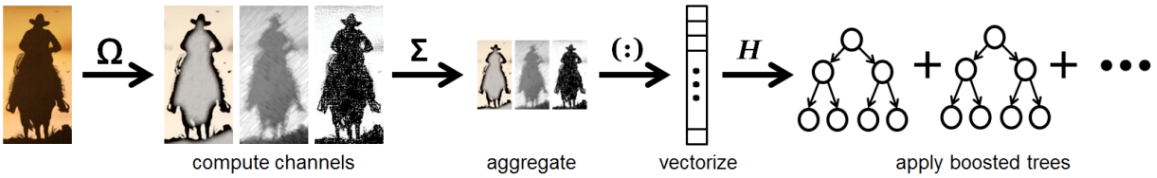
In the channel feature computation, from every training image, a number of channels are computed at different scales. Specifically, normalized gradient magnitude, histogram of oriented gradients (HOG, covered at subsection 4.2.2) and LUV color channels (LUV refers to the coordinate axes of the specific color space). Then, pixel blocks are summed and smoothed for every channel. Features correspond to single pixels lookups in those aggregated channels, where fast feature pyramids is used for efficiency.

The learning process is considered as supervised and uses the Adaptative Boosting (AdaBoost) algorithm. Information about AdaBoost is taken from [Bro16]. Such algorithm creates a strong classifier of a given class by combining weak classifiers. Weak classifiers are simple decision trees that, from a given input, yield a binary output, that is, the input belongs to a class or not. They perform by themselves just above random chance. When several weak learners are combined and their weights iteratively chosen based on their accuracy on the training set (more accurate models (decision trees) contribute more to the final prediction), a strong learner can be achieved. This results in an overall high accuracy, and is the core principle of AdaBoost.

In the particular case of the ACF classifier, positive and negative instances from the images composing the training set are fed into the weak learners and the boosting algorithm finds the best weights combination through the learning process. The output of the training function is an ACF detector object for the training set class.

Once the ACF detector has been trained for detection of a given class of objects, it is possible to apply it to new images. A MatLab built-in function is used for this purpose. That function is a method of the ACF detector object, that is applied by calling *detect*

and passing the detector object and the image it should be applied to as inputs. For every image, the detector will find features by computing the aforementioned channels and aggregate them. Then, those features are passed to the trained decision trees, that output detected objects of the given class. The output of the function is a vector of scores and bounding boxes coordinates for the objects found.



Source: jhui.github.io/2017/01/15/OCR-with-deep-learning/

Figure 4.23: ACF detector algorithm

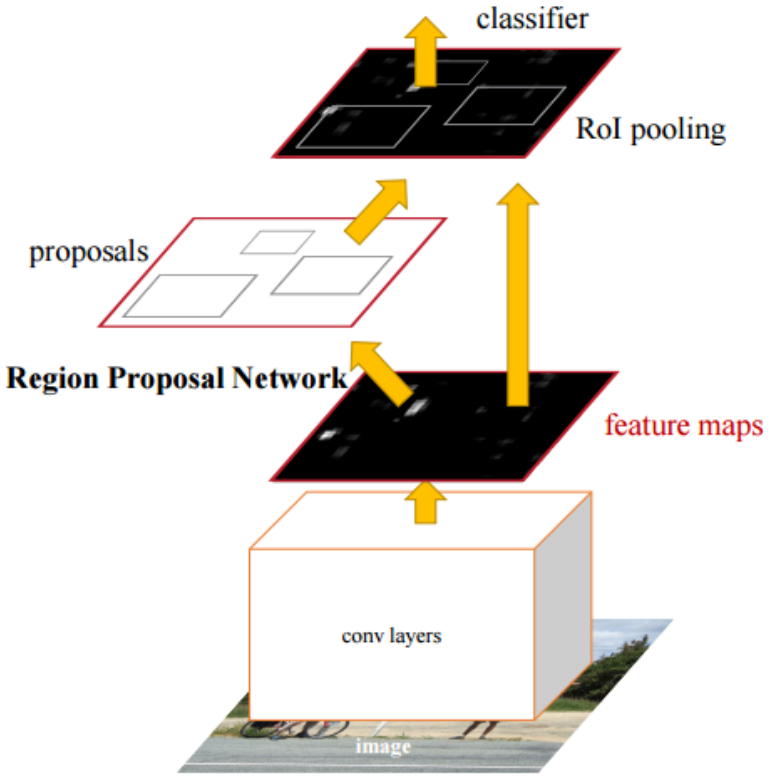
Faster RCNN Detector

A Faster Region-based Convolutional Neural Network (RCNN) detector is a multiclass detector. It is composed of a modified CNN, that includes a layer known as Region Proposal Network (RPN), responsible for outputting regions of interest from images that can later be classified by the last layers of the CNN.

The implementation of Faster RCNN detection in MatLab is performed by applying specific m-files and built-in function provided by MathWorks through the Computer Vision System Toolbox. In the implementation of this method, the training of the detector takes place first, off-line. Then, it can be applied to images (on-line). In this subsection, the structure of the detector is first covered. Then, the training implementation is treated and finally, the detection implementation.

The structure of the particular type of CNN used in this method can be described as follows. First, there is a series of convolutional layers that extract features from input images, creating activation maps. Then, the RPN is applied. The RPN is a convolutional network that takes activation maps and outputs regions of interest based on the location of features. Next, each area enclosed by a ROI is passed to other convolutional layers, that finally output the classification scores for every ROI.

The training of a Faster RCNN is implemented by calling the function *trainFasterRCNNObjectDetector*, that takes a network and labeled images (from Matlab's Image Labeler App) as inputs and outputs a *FasterRCNNObjectDetector*. Two different networks have been used: AlexNet and VGG16. The training function uses gradient descent



Source: www.slideshare.net/JinwonLee9/pr12-faster-rcnn170528

Figure 4.24: Faster RCNN structure

(where backpropagation is applied, subsection 4.3.1) and performs transfer learning internally. It is composed of four consecutive training steps, two for specifically training the RPN and two for the CNN:

1. **Initial RPN training:** All parameters of the RPN are optimized using a set of training images with labeled bounding boxes.
2. **Initial CNN training:** The parameters belonging to the rest of the layers of the CNN are trained for classification of the image regions proposed by the RPN.
3. **RPN retraining:** This step constitutes a fine-tuning of the parameters obtained from the first step of the training.
4. **CNN retraining:** This step is the equivalent to the third step for the CNN layers that are not part of the RPN. The parameters of those layers are initialized to the values obtained at the second step.

A trained Faster RCNN detector is able to take an input image and output a series of detected objects of the classes for which the detector has been trained. This takes

place by using the *detect* method of the *FasterRCNNObjectDetector* object, that takes the detector itself and the image where detection is to take place as inputs and outputs a series of bounding boxes, scores and labels for the objects detected within them.

4.5 Evaluation of Detection Methods

The four detection methods presented in section 4.4 and implemented in MatLab were evaluated by being applied to a series of image sets from which precision and recall measurements were obtained. Average detection time for every method was also measured. This section first presents the methodology used for performing the evaluation and then compares the different methods based on the results obtained.

4.5.1 Methodology

The procedure for evaluating the quality of the different detectors implemented is composed of the following steps:

1. Training of the classifiers and detectors
2. Application of detection methods to evaluation image sets
3. Extraction of detection results for every detection method

Both the two classifiers (SVM and CNN) and the two detectors (ACF and Faster RCNN) are trained using the ImageNet set of images introduced at subsection 2.3.1, that contains images categorized in a number of classes, namely buoys and different types of vessels. Those classes were chosen in a particular way with the aim that variations in shape and appearance of objects within a given categories are considerably smaller that variations in shape and appearance among objects of different classes. For that reason there are two categories for sailboats: *small_sailboat* and *large_sailboat*, since creating a single sailboat category would imply including in the same class objects that share many features with boats or speedboats, such is the case for many small sailing boats, together with large vessels, like sailing ships, that have almost no resemblance to those small vessels.

Each classifier or detector was trained as described in their respective subsection in section 4.4, so the classifiers received as inputs the categorized images from ImageNet, whereas the detectors received those same images labeled (that is, with labeled bounding boxes around objects). For the classifier and detection methods based on CNN, the training made used of transfer learning, that is much more efficient that training the whole network. It is also worth noting that, for those same methods based on CNN, two different networks were used: AlexNet and VGG16. Table 4.2 contains all the classifiers

and detectors used:

	Conventional	CNN	
Classifier	SVM	CNN - AlexNet	CNN - VGG16
Detector	ACF	Faster RCNN - AlexNet	Faster RCNN - VGG16

Table 4.2: Classifiers and detectors used for evaluation

For every classifier or detector trained, there were different parameters that could be modified by the student, and that affect in different ways and degrees the performance of those classifiers and detectors. Due to amount of parameters that could be tuned and the significant amount of time necessary for training any classifier or detector for a given combination of parameter values, only certain parameters were tuned. Most notably the number of stages (for the ACF detector), or epochs (for the CNN based classifiers and detectors), that determine the total number of iterations that should take place in the optimization procedure of training. Different values were tried for every case, and the ones that yielded better results at evaluation were chosen. It is important to emphasize that the final combination of parameters chosen for any classifier or detector, although better that any other tried, might very likely not be the optimal one.

The trained classifiers and detectors are applied, as part of their respective detection methods, to various image sets that constitute the evaluation dataset. These image sets are: Hundested images, Singapore Maritime Dataset visible video and Singapore Maritime Dataset Near Infra-Red (NIR) video (presented all of them at section 2.3). In order to be able to extract detection results, these image sets have their corresponding Ground Truth (GT) tables, that contain, for every image, the labels and bounding boxes of the actual objects present in the image. Those GT tables can be provided together with the dataset (as is the case for the Singapore Maritime Dataset) or built by the student with MatLab's Image Labeler App (as happens to Hundested images). Results are obtained for every combination of detection method (taking into account that, for CNN based methods there are two implementations: one with an AlexNet network and the other with VGG16) and image set, and consist of precision-recall curves. Also, average detection times are registered and used to compare detection speed.

It is necessary to introduce some concepts for a proper interpretation of precision-recall curves. First of all, it needs to be taken into account that results regarding object detection are based on the comparison of the actual objects present on the images (that are represented by GT data) and the objects detected by the detector. If the bounding box of a detected object corresponds to that of a GT object, and so do their labels (that is, the object is both correctly localized and classified), there is a match.

The criteria for determining the correspondence of detected and GT overlapping bounding boxes is based on a detection evaluation metric known as Intersection over Union (IoU). The information about this metric is based on [Zan17] and [Ros16]. Intersection over Union is a measurement of the overlap of two bounding boxes, that is defined as:

$$IoU = \frac{Area(BB_d \cap BB_{GT})}{Area(BB_d \cup BB_{GT})} \tag{4.24}$$

, where $BB_d \cap BB_{GT}$ represents the intersection of the detected and GT bounding boxes, and $BB_d \cup BB_{GT}$ their union.

This measure is used in many detection competitions, such as Pascal VOC Challenge ([Arl18]). In order to accept or reject the match between a detected and a GT bounding box, a threshold for the IoU measurement needs to be set. In this thesis, the standard value of 0.5 that is normally used in detection competitions ([Arl18]) has been used. It is worth noting that a match not only requires the value of IoU to be equal or higher than the threshold, but the category of the object has to be determined correctly.

The next definitions (True Positive, False Postive and False Negative) take the concept of match or correspondence as a starting point and allow to introduce the expressions for precision and recall. These definitions are based on [Zan17] and [Shn14]:

- **True Positive (TP):** A ground truth object is correcly detected.
- **False Positive (FP):** A detected object does not correspond to any ground truth object.
- **False Negative (FN):** A ground truth object is not detected.

Precision is the fraction of detected objects that are correct. The higher its value, the lower the possibility that a detected object is wrongly reported (something referred to as false alarm). It is mathematically expressed as the number of true positive detections (N_{TP}) relative to the sum of true positive and false positive detections (N_{FP}):

$$precision = \frac{N_{TP}}{N_{TP} + N_{FP}} \tag{4.25}$$

Recall is the fraction of Ground Truth objects correctly detected. The higher its value, the lower the likelihood of missed detection. It can be expressed mathematically as the number of true positives relative to the sum of true positives and false negatives (N_{FN}):

$$recall = \frac{N_{TP}}{N_{TP} + N_{FN}} \tag{4.26}$$

Precision and recall are both important measurements for the evaluation of object detection. They are inversely related, and a good way to represent such relationship is by plotting the precision-recall curve. This type of plot is widely use for detection evaluation in the most important object detection competitions. The information regarding precision-recall curve is based on [Arl18].

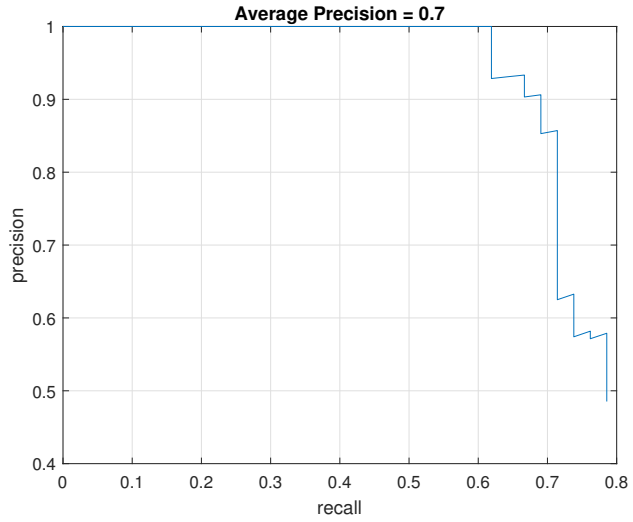
A precision-recall curve is calculated from all the true positive and false positive detections obtained from all the images of a given image set. Those detections (this includes both true positives and false positives) are ordered by their classification score, from higher to lower. Then, an accumulative process takes place, in which, for every detection (as stated before, in descending score order) the values of precision and recall are computed, taking into account all the previous detections. For the computation of recall, in particular, the number of true positives is divided by the total number of GT objects. That process continues until there are no detected objects left. Algorithm 1 shows the sequence of steps necessary for building a precision-recall curve.

Algorithm 1 Precision-recall curve computation

Input: $TP\_scores, FP\_scores, N_{GT}$
Output: $precision\_vector, recall\_vector$

$scores = [TP\_scores, FP\_scores]$
$scores = scores(higher \text{ to } lower)$
for $i = 1, i++, i \leq num\_scores$ **do**
  if $scores(i)$ is TP **then**
    $N_{TP}++$
  else
    $N_{FP}++$
  end if
  $precision\_vector(i) = \frac{N_{TP}}{N_{TP}+N_{FP}}$
  $recall\_vector(i) = \frac{N_{TP}}{N_{GT}}$
end for

From the computation of the precision and recall vectors in algorithm 1, it can be deduced that the final value both for detection and for recall ($precision(num\_scores)$ and $recall(num\_scores)$) corresponds to the total value of that measurement. For that reason, in subsection 4.5.2, every time precision or recall values are mentioned, they have been obtained by looking at the end of the precision-recall curve. The curve starts at $precision = 1$, $recall = 0$, that in the graph corresponds to the upper left corner, and ends at $precision(num\_scores)$, $recall(num\_scores)$, that corresponds to the lower right corner. Image 4.13 shows a typical precision-recall single class curve.



Source: se.mathworks.com/help/vision/ref/evaluatedetectionprecision.html

Figure 4.25: Example of precision-recall curve

An important value that can be obtained from a precision-recall curve is the Average Precision (AP). For an unambiguous computation of AP, 11 equally spaced recall points are chosen, from 0 to 1: $[0, 0.1, 0.2, ..., 1]$. AP corresponds to the average value of the precision across all those recall values. It can be computed by applying the following equation to the precision values of a precision-recall curve:

$$AP = \frac{1}{11} \sum_{recall_i} precision(recall_i) \tag{4.27}$$

,where 11 corresponds to the number of recall points chosen, $recall_i$ refers to the recall value for a specific point and $precision(recall_i)$ is the maximum precision for any recall larger than or equal to $recall_i$:

$$precision(recall_i) = max_{\tilde{r} \geq recall_i} precision(\tilde{r}) \tag{4.28}$$

Given a certain pair of total values of precision and recall, this variable (AP) rewards detectors that give higher classification scores to true positive detections.

For multiclass detectors, a value that combines AP values for every class is obtained. It is known as mean Average Precision (mAP), and this value is used for determining the performance of detectors in detection competitions ([Arl18]). As its name suggests, it takes the AP value for every class (for which there are GT objects) and calculates the mean.

The specific types of precision-recall curves used for reflecting evaluation results are now presented. For every detection method being applied to any of the evaluation image sets, two precision-recall curves are generated. One of them evaluates localization, that is, it assesses the quality of a detector when it comes to finding ROIs in images, regardless of their classification. This is achieved by considering all detected objects as part of one category, so that there is no class distinction between objects. The other curve corresponds to a multiclass detection curve, in which there is a detection curve and an AP value per every class. That is, both localization and classification are being taken into account in the curve.

4.5.2 Results

In this subsection, the results obtained from the application of every type of detector to each evaluation dataset are presented and discussed, with focus on the comparison between detection methods. First, MatLab implementation aspects are covered. Next, the results are presented for every image set.

Precision-recall curves can be obtained in MatLab by just applying the function *evaluateDetectionPrecision* to detection results. It takes as inputs the detection results and the GT data for a given set of images and outputs precision and recall values, together with average precision, for every class of objects. Precision and recall can be plotted, obtaining the aforementioned curve. Averate detection time is computed using the pair *tic* - *toc* functions just before and after the detection function and then calculating the average value for the whole image set. Regarding detection time, it should be noted that its values are not valid for real time implementation (such an implementation would require the use of a more suitable software than MatLab), but rather to compare detection methods.

The reader should bear in mind that, within this subsection, the terms ”precision” and ”recall” of a given curve refer to total precision or recall of that curve. Those values can therefore be obtained by looking at the end of the precision-recall curve (bottom right corner of the curve). Note also that, in images showing detected objects, red bounding boxes are GT and yellow bounding boxes are detected objects.

One last consideration seems pertinent before presenting the results. Absolute values of precision, recall or AP don’t have any practical meaning by themselves: they quantify the quality of an object detector, but it is only when compared to the minimum required quality for a certain application that they allow to determine if a detector is good enough (for the specific task) or not. The task of object detection in this thesis is the detection of objects in the surroundings of the vessel such that collisions can be avoided and a safe navigation can take place.

That means that it is required that all objects in the vicinity of the vessel must be properly located and classified, whereas objects further away (such as near the horizon line) are not so relevant, although it is also desired to detect them. This means that recall values should be high, but how much depends on the specific situation: if most objects are close to the horizon, low recall values would be acceptable for a safe navigation, whereas if there are objects only close to the vessel, recall should be 1 or close to 1 (the possibility of using a distance based recall is introduced in subsection 4.5.3). That is, no missed detections in the vicinity of the vessel are acceptable.

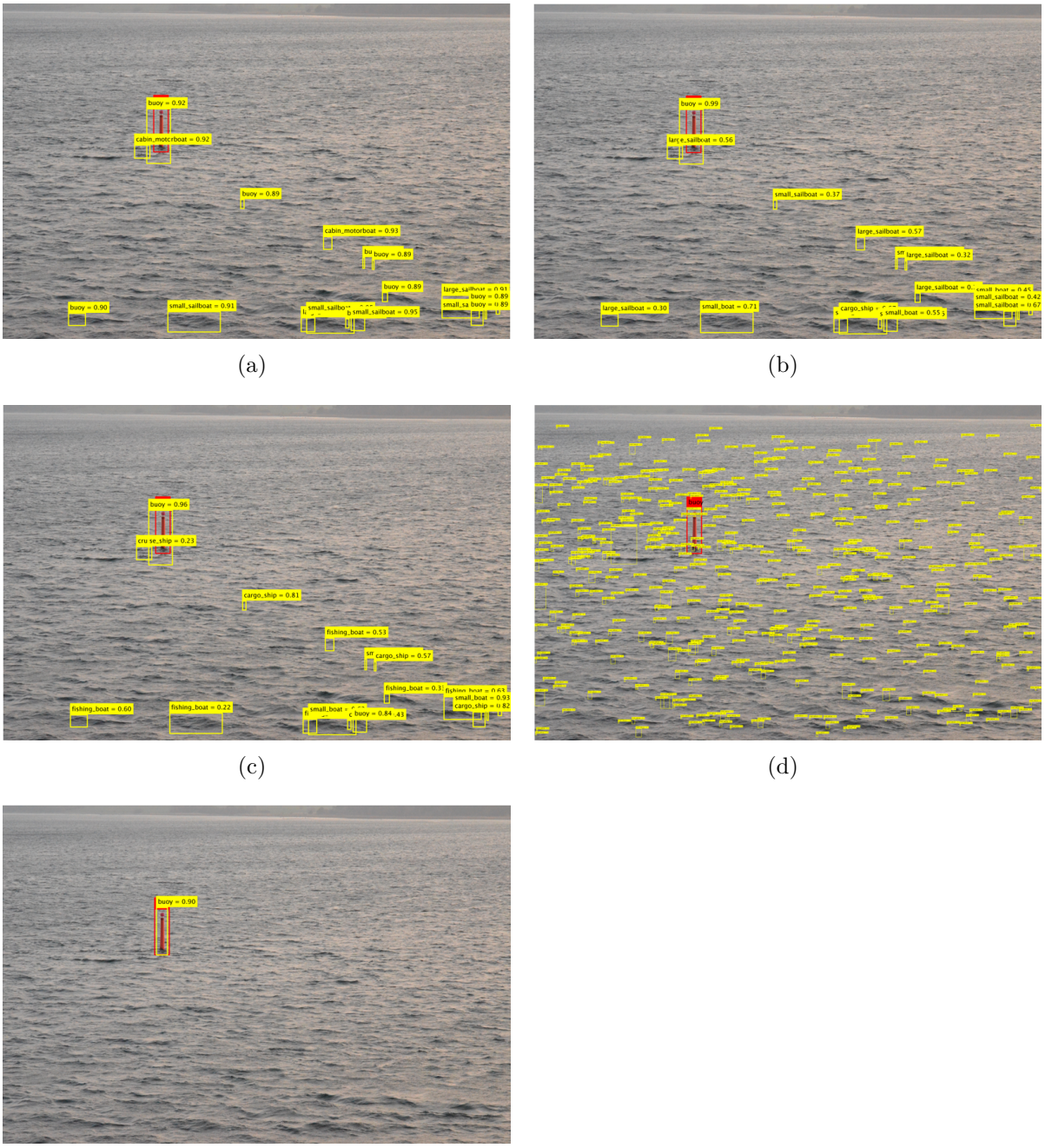
On the other hand, any false positive that might be detected does not create a risk in itself if is due to mistaking the background for an object. However, if the impact on the guidance system of the ship is taken into account, the detection of another vessel where there is only water would result in a series of unnecessary manoeuvres, thus producing an inefficient and erratic navigation. Any false positive that is the result of a missclassification of an object is potentially more dangerous, since the vessel should react differently to different types of objects, and missclassifying one might result in taking a wrong course of action, with possibly serious consequences. For all the above, precision must be high, with values above 0.9 guaranteeing a safe navigation.

The practical considerations for comparing precision and recall values introduced in the two previous paragraphs are taken into account when evaluating detection performance in this subsection. The reader should note that the detection is performed for single images. The possibility of decreasing the number of false positives by applying optical flow to a sequence of images is treated in 4.5.3.

Hundested Images

Hundested image dataset (subsection 2.3.4) contains relatively few objects, when compared with the other image sets used for evaluation. Among them, the most common ones are buoys. There are also some ferries, cargo ships, small boats and small sailboats. Many of those objects appear quite distant. The light conditions of many of the images, having being taken in the late evening, with Sun reflections in the water and not very clear visibility of the objects color and texture, are not particularly favourable either. Some other images present near objects with clear daylight conditions or overcast conditions.

Not all detection methods could be applied to this image set. In particular, the Faster RCNN detector using the VGG16 model gave errors when applied to Hundested images related to lack of memory space. The student tried, unsuccessfully, to solve the problem, so that specific detector is not evaluated for this set of images. As for the rest of detectors, images in 4.26 show an example of a particular image from the Hundested set featuring a buoy being detected by the different detectors.



(a) (b)

(c) (d)

(e)

Figure 4.26: Object detection on Hundested sample image: (a) ROI + SVM; (b) ROI + CNN (Alexnet); (c) ROI + CNN (VGG16); (d) ACF; (e) Faster RCNN (AlexNet)

Images (a), (b) and (c) from figure 4.26 correspond to detection methods that use the ROI finder algorithm. That explains why they have localized exactly the same bounding

boxes. They differ in the classification technique, though, and that is reflected in the labels and scores of those bounding boxes. SVM classification correctly classifies the bouy, but also gives high scores to false positives (waves that have been mistaken for objects by the ROI finder), whereas the CNN classification technique, regardless of the specific NN used, also classify correctly the buoy, but give lower scores to false positives (that could then be discarded by applying a score threshold). ACF detection produces a large amount of false positives, mistaken any part of the water that stands out from the background, such as waves, for objects. Faster RCNN does not produce any false positive, and correcly locates and classifies the bouy.

From all the classifier-based detectors, results for only one of them are shown, since the localization curve is the same for all of them and classification-wise, differences are not significant. In particular, the precision-recall curves shown correspond to those of the CNN classifier method using AlexNet network, that yields slightly better results than the other methods.

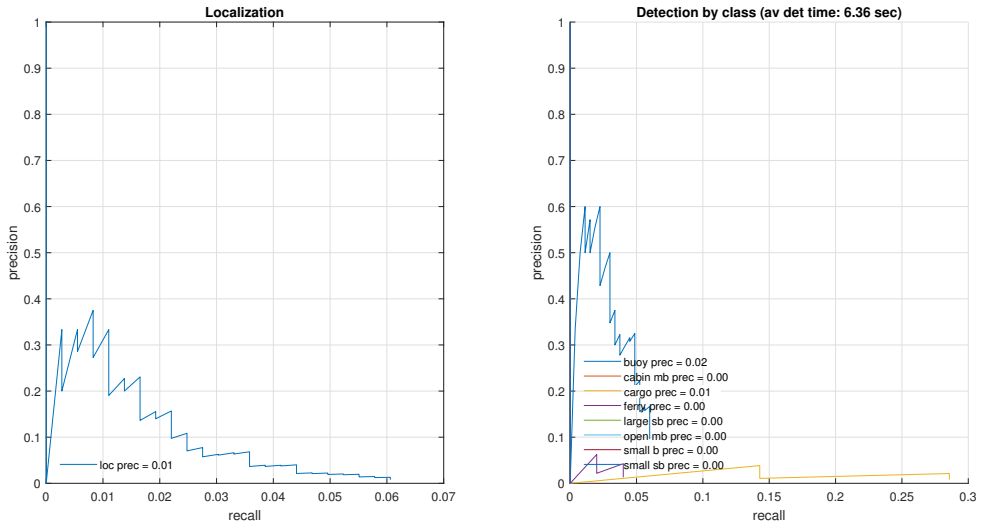


Figure 4.27: Precision-recall curve for ROI finder + CNN classifier (AlexNet)

From the localization curve (left) on figure 4.27, it is clear that the ROI finder algorithm achieves very poor results, both in precision and recall. That is, there are many false positives (as seen in pictures (a), (b) and (c) form image 4.26), reflected in the small values of precision, but also many missed detections, reflected in the even small values of recall (in total, only 6% of all the GT objects were localized). Localization AP is only 0.01. Per category (right on figure 4.27), buoys are more precisely detected than any other class (AP of 0.02), but both AP, precision and recall values are very low even for that class.

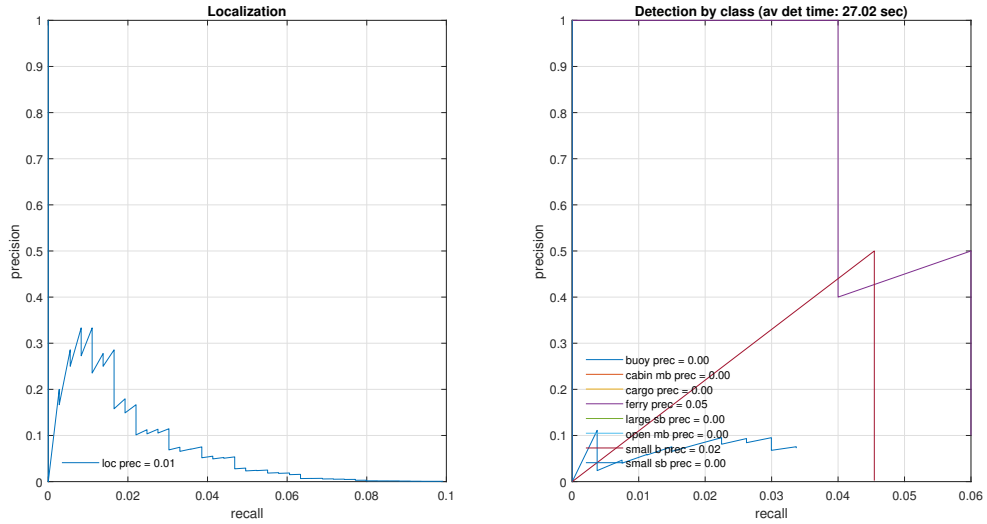


Figure 4.28: Precision-recall curve for ACF detector

ACF detector performace (figure 4.28) is similar to that of classifier-based detectors, particularly when it comes to localization. Precision is very low (high likelyhood for false alarms), in line with image (d) from figure 4.26. When it comes to recall, approximately 10% of all the GT objects are localized. Localization AP is 0.01. Regarding per class detection, precision is very low for every class (the highest corresponding to ferry at around 10%), while recall values don't even reach 7% for any category. That is, not only there are many detections that mistake background with objects, but the GT objects to be detected are most of the times not detected. The highest AP corresponds to the ferry class, and it has a value of only 0.05.

Precision-recall curves for the Faster RCNN detector, using AlexNet, (figure 4.29) yield good precision values, but still low recall values. When it comes to localization, there is a total precision of 60%, which means that false alarm rate is 40%, a value not acceptable in practice but considerably higher than the ones from the previous methods. Localization recall, however, is still lower than 10%. Localization AP has a value of 0.07, which is considered low, mainly due to the low recall. As of per class detection, buoys present a high precision (higher than 80%) but low recall. That is, most detected buoys are correct, yet most GT buoys are not detected at all. Buoy AP is 0.07. Ferry has a precision of 1 (no false detections for that category) and a recall of 0.1, so AP is 0.1, the best value for any category. From the rest of classes, only small boats perform any better than for the previous detectors.

Table 4.3 summarizes results for every method applied to Hundested images, including localization AP, mAP and average Detection Time (aDT).

From localization AP and mAP values, it is clear that the Faster RCNN detector

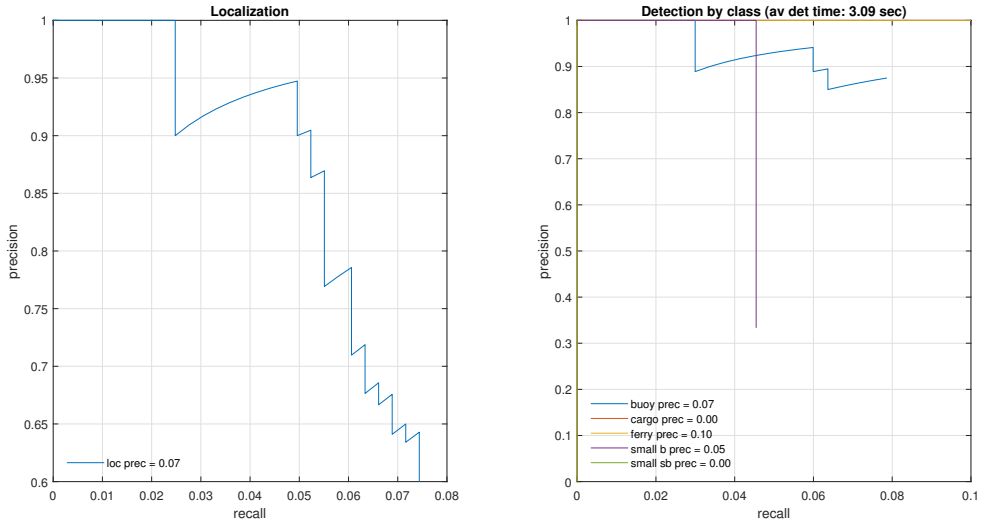


Figure 4.29: Precision-recall curve for Faster RCNN detector (AlexNet)

	ROI + SVM	ROI + CNN (AlexNet)	ROI + CNN (VGG16)	ACF	Faster RCNN (AlexNet)
Localization AP	0.00	0.01	0.00	0.01	0.07
mAP	0.01	0.00	0.00	0.01	0.02
aDP (seconds)	4.22	6.36	8.03	27.02	3.09

Table 4.3: Evaluation results for Hundested images

outperforms the rest (although AP values for that detector are quite low). It also has the lowest average detection time, while ACF's average detection time is much higher than the rest (mainly due to the large amount of false positive detected at every image by this detector).

Singapore Maritime Dataset visible video

The visible video of the Singapore Maritime Dataset (introduced at subsection 2.3.2) is made up of color pictures (frames) taken at the strait of Malacca. Being part of one of the most important sea routes in the world, the one connecting Europe and the Middle East to East Asia, the most frequent objects in these images are cargo ships. There are also some ferries and sailboats. Most of the vessels appear far away from the camera, and overlapping between them is common. Since the images are stored as video frames, an iterative process was implemented in MatLab, by which a PNG image was produced out of each frame, and then objects were detected on it. For this image dataset, unlike for Hundested images, all the different detectors were applied. Images in 4.30 show an example of detection peformed to a sample frame from the Singapore Maritime Dataset

visible video by all the detectors.

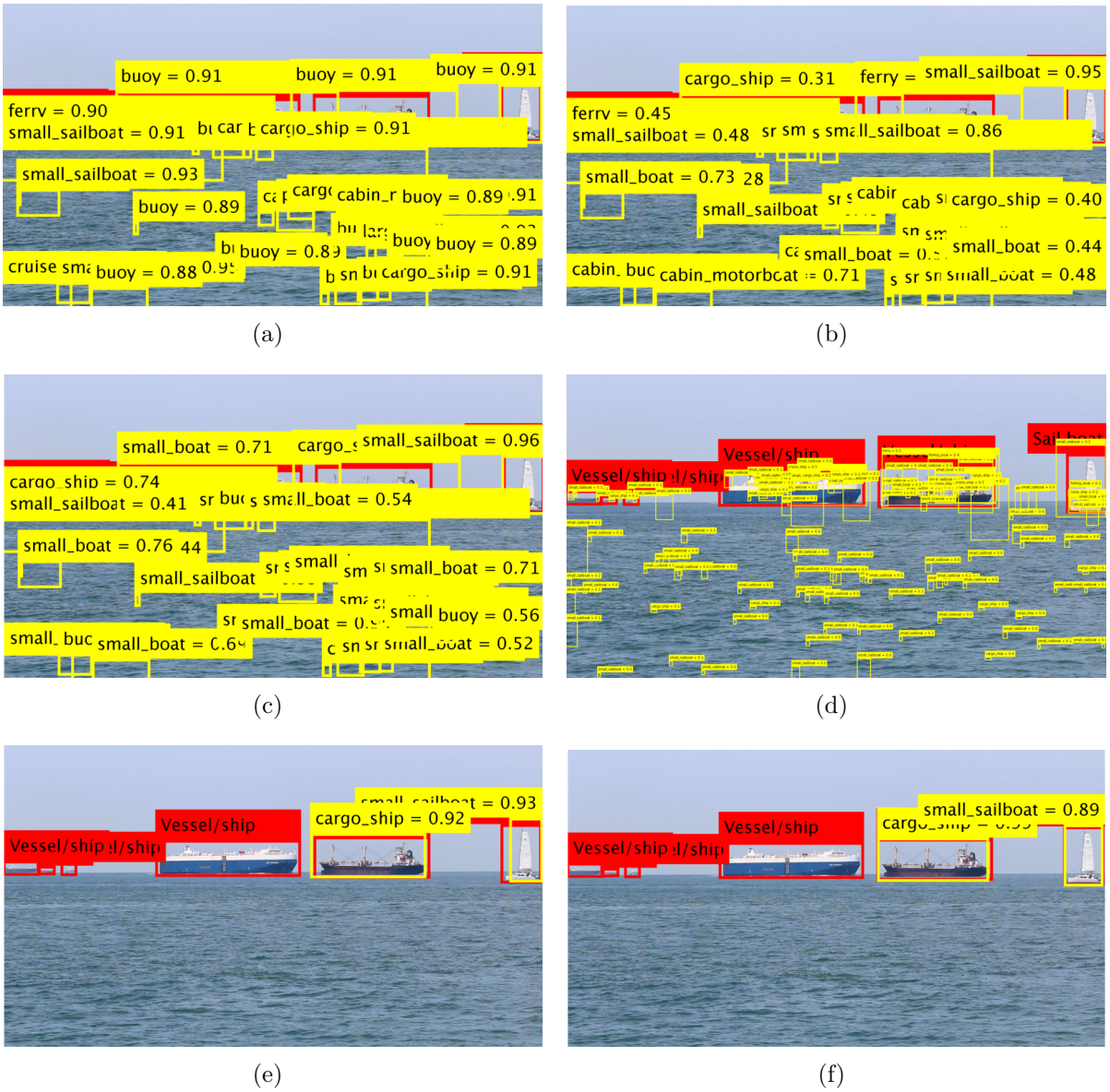


Figure 4.30: Object detection on SMD visible video sample frame: (a) ROI + SVM; (b) ROI + CNN (Alexnet); (c) ROI + CNN (VGG16); (d) ACF; (e) Faster RCNN (AlexNet); (e) Faster RCNN (VGG16)

As was the case for Hundested images, pictures (a), (b) and (c) from 4.30 present the same bounding boxes, as the ROI algorithm is common to those methods. Some GT objects are localized, but there are many false positive localizations. As for classification, SVM gives poor results, labeling some GT objects such as a small sailboat as buoys, and giving high scores to false positives. On the other hand, both CNN classifiers give high

scores to the correctly classified GT objects, and relatively low scores to false positives. The ACF detector is able to correctly detect, although with very low confidence, some GT objects in the image (a cargo ship and a small sailboat), but also generates a large amount of false positives (although with even lower scores). Faster RCNN detectors correctly detect a couple of GT objects with high scores, and don't produce any false positive. However, they fail to detect all the objects present on the image (low recall).

Results for SVM based detector are not shown, since they are quite silimar to those of CNN based detector using AlexNet. From the Faster RCNN detectors, being their performances similar, also the results for only one of them are shown (VGG16 has been chosen, but the results of AlexNet could also have been chosen). It is worth noting that the categories for these series of results differ from the ones used for Hundested or for the final experiments. This is due to the fact that the labels for GT objects provided by the Singapore Maritime Dataset are not exactly the same as the ones used by the student. A conversion between categories has been used.

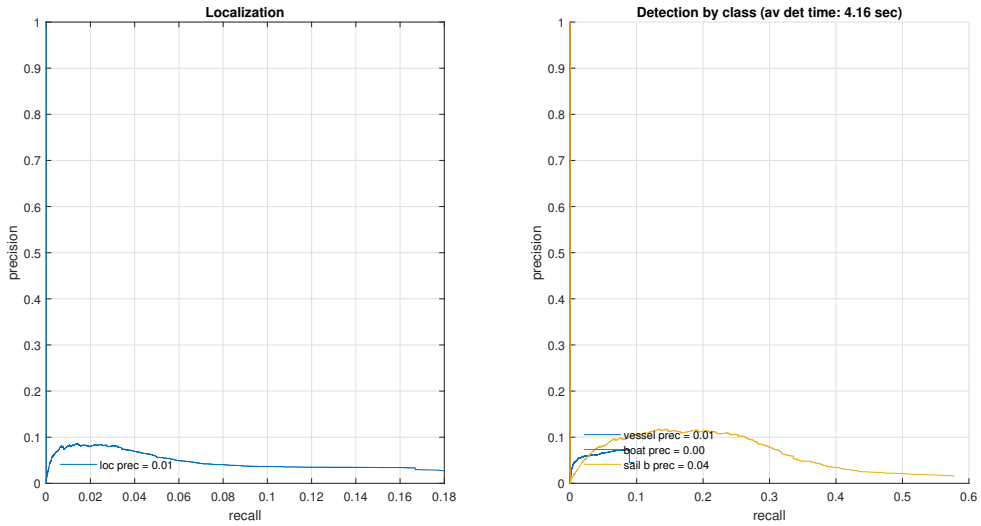


Figure 4.31: Precision-recall curve for ROI finder + CNN classifier (AlexNet)

The localization curve for classifier-based detectors (left curve in figure 4.31) presents a very low value of precision (less than 10% of all the objects detected actually match a GT bounding box), and low values of recall (almost 20% of all GT boundinb boxes are localized). Localization AP is thus only 0.01. The class detection curve shows, for CNN based (AlexNet) detector, very low values of precision for all the classes (always below 10%), but for one of them there is a relatively high value of recall: almost 60% of all sailboats in the dataset are correctly matched. The difference between the recall of this category and any other (namely, vessel category, with a recall of around 9%) might be due to the fact that sailboat objects appear closer in the dataset images than

most vessels (cargo ships, cruise ships and ferries). Sailboats present the highest AP score, with a value of 0.04. The class detection precision-recall curve for the SVM based detector is very similar to the one presented here, with slightly lower values.

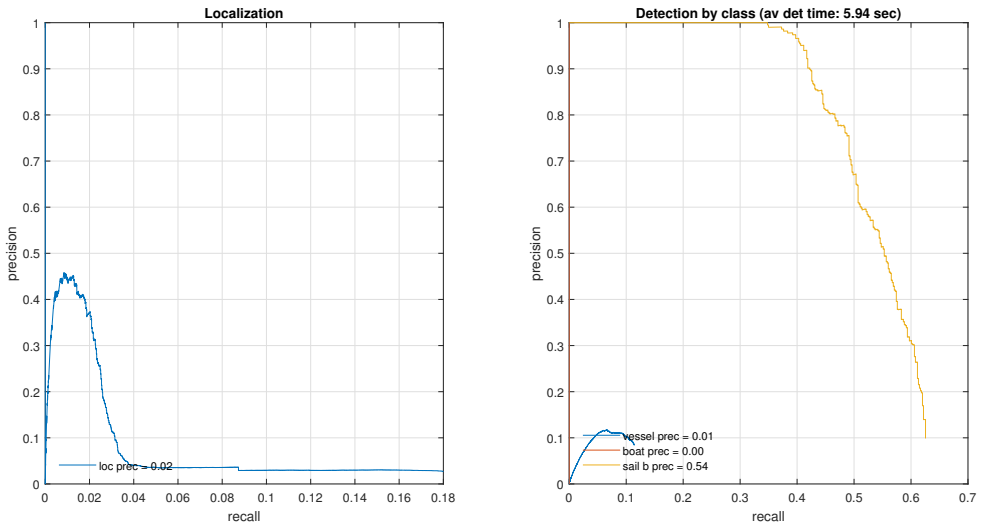


Figure 4.32: Precision-recall curve for ROI finder + CNN classifier (VGG16)

Precision-recall curves for CNN based detector using VGG16 differ from those of the other two classifier-based detectors presented above, not in its precision and recall final values, that are quite similar, but in its shape. Basically, scores are more consistent for this detector (higher scores for true positives, lower for false positives), which produces the shapes seen in figure 4.32 and gives higher AP values: 0.02 for localization and 0.54 for the sailboat category.

Precision-recall curves for the ACF detector are shown on figure 4.33. The localization curve has a very low value for precision (ACF detector usually generates many false detections, for example mistaken waves for vessels) and a low value for recall, comparable to that of the ROI algorithm. Localization AP has a value of 0.02. Regarding class detection, the precision-recall curve on the right of the figure show, for sailboats, very low precision but a recall higher than 60%, whereas for vessels, small values of both. The higher AP corresponds to vessels, with a value of 0.05 (higher precision values than sailboat along the curve).

Precision-recall curves for the Faster RCNN detector using VGG16 are shown on figure 4.34. Those curves for the same type of detector using AlexNet are very similar. When it comes to localization, precision is very high (over 93% of all the bounding boxes detected correspond to a GT bounding box), and recall, although not being high (almost 25%), allows to detect near one fourth of all the GT objects. Localization AP is 0.23.

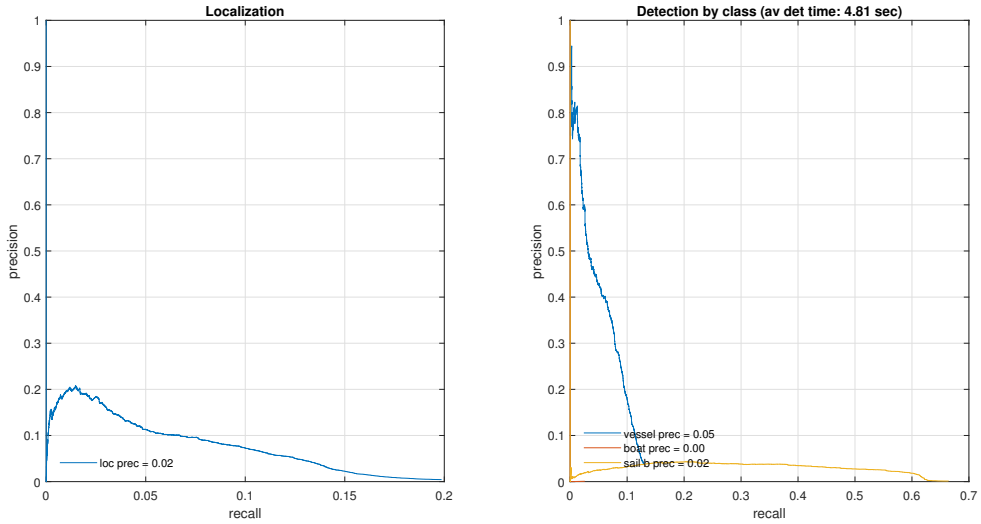


Figure 4.33: Precision-recall curve for ACF detector

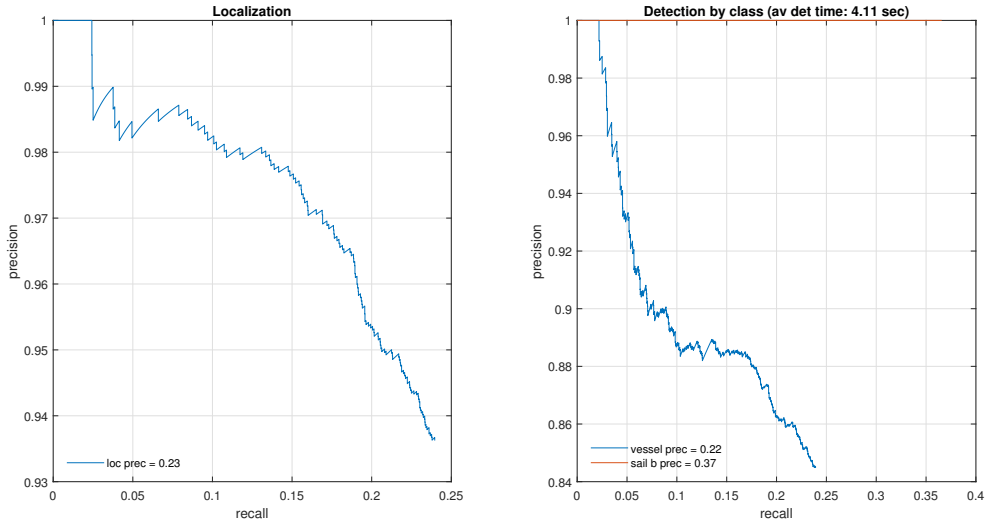


Figure 4.34: Precision-recall curve for Faster RCNN detector (VGG16)

Regarding classes detection, different types of vessels are detected with high precision and medium-low recall. Actually, the precision-recall curve for vessels is similar to that of localization, just presenting somewhat lower precision values. That similarity stems from the fact that vessels make up most of the objects present in the images. The best performing category, however, corresponds to sailboats, that have a precision of 100% (not a single false positive) and a medium recall of 37%. Unsurprisingly, the highest AP corresponds to this category, with a value of 0.37.

Table 4.3 summarizes results for every method applied to the visible video of Singapore Maritime Dataset, including localization AP, mAP and average detection time (aDT).

	ROI + SVM	ROI + CNN (AlexNet)	ROI + CNN (VGG16)	ACF	Faster RCNN (AlexNet)	Faster RCNN (VGG16)
Localization AP	0.01	0.01	0.02	0.02	0.22	0.23
mAP	0.01	0.02	0.18	0.02	0.36	0.20
aDP (seconds)	3.65	4.16	5.94	4.81	1.93	4.11

Table 4.4: Evaluation results for Singapore visible images

The values for localization AP and mAP on table 4.4 show a significant different between the quality of Faster RCNN detectors and the rest, in favour of the first. Although, within that method, using one network or the other does not produce major differences in performance, it does produce a great difference in detection time, being the detector using AlexNet more than twice as fast as the one using VGG16.

Singapore Maritime Dataset NIR video

The NIR (Near Infrared) video of the Singapore Maritime Dataset (subsection 2.3.2) is composed of greyscale frames from the near infrared range of the light spectrum, taken in waters of Singapore. In the foreground of many of the images there is a ferry, whereas in the background there are various cargo ships, that are blurred because of the distance and that overlap each other. Since the images are stored as video frames, an iterative process was implemented in MatLab, by which a PNG image was produced out of each frame, and then objects were detected on it. For this image dataset all the different detectors were applied. Images in 4.30 show an example of detection peformed to a sample frame from the Singapore Maritime Dataset visible video by all the detectors.

Images (a), (b) and (c) from figure 4.35, that correspond to classifier-based detectors, all use the ROI finder algorithm to localize objects, so the detected bounding boxes are the same in the three methods. From the sample image it can be noticed that the objects in the background of the image are not detected at all, whereas, for the ferry in the foreground, various boxes are drawn (one-to-many match), giving high scores for various wrong categories. The ACF detector (d) produces many small bounding boxes, that are false detections, particularly in the ferry, where there is a box encompassing the whole object, but missclassifying it. The only two methods that manage to properly detect the ferry in the foreground, without false positives, are the ones using Faster RCNN detection, although they also fail to detect any other object.

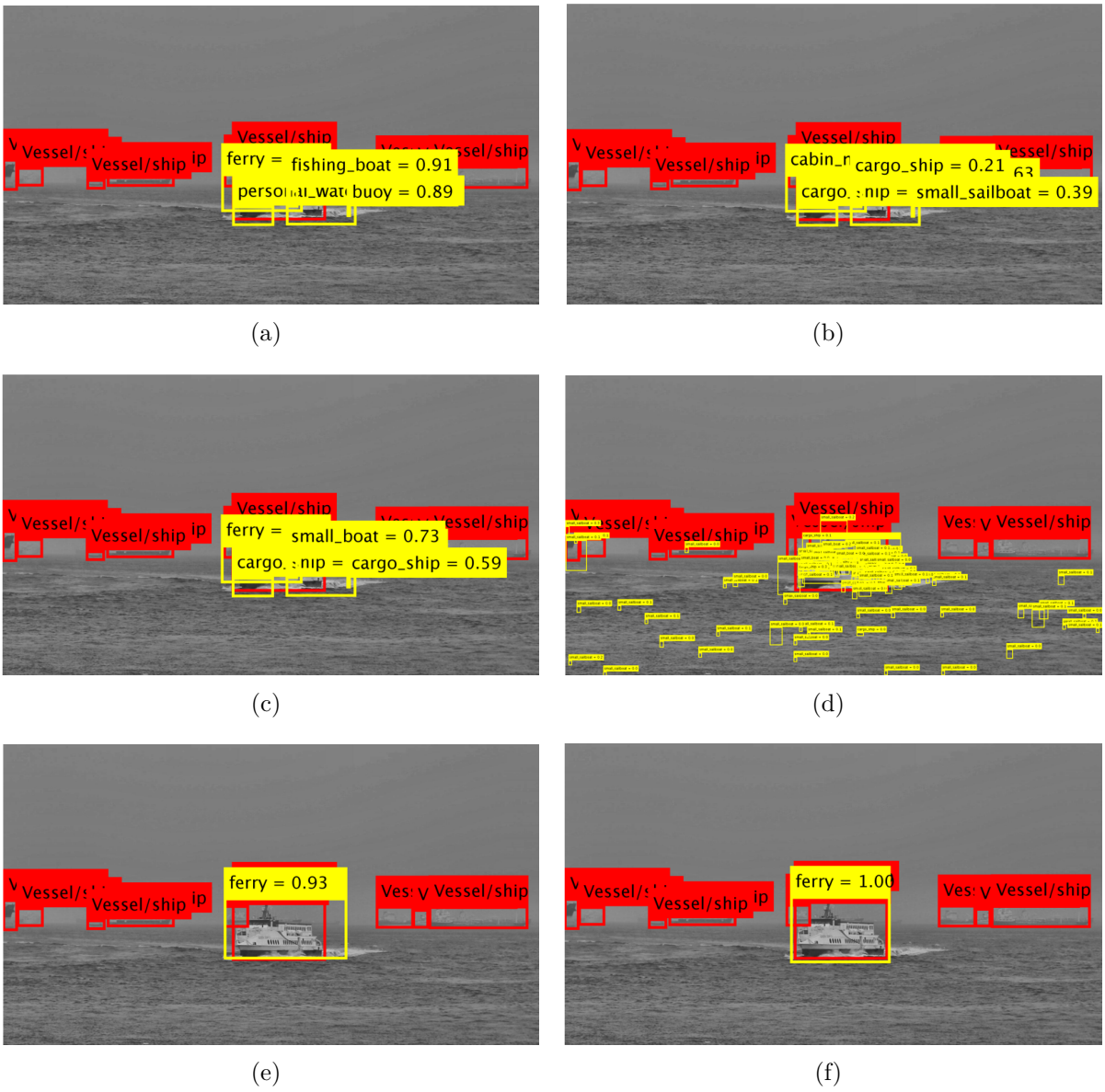


Figure 4.35: Object detection on SMD NIR video sample frame: (a) ROI + SVM; (b) ROI + CNN (Alexnet); (c) ROI + CNN (VGG16); (d) ACF; (e) Faster RCNN (AlexNet); (e) Faster RCNN (VGG16)

Results for classifier-based detectors are very similar between them. For that reason, only results for one of them (SVM) are shown. Results for Faster RCNN detectors are very similar between them, but they are quite different from those of the ACF detector. Thus, results for one of the Faster RCNN detectors (VGG16) and for the ACF detector are shown. As is the case for Singapore Maritime Dataset visible video, the classes of GT objects of the NIR video differ from those used by the student, that are applied to Hundested and final experiment image set objects. For that reason, a conversion from

the student defined classes to the Singapore Maritime Dataset ones has been used.

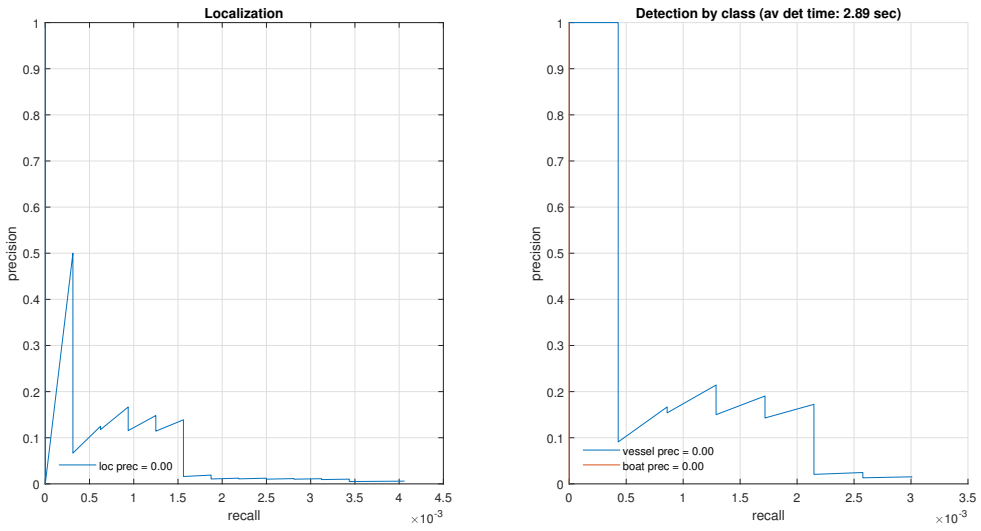


Figure 4.36: Precision-recall curve for SVM based detector

Both precision-recall curve at figure 4.36 are very similar, due to the presence on the dataset of basically one category. Both for localization and for vessel (that includes ferry) detection, precision and recall are extremely low (in both curves the two values are clearly under 1%), and, unsurprisingly, AP values are close to zero. The low value of precision is due to the false positives created by the ROI finder, whereas low recall stems from the missed detected objects in the background of images from the NIR dataset.

Localization and detection results are also similar between them for the ACF detector (figure 4.37), due to one class (vessel) predominantly appearing in the foreground of images and therefore being detected. For both curves, precision and recall are low (with values of 0.01 or lower), due to the large amount of false detections generated by the ACF algorithm and the incapability to detect objects in the background, respectively. Nonetheless, the good score distribution (higher scores for true positives, lower scores for false positives, in general) produces relatively good AP values, although still low and not acceptable for practical purposes. AP has a value of 0.04 for localization and of 0.02 for detection (mAP).

Precision-recall curves for the Faster RCNN detector using VGG16 as network look more like what is expected of a reasonably good detector, with perfect precision for high scores (flat part of the curve for low values of recall) and a sharp fall in precision for lower scores. Final precision is very high both for localization (95.5%) and for vessel detection (99.66%). The reason why precision is higher for a one class detection than for localization (in principle, for just one class detection, both values should be the same)

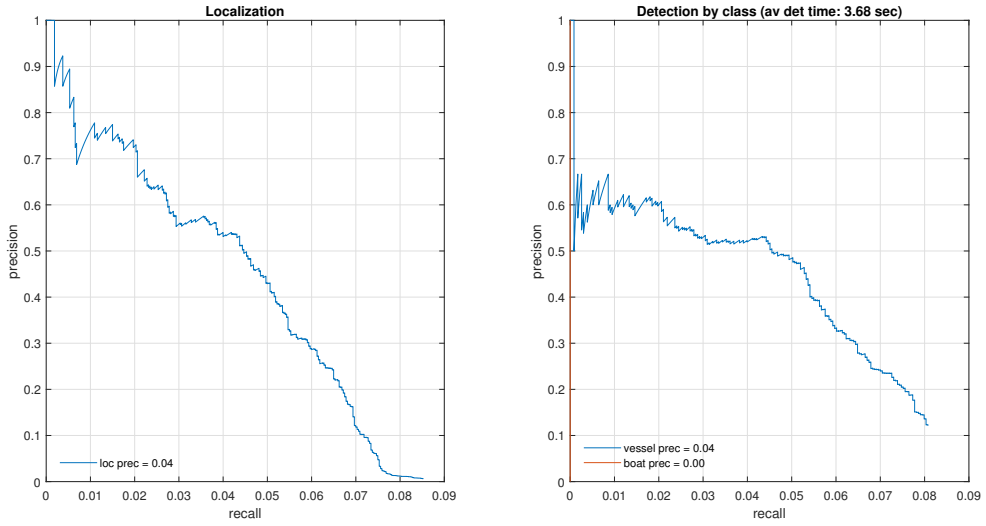


Figure 4.37: Precision-recall curve for ACF detector

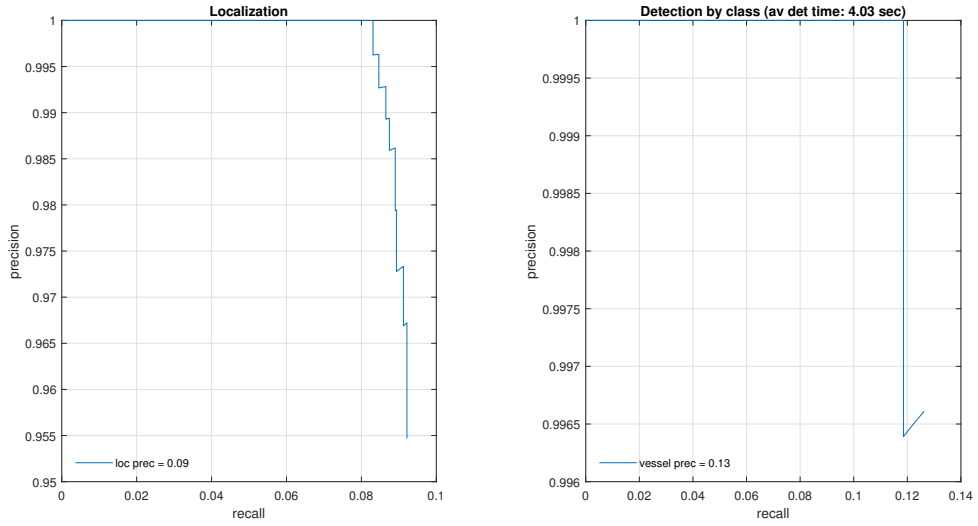


Figure 4.38: Precision-recall curve for Faster RCNN detector (VGG16)

is some missing data in the GT tables provided by Singapore Maritime Dataset, that also affects the previously presented results, although its impact is very limited. Recall, unlike precision, is low, with only around 10% of all GT objects being detected. As stated above, this is due to Faster RCNN detectors not being able to detect very distant objects, that, by nature, don't have many distinctive features. For that reason, AP values are not as high as precision distributions might suggest, with localization AP having a value of 0.09 and mAP having a value of 0.07.

	ROI + SVM	ROI + CNN (AlexNet)	ROI + CNN (VGG16)	ACF	Faster RCNN (AlexNet)	Faster RCNN (VGG16)
Localization AP	0.00	0.00	0.00	0.04	0.08	0.09
mAP	0.00	0.00	0.00	0.04	0.04	0.07
aDP (seconds)	2.89	3.83	5.47	3.68	1.85	4.03

Table 4.5: Evaluation results for Singapore NIR images

Table 4.5 sums up results from every detection method. Classifier-based detection methods have negligible values for localization AP and mAP, with the CNN classifier based detection method using VGG16 also presenting the highest detection time among all methods. ACF detection has low AP values and a medium range detection time. Faster RCNN methods present the highest AP values, with VGG16 yielding better detection AP (approximately double that of AlexNet) at the expense of a larger detection time (more than double). Although Faster RCNN methods clearly outperform the rest, their results are still poor (largest AP value of just 0.13), especially when compared to those for visible images from Singapore Maritime Dataset (4.4), so these methods seem to work better for images in the visible spectrum (more variety of NIR images should be evaluated in order to confirm that statement).

4.5.3 Further Comments on Evaluation

This subsection briefly introduces two concepts that could lead to improvements in detection results. One of them consists of a modified version of recall, the other takes into account a sequence of images, and not just one, to perform object detection.

A distance based recall takes into consideration the fact that, the closer an object is to the vessel, the more relevant its detection is for navigation. The current computation of recall considers all objects as equally important. Here, a modified computation of recall is proposed, where close objects would contribute more (N_{TP} or N_{FN} would be increased by one, depending on whether the object would be detected or not) than far away objects (N_{TP} or N_{FN} would increase by a value in the range $[0, 1]$) to the computation of recall. The distance of any object to the vessel would be approximately measured by computing the distance between the waterline of the object and the horizon line.

The detection methods implemented in this thesis only use the data contained in the image where detection is being performed. But, if the fact that each of these images is part of a sequence, where between one image and the two contiguous ones there is almost no difference (that is, these images are in practice frames of a video), the information extracted from the previous image could be exploited. A technique that could be used

for achieving this is optical flow constraint equation method ([Sca11]). Such technique would allow to determine where the objects detected in the previous image are in the current one. False positives where an object is detected where there is only background are very rarely repeated, and comparing the detected objects in the current frame with the expected ones from the previous one would allow to discard some of these types of false detections.

4.6 Summary

Chapter 4 covered four different object detection methods, from its theoretical foundations to the evaluation of their detection performances. At the beginning of the chapter, some basic theoretical concepts were introduced, treating digital images, image processing tools and convolutional neural networks principles. These theoretical basis were necessary for treating the four detection methods employed along the thesis, that were grouped into two categories: classifier-based detectors, that includes SVM classifier based detection (using an image ROI finder) and CNN classifier based detection (using the same image ROI finder), and pure detectors, that includes ACF detector and Faster RCNN detector. For each method, working principles and MatLab implementation aspects for both training and detection were covered, relating them to the previously exposed theory. Finally, the quality of those four detection methods was assessed by presenting an evaluation methodology that was applied to the output of every detection method being used on different image sets. That methodology was based on obtaining precision-recall curves and AP values for localization and detection. From the results, it was clear that Faster RCNN detectors constitute the most precise detection methods from the ones presented in the thesis.

CHAPTER 5

Experimental Implementation

This chapter presents the application of the four detection methods developed during the project to the image sets that correspond to the final experiment, with focus on the evaluation of detection performance. The chapter first introduces the image sets, detectors and evaluation methodology. It then presents the results obtained by using that methodology to evaluate the quality of the different detection methods applied to the image sets. The chapter is composed of the following sections:

- Section 5.1 introduces the detectors, image sets and evaluation methodology for detection.
- Section 5.2 presents and discusses the results obtained from applying the detection methods to the image sets.

5.1 Experiment

The detection experiment consists on applying the four object detection methods presented in section 4.4 to images from the Helsingør image dataset (2.3.3), both color and monochrome. For details about the methods and implementation aspects regarding their training or application, the reader is referred to subsection 4.5.1.

The images to which detection was applied, within the scope of this chapter, were taken by cameras that are part of the equipment employed by the DTU team developing the USV project. For that reason, their results are presented in this chapter, instead of the evaluation section of chapter 4. The evaluation takes place separately for color images (Helsingør color image set) and for monochrome images (Helsingør monochrome image set). The student built GT tables for both image sets using MatLab's Image Labeler App, so that results consisting on precision-recall curves could be obtained for every combination of detector and image set.

The methodology followed for the extraction of precision-recall curves, including all the related concepts, such as true positive, precision, recall, AP, etc., is not treated here, since it is already covered in subsection 4.5.1. The reader is referred to that subsection for details about the evaluation methodology and measurements.

The results presented in this chapter contain, as the ones in 4.5.2, two precision-recall curves: one assesses localization; the other one, detection as a whole (that is, localization and classification).

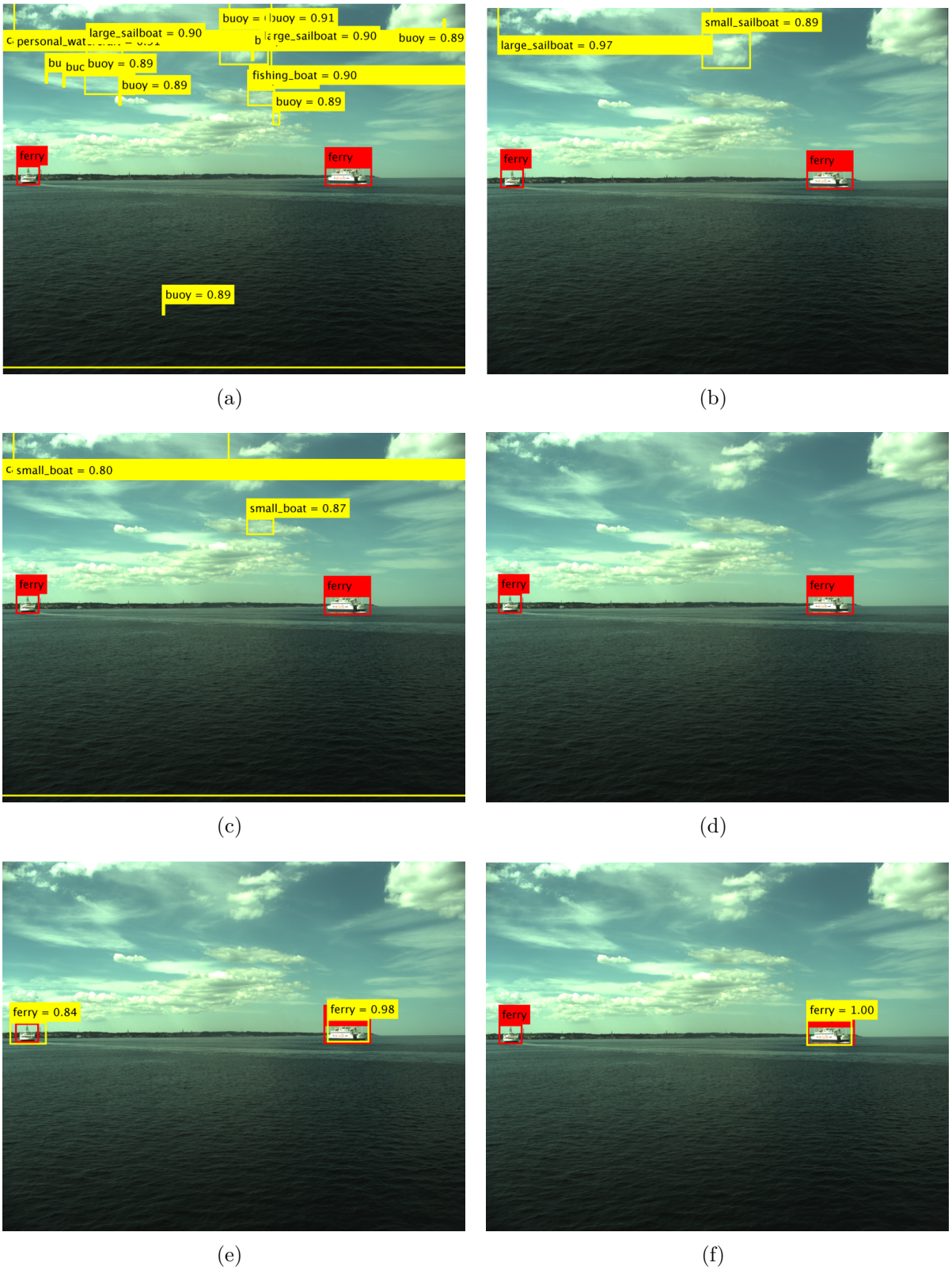
5.2 Results

In this subsection, results for every detection method applied to both Helsingør image sets are presented in form of precision-recall curves and discussed. The reader should be aware of the use of the terms ”precision” and ”recall” within this subsection: they refer to total values of their respective measurements. Therefore, they can be obtained by locating the end of the precision-recall curve (bottom right corner of the curve). Also take into account that, in images where detection has been used, red bounding boxes correspond to GT objects, whereas yellow bounding boxes correspond to detected ones.

5.2.1 Helsingør Color Images

Helsingør color images (subsection 2.3.3) were taken at the narrowest part of the Oresund strait, between the cities of Helsingør, Denmark, and Helsingborg, Sweden. The Oresund strait connects the Baltic sea to the North Sea, and is one of the busiest ferry routes in Europe. Thus, the image set contains many image with at least one ferry. The distance to the ferry is far to moderate for most images. Unfortunately, not many more types of vessels were present at the vicinity of the ferry were the equipment was mounted when the images were taken.

The images at figure 5.1 show the object detection performed by every detection method with every type of NN used at a sample image from the dataset. From the GT bounding boxes it is clear the presence of two ferries. The three classifier-based detection methods (images (a) to (c)) mistakenly detect objects in the sky, due to the presence of clouds that were not removed at the background subtraction step of the ROI finder (subsection 4.4.1), also given high classification score to those detections, which is undesirable. Furthermore, these methods fail to even localize any of the ferries. The ACF detector (d) does not detect any object in the image. As for Faster RCNN detectors, the one using AlexNet detects properly both ferries with high confidence, while the one using VGG16 only detects the closest one, but it does so with total certainty.



(a) (b)

(c) (d)

(e) (f)

Figure 5.1: Object detection on Helsingor sample color image: (a) ROI + SVM; (b) ROI + CNN (Alexnet); (c) ROI + CNN (VGG16); (d) ACF; (e) Faster RCNN (AlexNet); (e) Faster RCNN (VGG16)

Precision-recall curves are now presented. From all the classifier-based detectors, only the precision curve for one of them is shown. The reason being that all of them have very poor performances, so one example is enough to exemplify it. The CNN classifier based detector using VGG16 has been chosen, being its results slightly better than those of the other classifier-based methods. Results for the ACF detector are not shown, since no object was detected, which resulted in an empty detection curve. That performance starkly contrasts with that of the ACF detector for Hundested or Singapore Maritime Dataset images (subsection 4.5.2), where it produced many detections, most of them false positives. Finally, results for both Faster RCNN detectors (AlexNet and VGG16) are shown, since there is a clear difference between them.

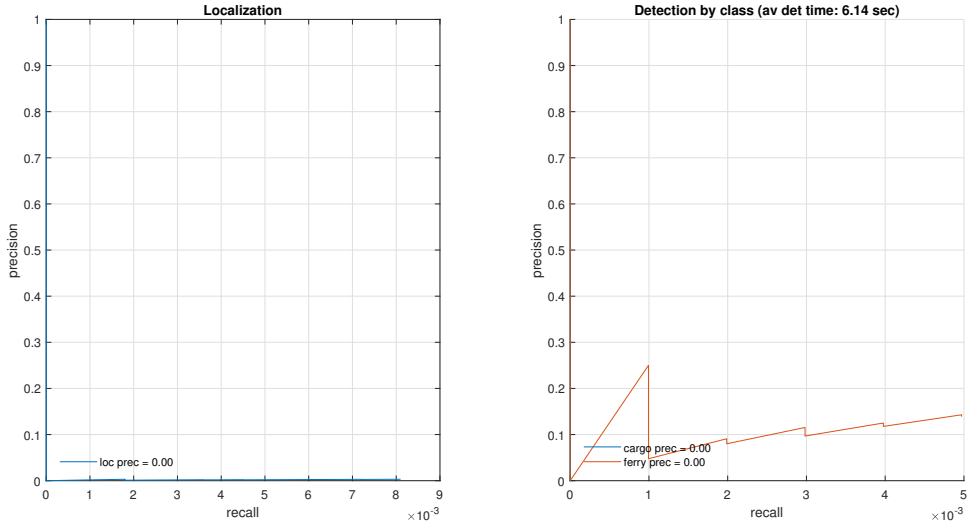


Figure 5.2: Precision-recall curve for ROI finder + CNN classifier (VGG16)

From the left curve at 5.2, it is clear that both precision and recall for classifier-based detectors are extremely low. That is in accordance with images (a) to (c) from figure 5.1: low values of precision are due to the misslocalization of detected objects, that results in a large number of false positives compared to just a few true positives; and low values of recall stem from the fact that in very few images the GT boxes are actually matched by bounding boxes created by the detector. Very similar terms can be used to comment detection results (right curve on figure 5.2). This results in AP and mAP values of 0.00. The most plausible explanation for this poor results is that the complexity in color and texture of the sky made the ROI finder find more objects (that is, elements that stand out agains their background) in clouds that in the sea.

Curves at figure 5.3 show detection results for the Faster RCNN detector employing AlexNet. Note that the curves are almost identical. This is the case because there is one class over represented in the dataset (ferries). It also shows that the classification task

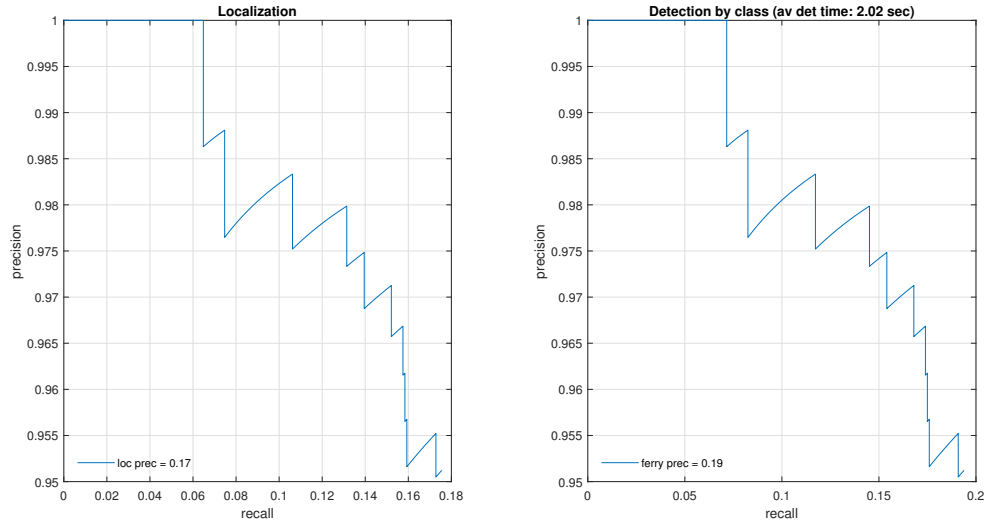


Figure 5.3: Precision-recall curve for Faster RCNN detector (AlexNet)

works very well (otherwise, detection results should be worse than localization ones). Because of that similarity, the following comments apply to both curves. Precision is very high (over 95% of all detected objects are true positives), and classification scores are distributed nicely through the curve (flat region for low recalls at precision 1 and a relatively sharp fall of precision for higher recalls), which means that high scores correspond, in general, to true positives. Recall values, however, are lower than 0.2, so less than one out of every five GT objects is correctly detected, which is a low performance. Overall, AP values of 0.17 and 0.19 for localization and detection, respectively, show that, although precision is very high, recall values remain low.

The curves at figure 5.4 show precision-recall results for the application of the Faster RCNN detector using VGG16 to Helsingør color images. The curves for localization (left) and ferry detection (right orange) are very similar. Again, that is due to the over representation of the ferry class in the dataset. For both curves, precision values are high (around 80% of all objects are correctly located and detected), with detection values slightly higher (this suggests a very high classification accuracy). As for recall, values are medium-low, with approximately one out of every three GT objects being localized and detected (altough, for detection, that is only the case for the ferry class, since on GT object of the cargo ship class is detected). Overall, localization AP has a moderately good value of 0.27, with AP for the ferry class presenting a value of 0.30.

Table 5.1 contains AP, mAP and aDP values for every detection method applied to the Helsingør image dataset.

At table 5.1 the gap in performance between Faster RCNN detectors and the rest is

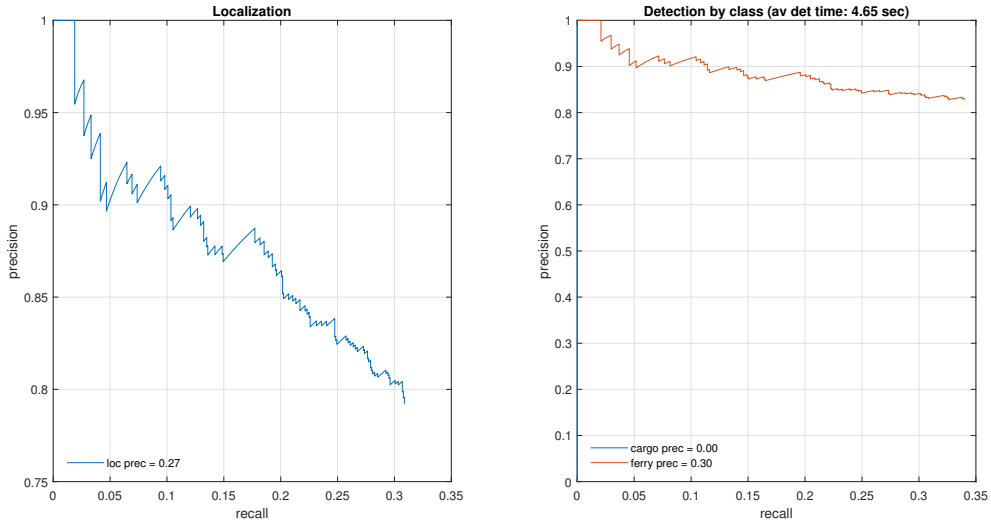


Figure 5.4: Precision-recall curve for Faster RCNN detector (VGG16)

	ROI + SVM	ROI + CNN (AlexNet)	ROI + CNN (VGG16)	ACF	Faster RCNN (AlexNet)	Faster RCNN (VGG16)
Localization AP	0.00	0.00	0.00	0.00	0.17	0.27
mAP	0.00	0.00	0.00	0.00	0.10	0.15
aDP (seconds)	4.10	4.40	6.14	5.77	2.02	4.65

Table 5.1: Evaluation results for Helsingør color images

clear. Although localizaiton AP and mAP values for Faster RCNN detectors are not very high, their lowest value is 0.1, with the localization AP for the VGG16 neural network reaching 0.27. In contrast, all localization AP and mAP values for the remaining methods are negligible. The better performance of the Faster RCNN detector using VGG16 when compared to the one using AlexNet comes at a cost: its detection time more that doubles that of AlexNet's detector on average.

5.2.2 Helsingør Monochrome Images

Helsingør monochrome images were taken simultaneously to color ones, so also mainly ferries are present on them. For most images, ferries are at a far to moderate distance. This image set does not contain objects of other classes.

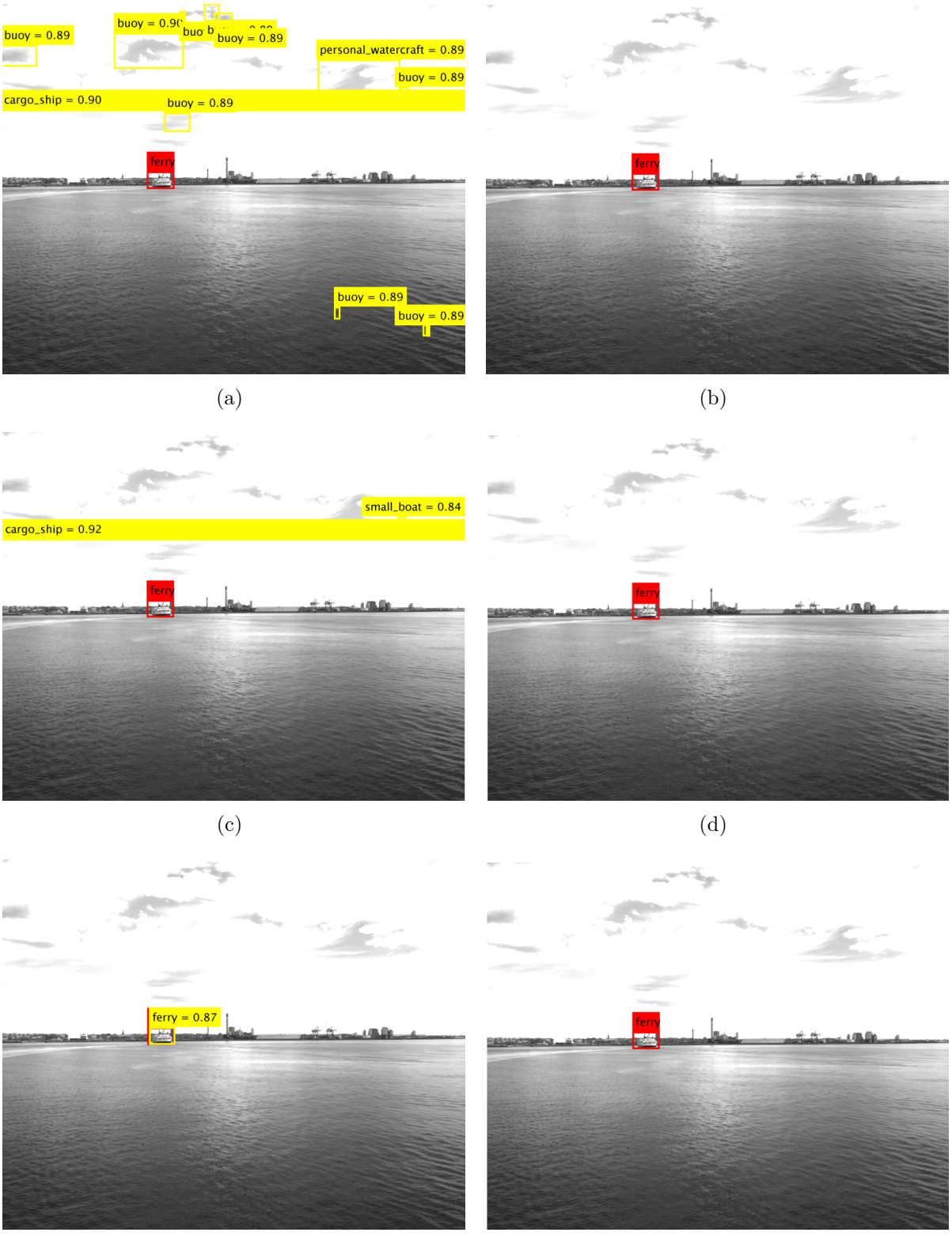


Figure 5.5: Object detection on Helsingor sample monochrome image: (a) ROI + SVM; (b) ROI + CNN (Alexnet); (c) ROI + CNN (VGG16); (d) ACF; (e) Faster RCNN (AlexNet); (e) Faster RCNN (VGG16)

Images at figure 5.5 show the GT and detected objects for a sample image from the Helsingør monochrome image set for all detectors. There is a ferry in the image, at a relatively far distance. Classifier-based detector either detect most objects in the sky (with high classification scores), as was the case for images from the Helsingør color dataset, or don't detect any object. The ACF detector doesn't detect any object. From the Faster RCNN detectors, the one making use of AlexNet manages to properly detect the ferry in the image, not producing any false detection. The one using VGG16 does not produce any detection at all.

None of the classifier-based detectors are able to properly detect a single object among all the GT objects present in the dataset. For that reason, their precision-recall curves are not presented. As for pure detectors, their precision-recall curves are quite different from one another, so all of them are shown.

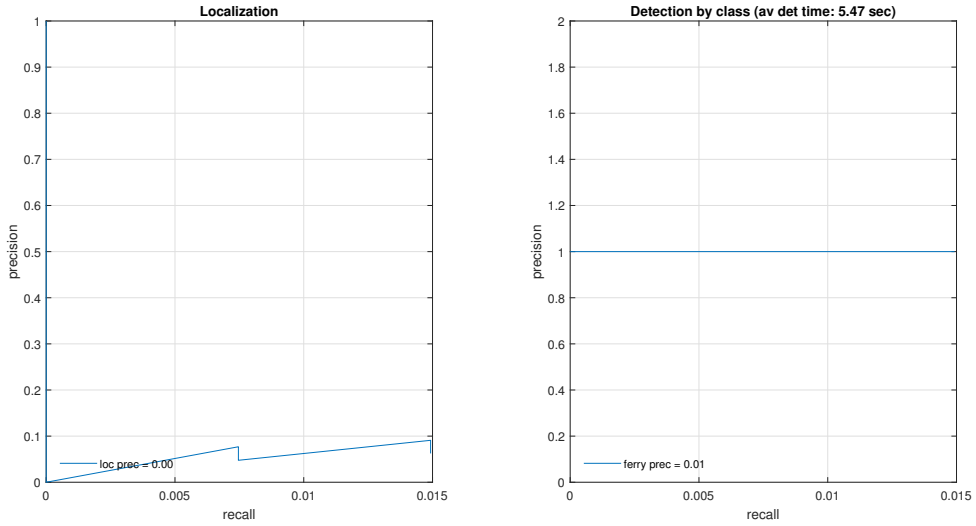


Figure 5.6: Precision-recall curve for ACF detector

Curves in figure 5.6 shown localization and detection performance for the ACF detector. Apart from the vey low value in recall (just 1.5% of all GT objects are detected), it should be noted that the detection curve (right) is wrong. For one class detection, localization and detection curves should be identical (if classification accuracy is perfect) or almost identical. Taking into account the performance of the ACF detector, the precision-recall curve for localization seems reasonable, but the one for detection, with perfect precision, does not. The student ignores what exactly went wrong when obtaining the curves, but it is obvious that the detection results are not correct. The student would like to point out that, for all the rest of the results obtained, there is the expected correspondence between localization and detection curves, so no other results suffer from this mismatch. Assuming that the localization curve for ACF detector was

correctly obtained (left curve on figure 5.6), both precision and recall are very low, with a negligible AP.

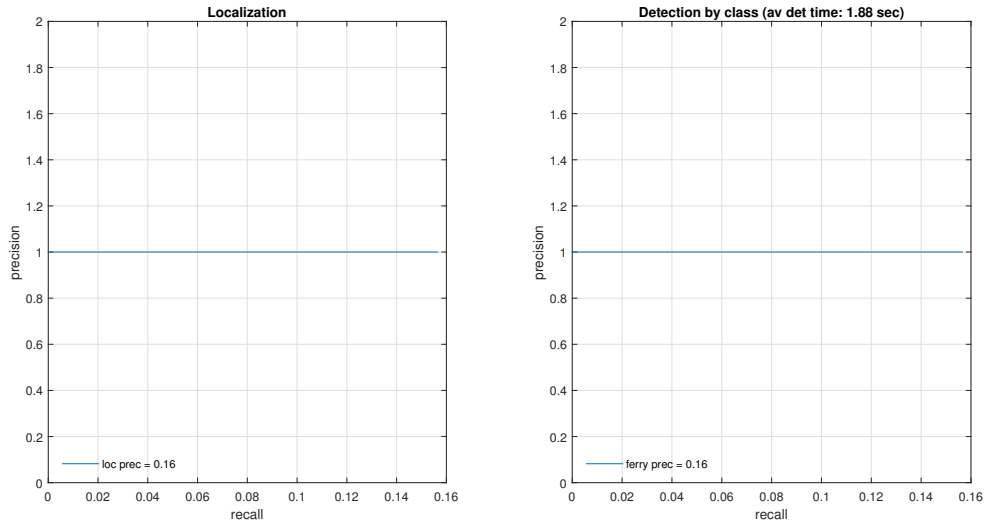


Figure 5.7: Precision-recall curve for Faster RCNN detector (AlexNet)

Curves in figure 5.7 show results for the Faster RCNN detector employing AlexNet. As already stated, since there is only one class in the dataset and classification accuracy is very high for Faster RCNN detectors, both curves are identical. Precision has a value of 1, which means that not a single false positive was produced for any image of the dataset. Recall, however, shows that only 15% of all GT objects were detected, which is a low value. If precision and recall are both considered, it is clear why AP and mAP values are both 0.15 (AP computation expression in equation 4.27).

Figure 5.8 contains the localization and detection curves for the Faster RCNN detector that uses VGG16. Due to only one class being present in the GT data and a very high classification accuracy, both curves are identical and are therefore analyzed together. Precision is quite high, with 88% of all the detected objects corresponding to true positives, but is still far from the perfect accuracy of AlexNet. Recall, on the other hand, has a moderately good value, since more than 40% of all GT objects are detected. This is considerably better than for AlexNet. Overall, AP values are 0.38, that are the best AP values among all the results for any dataset.

AP, mAP and aDP values for all the detection methods applied to the Hundested monochrome image set are gathered in table 5.2 (mAP for the ACF detector is 0.00 and not 0.01, since the corresponding detection curve is not correct).

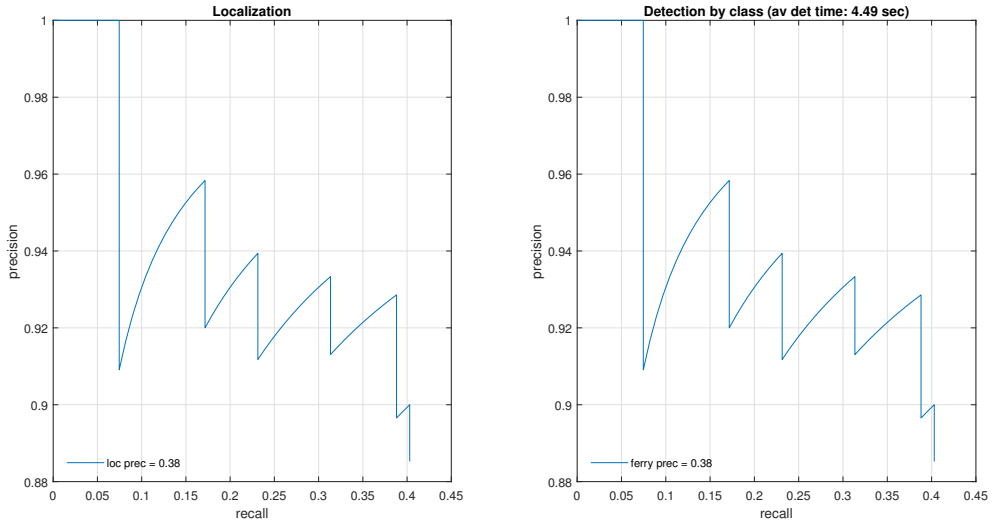


Figure 5.8: Precision-recall curve for Faster RCNN detector (VGG16)

	ROI + SVM	ROI + CNN (AlexNet)	ROI + CNN (VGG16)	ACF	Faster RCNN (AlexNet)	Faster RCNN (VGG16)
Localization AP	0.00	0.00	0.00	0.00	0.16	0.38
mAP	0.00	0.00	0.00	0.00	0.16	0.38
aDP (seconds)	2.82	3.24	4.95	5.47	1.88	4.49

Table 5.2: Evaluation results for Helsingør monochrome images

Table 5.2 endorses the much higher quality of Faster RCNN detectors when compared to the rest. In particular, the Faster RCNN detector using VGG16 yields considerably good AP values, although its detection time more than doubles that of AlexNet.

5.3 Summary

Chapter 5 presented the experiment conditions (namely, the image sets) where detection by all the implemented detectors would take place. Then, it presented and discussed the results obtained from every combination of detector and image set. It was clear from those results that Faster RCNN detectors greatly outperform the other detectors, achieving in one case (Faster RCNN detector with VGG16 applied to Helsingør monochrome images) a relatively good performance from the application point of view.

CHAPTER 6

Conclusion

This chapter closes the document. First, a summary of the content of the document is given. Then, the most relevant findings are highlighted. Finally, potential areas of improvement are suggested.

6.1 Overview

The topic of the thesis, that is object detection on digital images, is first put into context within the larger project that this thesis is part of in chapter 3. That is done in an increasingly specific sequence of systems, from the general autonomous vessel to the sensor fusion module. Then, the object detection system itself is treated, in chapter 4. Since that chapter covers the topic of the thesis, it is also the most extensive and detailed one. It first introduces the theoretical foundations for all the detection methods, including the basics of digital images, image processing and convolutional neural networks. It then covers the four detection methods developed, that includes two classifier-based detectors: ROI finder + SVM classifier and ROI finder + CNN classifier, and two pure detectors: ACF detector and Faster RCNN detector, including implementation aspects in MatLab. The chapter finishes by presenting the evaluation of the four methods, giving first a description of the evaluation methodology and then applying it in order to obtain and compare results that measure each detector performance. In chapter 5, methodology and results for the experiment conducted by applying the different detectors to images taken by the cameras that are part of the equipment of the project are presented. The methodology and the format of the results are the same as those for the evaluation of chapter 4.

6.2 Findings

Some conclusions can be draw from the results obtained, regarding the possibility of implementing an actual detector from the methods studied. Classifier-based methods, that make use of a ROI finder, are only suitable for situations where there is a neat line separating sea and sky and where the background is relatively simple (not clouds in the sky or waves in the sea), which discards its use for practical applications. The ACF detector seems to not be well suited for object detection at sea, since it very often mistakes waves or other features of the background for objects. Faster RCNN

detectors have consistently proved to perform much better than the previously mentioned detectors, providing very high values of precision with high classification scores. Their performance when it comes to recall, however, is moderately good at best. For this type of detector, the use of VGG16 as neural network model produced the best results, while also increasing detection time when compared to the use of AlexNet.

6.3 Future Work

Should the work developed in this thesis be expanded, it is suggested that only Faster RCNN detectors are considered, given the poor performance of any other detector implemented. The Faster RCNN detectors implemented in this project have room for improvement, especially when it comes to increase the number of GT objects detected. Although it might be argued that it is not vital to detect objects that are far away from the vessel (this could be reflected on an adaptive recall measurement, where the weight of the GT object on the total recall value depended on its distance to the vessel), it is still desirable to increase the distance and robustness of object detection as much as possible. For that purpose, it is proposed to train detectors with more images, especially from far away and blurred objects. That might enhance detection of distant objects, since the detectors implemented in the thesis were trained with images rich in features and they therefore fail to detect objects with not many of them. As for precision, although it presents very high values, it can still be improved and make more robust by taking advantage of the fact that images where detection is to be performed in a real scenario are presented as a sequence, with small changes from one image to the next. That can be exploited by applying an optical flow method that allows to discard false positives.

Bibliography

[Arl18] Timothy C. Arlen. *Understanding the mAP Evaluation Metric for Object Detection*. March 2018. URL: https://medium.com/@timothycarlen/understanding-the-map-evaluation-metric-for-object-detection-a07fe6962cf3.

[Bla15] D. Hermann; R. Galeazzi; J.C. Andersen; M. Blanke. *Smart Sensor Based Obstacle Detection for High-Speed Unmanned Surface Vehicle*. International Federation of Automatic Control, 2015.

[Bra04] Gabriella Csurka; Christopher R. Dance; Lixin Fan; Jutta Willamowski; Cédric Bray. *Visual Categorization with Bags of Keypoints*. Xerox Research Centre Europe, 2004.

[Bro16] J. Brownlee. *Boosting and AdaBoost for Machine Learning*. April 2016. URL: https://machinelearningmastery.com/boosting-and-adaboost-for-machine-learning.

[Das17] Siddharth Das. *CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more ….* November 2017. URL: https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df.

[Low] Lowrance. *Broadband 3G Radar*. URL: http://ww2.lowrance.com/en-au/Products/Radar/Broadband-3G-Radar-en-au.aspx.

[Nun06] Gonçalo Monteiro; Cristiano Premebida; Paulo Peixoto; Urbano Nunes. *Tracking and Classification of Dynamic Obstacles Using Laser Range Finder and Vision*. University of Coimbra, 20106.

[Per14] Piotr Dollár; Ron Appel; Serge Belongie; Pietro Perona. *Fast Feature Pyramids for Object Detection*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2014.

[rad] radartutorial.eu. *Frequency-Modulated Continuous-Wave Radar (FMCW Radar)*. URL: http://www.radartutorial.eu/02.basics/Frequency%5C%20Modulated%5C%20Continuous%5C%20Wave%5C%20Radar.en.html.

[Ros16] Adrian Rosebrock. *Intersection over Union (IoU) for object detection*. November 2016. URL: https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/.

[Sca11] Roland Siegwart; Illah R. Nourbakhsh; Davide Scaramuzza. *Introduction to Autonomous Mobile Robots*. Second edition. Massachusetts Institute of Technology, 2011.

[Shn14] Afzal Godil; Roger Bostelman; Will Shackleford; Tsai Hong; Michael Shneier. *Performance Metrics for Evaluating Object and Human Detection and Tracking Systems*. U.S. Department of Commerce, 2014.

[Tan17] Espen Johansen Tangstad. *Visual Detection of Maritime Vessels*. Department of Engineering Cybernetics. Norwegian University of Science and Technology, 2017.

[Xse] Xsens. *MTi-G-710*. URL: `https://www.xsens.com/products/mti-g-710/`.

[Zan17] Yang Zhang; Qing-Zhong Li; Feng-Ni Zang. *Ship Detection for Visual Maritime Surveillance from non-Stationary Platforms*. Shandong Provincial Key Laboratory of Ocean Engineering. Ocean University of China, 2017.

DTU Electrical Engineering
Department of Electrical Engineering
Technical University of Denmark

Ørsteds Plads
Building 348
DK-2800 Kgs. Lyngby
Denmark
Tel: (+45) 45 25 38 00

www.elektro.dtu.dk