



# **Real-time vehicle detection using low-cost sensors**

by

**Konstantinos Kourantidis**

A Doctoral Thesis

Submitted in partial fulfilment of the requirements for the award of

Doctor of Philosophy of Loughborough University

May 2019

© Konstantinos Kourantidis (2019)

## **Abstract**

Improving road safety and reducing the number of accidents is one of the top priorities for the automotive industry. As human driving behaviour is one of the top causation factors of road accidents, research is working towards removing control from the human driver by automating functions and finally introducing a fully Autonomous Vehicle (AV). A Collision Avoidance System (CAS) is one of the key safety systems for an AV, as it ensures all potential threats ahead of the vehicle are identified and appropriate action is taken. This research focuses on the task of vehicle detection, which is the base of a CAS, and attempts to produce an effective vehicle detector based on the data coming from a low-cost monocular camera. Developing a robust CAS based on low-cost sensor is crucial to bringing the cost of safety systems down and in this way, increase their adoption rate by end users.

In this work, detectors are developed based on the two main approaches to vehicle detection using a monocular camera. The first is the traditional image processing approach where visual cues are utilised to generate potential vehicle locations and at a second stage, verify the existence of vehicles in an image. The second approach is based on a Convolutional Neural Network, a computationally expensive method that unifies the detection process in a single pipeline. The goal is to determine which method is more appropriate for real-time applications. Following the first approach, a vehicle detector based on the combination of HOG features and SVM classification is developed. The detector attempts to optimise performance by modifying the detection pipeline and improve run-time performance. For the CNN-based approach, six different network models are developed and trained end to end using collected data, each with a different network structure and parameters, in an attempt to determine which combination produces the best results.

The evaluation of the different vehicle detectors produced some interesting findings; the first approach did not manage to produce a working detector, while the CNN-based approach produced a high performing vehicle detector with an 85.87% average precision and a very low miss rate. The detector managed to perform well under different operational environments (motorway, urban and rural roads) and the results

were validated using an external dataset. Additional testing of the vehicle detector indicated it is suitable as a base for safety applications such as CAS, with a run time performance of 12FPS and potential for further improvements.

## **Acknowledgements**

This thesis would not have been possible without the contribution and continuous support of many people. Throughout the years, the people by my side guided and supported me and I would like, through a few words at least, thank them for their help.

Foremost, I would like to express my gratitude to my supervisor, Professor Mohammed Quddus for his guidance and support throughout the duration of my studies. Without him, continuously encouraging and motivating me, helping me improve myself, I would not have been able to climb this mountain.

Special acknowledgements to those who supervised and guided me, even for a short period of time. Dr Marianna Imprialou, supervisor and friend, always available to provide help, guidance and listen to my concerns. Professor Abigail Bristow and Dr. Lucy Budd, who even for a while, assisted me in this effort.

I am also thankful to the researchers in the Transport Studies Group, colleagues and friends alike, who assisted me in various times by helping me with my work and providing feedback when I needed it.

Special thanks to my partner, Vivi, who was always there for me. Her patience, love and support made this all possible and more enjoyable. I thank her for going through this when it was the hardest and for putting up with my quirks!

I am grateful to all my good friends, for their constant support and making life in Loughborough better. Dimitris, Elli, Vasilis, Thanos thank you all!

Last but not least, I have to thank my parents, Giannis and Katerina, as well as my sister Chrysa, for their help throughout the years. They encouraged me always, supported my choices and without them, I would not be here today. I cannot repay their love enough and I dedicate this thesis to them.

# Contents

Abstract.....	i
Acknowledgements.....	iii
Contents .....	iv
List of Figures.....	viii
List of Tables .....	xii
Abbreviations.....	xiii
1 Introduction.....	1
1.1 Background .....	1
1.2 Problem statement.....	4
1.3 Research importance .....	6
1.4 Aim and objectives.....	7
1.5 Thesis outline .....	7
2 Literature Review .....	9
2.1 Introduction .....	9
2.2 Initial vehicle detection (Hypothesis Generation).....	10
2.2.1 Motion-based approaches .....	10
2.2.2 Appearance-based approaches .....	11
2.3 Hypothesis Verification .....	17
2.3.1 Template matching .....	18

2.3.2	Object classifier methods.....	18
2.4	Image feature extraction.....	25
2.5	Histogram of Oriented Gradients (HOG) and vehicle detection.....	28
2.5.1	HOG feature extraction.....	28
2.5.2	HOG-based vehicle detection.....	36
2.6	Convolutional Neural Networks and vehicle detection.....	42
2.6.1	Architecture of a Convolutional Neural Network (CNN).....	42
2.6.2	CNN-based vehicle detection.....	51
2.7	Range estimation using a monocular camera.....	59
2.8	Knowledge gap.....	61
3	Methodology.....	64
3.1	Introduction.....	64
3.2	Vehicle detection I – Histogram of Oriented Gradients (HOG) and Support Vector Machine (SVM) classification.....	64
3.2.1	Hypothesis Generation (HG).....	66
3.2.2	Hypothesis Verification.....	83
3.3	Vehicle detection II – Convolutional Neural Network (CNN).....	85
3.4	Range estimation using perspective geometry.....	99
4	Data collection and pre-processing.....	101
4.1	Introduction.....	101
4.2	Data collection.....	101

4.3	HOG based detector training data .....	106
4.3.1	Training set .....	106
4.3.2	Validation set .....	108
4.4	CNN-based detector training data .....	108
4.5	Testing dataset.....	112
4.6	Limitations of the dataset .....	113
4.7	Summary .....	113
5	Results.....	115
5.1	Introduction .....	115
5.2	HOG-based detector and SVM classification results.....	116
5.3	CNN-based vehicle detector results .....	136
5.4	Summary of results .....	159
6	Application of the Detector and Discussion .....	161
6.1	Introduction .....	161
6.2	Discussion on vehicle detection models .....	161
6.3	Discussion on the processing time .....	164
6.4	Potential application of the detector.....	167
6.5	Discussion summary .....	173
7	Conclusions.....	175
7.1	Introduction .....	175
7.2	Achieving the aim and objectives .....	177

7.3	Contribution to knowledge.....	178
7.4	Study limitations .....	180
7.5	Extensions and suggestions for future research .....	182
	References.....	184



## List of Figures

Figure 2-1: Original image (a), shadow segmentation (b).....	12
Figure 2-2: Edge detection .....	15
Figure 2-3: Rear light detection .....	17
Figure 2-4: ANN (n inputs, 2 hidden layers, 1 output) .....	20
Figure 2-5: Example of hyperplane separating two feature classes .....	22
Figure 2-6: Sample image .....	26
Figure 2-7: Gabor features (5 scales/8 orientations) .....	27
Figure 2-8: Gabor output .....	27
Figure 2-9: HOG descriptor generation .....	29
Figure 2-10: HOG cell/block performance .....	30
Figure 2-11: Bin size effect on miss rate .....	32
Figure 2-12: Bin size effect on accuracy/computational time .....	32
Figure 2-13: Example of a 9-bin HOG histogram for a cell .....	33
Figure 2-14: Effect of normalisation schemes .....	34
Figure 2-15: Summary of steps for HOG extraction .....	36
Figure 2-16: Effect of bin size and signed/unsigned orientation .....	37
Figure 2-17: Vertical HOG (v-HOG) structure .....	38
Figure 2-18: Effect of bin size on detection rate .....	39
Figure 2-19: Regular NN (left) - CNN (right) .....	43

Figure 2-20: Hyperbolic tangent function .....	46
Figure 2-21: Sigmoid function.....	46
Figure 2-22: Rectified Linear Unit function .....	47
Figure 2-23: Example of max pooling operation.....	48
Figure 2-24: Pooling operation .....	49
Figure 2-25: R-CNN flowchart .....	52
Figure 2-26: Fast R-CNN flowchart .....	53
Figure 2-27: The RPN in Faster R-CNN .....	54
Figure 2-28: Faster R-CNN flowchart .....	54
Figure 2-29: R-FCN flowchart .....	55
Figure 2-30: YOLO detection system .....	57
Figure 2-31: SSD network architecture .....	57
Figure 3-1: Detection system.....	65
Figure 3-2: Sample image.....	67
Figure 3-3: Sample image with Sobel filter applied.....	67
Figure 3-4: Sample image with Canny filter applied.....	67
Figure 3-5: Proposed HG system.....	70
Figure 3-6: HOG visualisation.....	73
Figure 3-7: Individual rose plot .....	74
Figure 3-8: Cell with strong horizontal elements .....	75
Figure 3-9: Example of cell processing .....	76

Figure 3-10: 4 and 8-way connectivity .....	78
Figure 3-11: CC clustering output .....	79
Figure 3-12: Improved object separation output.....	80
Figure 3-13: Original bounding boxes.....	82
Figure 3-14: Bounding boxes after small object removal.....	82
Figure 3-15: Image feature extraction process .....	83
Figure 3-16: Example of successful vehicle detection .....	84
Figure 3-17: Summary of developed CNNs .....	87
Figure 3-18: Reference network architecture .....	88
Figure 4-1: Loughborough University instrumented vehicle .....	102
Figure 4-2: Locations of M1 motorway, Loughborough and Nottingham.....	103
Figure 4-3: Image dataset samples.....	104
Figure 4-4: Variables measured by the radar sensor .....	105
Figure 4-5: Examples of positive training images .....	107
Figure 4-6: Examples of negative training samples (a) .....	107
Figure 4-7: Examples of negative training images (b) .....	108
Figure 4-8: Example of data augmentation.....	111
Figure 5-1: Confusion matrix - SVM linear kernel 128x128 .....	119
Figure 5-2: ROC curve - SVM linear kernel 128x128 .....	119
Figure 5-3: TPR/FNR - SVM linear kernel 128x128 .....	120
Figure 5-4: Confusion matrix - SVM F.Gaussian kernel 128x128.....	121

Figure 5-5: ROC curve - SVM F.Gaussian kernel 128x128.....	122
Figure 5-6: TPR/FNR - SVM F.Gaussian kernel 128x128 .....	122
Figure 5-7: Confusion matrix - SVM linear kernel 256x256 .....	123
Figure 5-8: ROC curve - SVM linear kernel 256x256 .....	124
Figure 5-9: TPR/FNR - SVM linear kernel 256x256 .....	124
Figure 5-10: CNN vehicle search window .....	137
Figure 5-11: FP detections in testing image .....	140
Figure 5-12: Model 2 output image .....	141
Figure 5-13: Model 4 output image .....	143
Figure 5-14: Model 6 Precision/Recall curve .....	146
Figure 5-15: Model 6 output images.....	147
Figure 5-16: Precision/Recall curve for dash cam videos .....	157
Figure 5-17: Detector output for dash cam video - Sunny .....	158
Figure 5-18: Detector output for dash cam video – Urban.....	158
Figure 6-1: CNN detector search space .....	165
Figure 6-2: Reduced search space .....	166
Figure 6-3: Range estimation - Video 1.....	169
Figure 6-4: Range estimation - Video 2.....	170
Figure 6-5: Range estimation - Video 3.....	171
Figure 6-6: Range estimation - Video 4.....	172

## List of Tables

Table 1-1: Advantages and disadvantages of active and passive sensors.....	5
Table 2-1: Features and classifiers in the literature .....	40
Table 3-1: Vector file size comparison.....	72
Table 5-1: Linear SVM classifier (128x128) performance .....	118
Table 5-2: F.Gaussian SVM classifier (128x128) performance .....	120
Table 5-3: Linear SVM classifier (256x256) performance .....	123
Table 5-4: HOG-based detector performance.....	126
Table 5-5: AP performance for CNN models.....	139
Table 5-6: Precision by environment type.....	148
Table 6-1: Run-time performance 1.....	165
Table 6-2: Run-time performance 2.....	166
Table 6-3: Range estimation for camera and radar.....	168

## Abbreviations

ABS:	Anti-lock Braking System
ACC:	Adaptive Cruise Control
AV:	Autonomous Vehicle
ADAS:	Advanced Driver Assistance Systems
CAS:	Collision Avoidance System
LIDAR:	Light Detection and Ranging
CNN/ConvNN:	Convolutional Neural Network
HG:	Hypothesis Generation
HV:	Hypothesis Verification
HOG:	Histogram of Oriented Gradients
ROI:	Region of Interest
IR:	Infrared
ANN:	Artificial Neural Network
SVM:	Support Vector Machine
NN:	Neural Network
PCA:	Principal Component Analysis
SIFT:	Scale-Invariant Feature Transform
TP, FP, TN, FN:	True Positive, False Positive, True Negative, False Negative
RGB:	Red, Green, Blue (colour images)

ReLU:	Rectified Linear Unit
FC:	Fully Connected
R-CNN:	Regions with Convolutional Neural Network
R-FCN:	Region-based Fully Convolutional Neural Network
SSD:	Single Shot Detector
YOLO:	You Only Look Once
SS:	Selective Search
RPN:	Region Proposal Network
AP/ mAP:	Average Precision/ mean Average Precision
FPS:	Frames per second
TTC:	Time to Collision
AdaBoost:	Adaptive Boosting
CC:	Connected Components
NIR:	Near Infrared
FOV:	Field of View
ROC curve:	Receiver Operation Characteristic curve
FNR/FPR/TPR:	False Negative Rate/ False Positive Rate/ True Positive Rate
IoU:	Intersection over Union
PR curve:	Precision/Recall curve

# 1 Introduction

## 1.1 Background

The introduction of the motor vehicle in the late 19<sup>th</sup> century brought a revolution in human mobility and completely transformed human societies. Motorised transport became the predominant means of moving people and goods and it is estimated that by the end of 2016, the global vehicle population stood at 1.32 billion cars and trucks (Petit, 2019).

As with other technologies, this increase in mobility brought new challenges in safety, pollution and energy demand. Vehicles have become one of the leading causes of deaths around the world, with road traffic fatalities reaching 1.35 million in 2016 according to the World Health Organisation (World Health Organisation, 2018). Despite efforts to reduce the number of fatalities, the number remains unacceptably high, with traffic deaths now being the leading cause of death for children and young adults aged 5-29 years (World Health Organisation, 2018).

According to Haddon (1980), vehicle collisions are the outcome of vehicle, environmental and human factors and to tackle this problem, a systematic approach is required at each stage (pre-crash, crash, post-crash stages). In addition to other interventions such as educational campaigns, enforcement etc., technological solutions can contribute to reduce the number of accidents happening in the first place. This need, to enhance vehicle safety has driven the development of safety systems throughout the years. Initially, passive safety systems such as seatbelts (introduced in the 1960s), crush zone (1970s) and airbags (1980s) improved the crashworthiness of vehicles and reduced the passenger fatalities and injuries. Active systems such as ABS (introduced in the 1970s), traction control (1980s), brake assist (1990s) and ACC (Adaptive Cruise Control)/blind spot detection/ lane departure detection systems in the 2000s brought further improvements in road transport. Now, even more advanced systems such as autonomous (driverless) intelligent vehicles are developed and are expected to revolutionise vehicular safety over the next few years (Eskandarian, 2012).



An autonomous -or driverless- vehicle (AV) is a vehicle capable of fulfilling the need to transport people or goods with little or no human input. Also known as a robotic vehicle, it is designed to travel between destinations without a human operator. To qualify as fully autonomous, it must be able to navigate without human intervention to a predetermined destination over roads that have not been adapted for its use. To clearly define the different levels of autonomy in vehicles, the Society of Automobile Engineers (SAE) introduced the following classification (SAE International, 2016):

**Level 0 – No automation:** The driver controls all aspects of the dynamic driving task, even when enhanced by warning or intervention systems.

**Level 1 – Driver assistance:** Driving mode-specific execution by a driver assistance system of either steering or acceleration/deceleration using information about the environment. The driver maintains control and performs all other aspects of the driving task.

**Level 2 – Partial automation:** Driving mode-specific execution by one or more driver assistance systems of both steering and acceleration/deceleration using information about the driving environment. The driver maintains control and performs all remaining aspects of the driving task.

**Level 3 – Conditional automation:** Driving mode-specific performance by an automated driving system of all aspects of the driving task with the expectation that the human driver will respond appropriately to a request to intervene.

**Level 4 - High automation:** Driving mode-specific performance by an automated driving system of all aspects of the driving task, even if a human driver does not respond appropriately to a request to intervene.

**Level 5 – Full automation:** Full-time performance by an automated driving system of all aspects of the driving task under all roadway and environmental conditions that can be managed by a human driver.

Autonomy is envisaged as the solution to many of the problems caused by the large number of vehicles in the streets today. Autonomous vehicles are expected to bring improvements in areas such as road safety, congestion, environmental pollution and energy consumption (Eskandarian, 2012; Litman, 2019). More specifically, some of the benefits expected from the introduction of higher levels of autonomy are:

- **Improved traffic safety**, by an overall reduction in the number and severity of crashes, improved reliability and faster reaction times compared to human drivers
- **Improvements in the traffic flow**, leading to reduced congestion, higher speed limits and reduced travel time
- **Fuel efficiency** by optimising fuel consumption, reducing stop & go driving and emissions
- **Time savings**, leading to reduced travel time, time required to find a parking space and manoeuvre the vehicle into spot
- **Removal of constraints related to human driving impairment**, such as disabilities, fatigue or sleepiness while driving, drink/drug driving
- **Economic benefits**. Not considering a higher initial cost to manufacture an AV compared to traditional vehicles, accident-related costs are expected to drop, along with fuel, maintenance and insurance costs.

By introducing Advanced Driver Assistance Systems (ADAS) and automating vehicle functionality, the ill-effects of human driving behaviour that leads to accidents (recognition errors, decision and performance errors) can be limited (NHTSA, 2015).

ADAS refers to the vehicle functions that an intelligent vehicle provides either completely autonomously or assists the driver with during driving. ADAS includes but is not limited to, systems such as ACC, Automatic Parking, Blind spot monitoring, Collision Avoidance Systems (CAS), Lane Change Assistance, Pedestrian protection systems and others.

Active safety systems such as CAS are designed to reduce the probability of an accident (Mukhtar et al., 2015). The Collision Avoidance functionality involves

detecting obstacles on the road that threaten the operation of the vehicle, the safety of the passengers/cargo as well as the vehicles and pedestrians in the surrounding environment. The system can either warn the user of the imminent collision or take longitudinal and lateral control of the vehicle in order to avoid the collision.

## **1.2 Problem statement**

A robust and reliable detection system is a crucial element for CAS. Obstacle detection is achieved by processing the data provided by environmental sensors (such as radar, cameras, LIDARs etc.) using detection and classification algorithms.

Sensors in CAS can be classified in two main categories: active and passive. Active sensors emit signals into the surrounding environment and capture the reflection to identify obstacles/targets. Sensors of this type are radar systems (emitting radar waves) and LIDAR (Light Detection and Ranging)/laser systems that use infrared signals or laser beams. Sensors of this type are able to measure distance directly without requiring high computational resources, are able to detect objects in larger distances compared to optical sensors and finally, their performance is robust in foggy or rainy conditions and during night time. Their main drawbacks are the high cost compared to vision systems; their increased power requirements (since they emit signals) and that same-type sensors interfere with each other (Sivaraman and Trivedi, 2013; Mukhtar et al., 2015; Eskandarian, 2012).

The most common passive sensor used for detection is the optical system (monocular/stereo camera). Cameras are low-cost solutions that are easier to install and maintain, offer higher resolutions and provide descriptive information and are also free from the interference problems active sensors face. Vision-based detection depends highly on the quality of acquired image (with quality depending on lighting and weather conditions) and requires more computing power to process the images. The table below gives a brief comparison between active and passive sensor systems:

**Table 1-1: Advantages and disadvantages of active and passive sensors**

Type of sensor	Advantages	Disadvantages
Active sensors (radar, LIDAR, laser)	<ol style="list-style-type: none"> <li>1. Direct distance measurements</li> <li>2. Longer detection range compared to camera</li> <li>3. Robustness against environmental conditions (fog, rain), during night time and complex shadows</li> </ol>	<ol style="list-style-type: none"> <li>1. Higher cost compared to vision systems</li> <li>2. Lower spatial resolution</li> <li>3. Interference between sensors of the same type</li> </ol>
Passive sensors (camera)	<ol style="list-style-type: none"> <li>1. Higher resolution and increased Field of View</li> <li>2. Lower cost compared to active sensors</li> <li>3. Useful descriptive information can be extracted from images</li> </ol>	<ol style="list-style-type: none"> <li>1. Quality of acquired data dependent on lighting and weather conditions</li> <li>2. Increased computational resources required to process images</li> </ol>

Currently, AV development efforts from automotive and technology companies such as Google, Tesla, Mercedes-Benz (Ziegler et al., 2014) or universities (Urmson et al., 2008; Broggi et al., 2014; Berlin Team et al., 2007; Wille et al., 2010; Rauskolb et al., 2009) make use of multiple or high cost sensors to achieve their functionality.

The robustness of the systems comes from fusing data obtained from multiple sources and eliminating the errors associated with the sensor systems. CAS that use multiple sensors lead to systems that are more reliable than those using only a single sensor (Premebida et al., 2007; Kmiotek and Ruichek, 2008; Chavez-garcia and Aycard, 2015; Bertozzi et al., 2008; García et al., 2017). For vehicle detection, it is

possible to fuse data from both active and passive sensors. During the fusion process, either the various sensor systems perform detection of objects/obstacles at the same time and validate each other or the system is built around one main system while the other secondary sensors validate the results of the main system (Rodriguez F. et al., 2010; Garcia et al., 2012).

While this approach increases the overall system's robustness, makes it more reliable and manages to collect the maximum amount of information from the surrounding environment, there are two major drawbacks to this approach.

The first one is that, due to the sheer number of sensors used for fusion, the total cost of the system rises significantly. Active sensors (radar but especially LIDAR systems) are significantly more expensive than vision systems. Despite costs dropping year by year due to the inevitable mass production of sensor systems, the total cost of full sensor suite is prohibitive making the mass adoption of safety systems in commercial vehicles much more difficult. The second significant drawback is that different sensor systems require different approaches and algorithms, making the whole system more complex and computationally expensive. Since different sensors generate data of different types from the surrounding environment, it is necessary to process them separately in order to fuse the information. For example, the point cloud generated by a LIDAR sensor cannot be directly used in conjunction with the video stream of a camera. Instead, it needs to be converted into a usable form before any object detection/classification process takes place.

### **1.3 Research importance**

This thesis will attempt to develop a simple and robust vehicle detector, able to perform under different environmental conditions. The detection system will be based on a vision sensor (a monocular camera in this case), which will act as the sole sensor system.

In the race to produce the first fully Autonomous Vehicle, cost is often overlooked and ADAS systems are offered at a premium to the end user. Cost however, is an important factor for the adoption of systems such as CAS in vehicles and therefore,

the goal should be to achieve the desired functionality with the lowest possible cost and at the same time, keep power and processing requirements low.

For that reason, it is essential to maximise the camera-based system's detection performance, producing the best possible results. Performance in this case will be measured by the system's ability to detect all vehicles on the road ahead, minimising any false detections and doing so in the most computationally efficient way.

This research project is meaningful, as the success in developing such a system will indicate that it is possible to achieve CA functionality in Intelligent and Autonomous Vehicles using low-cost sensors and pave the way for mass adoption of safety systems of this type in vehicles.

## **1.4 Aim and objectives**

The aim of this PhD research is to develop a reliable detector for identifying vehicles in real-time, based on a low-cost sensor, such as a monocular camera.

This aim will be fulfilled through the following objectives:

- To investigate existing vision-based approaches to vehicle detection
- To identify the methods that are more likely to produce the desired performance, given the limitations of the existing methods
- To collect, synthesise and process the data required to develop a vehicle detector
- To develop a method for detecting vehicles based on a low-cost monocular camera
- To assess and validate the performance of the developed vehicle detector

## **1.5 Thesis outline**

This thesis is organised in seven chapters. This section provides an outline of each chapter:

Chapter 2 conducts an extensive and critical literature review of vision-based vehicle detection. The review explores both traditional image processing-based vehicle detection as well as Convolutional Neural Network (CNN) based detection. Every stage of the two approaches is reviewed and the main findings of this review are discussed.

Chapter 3 presents the methodology of this thesis. Two vehicle detectors are developed, one following the traditional image processing approach (where the image is processed into a form where useful information can be extracted) and the other following CNN-based vehicle detection. Both attempt to optimise existing methods and improve detection performance. The approach followed to estimate distance to a moving vehicle ahead is also presented in this chapter.

Chapter 4 presents the data which are employed to build the models. The type of data used is presented along with a description of the collection process. The pre-processing of the dataset and its limitations are also discussed in this chapter.

Chapter 5 reveals the results for all the examined models. The developed detectors are evaluated based on established evaluation metrics and the best-performing detector is identified.

Chapter 6 discusses the findings of the Results chapter (Chapter 5). It also explores the run-time performance of the highest performing detector and determines whether it is suitable for real-time application. A simple range measurement application is developed and serves as a test to determine whether the developed vehicle detector can be the foundation upon which a complete CAS is based.

Finally, Chapter 7 summarises the findings of this research, discusses whether the goals originally set out are achieved and the limitations of this work. This is followed by a discussion for future research and improvements.

## 2 Literature Review

### 2.1 Introduction

Active safety systems such as a Collision Avoidance System (CAS) are designed to reduce the probability of an accident (Mukhtar et al., 2015). The Collision Avoidance functionality involves detecting obstacles on the road that threaten the safe operation of the vehicle, the safety of the passengers/cargo as well as the vehicles and pedestrians in the surrounding environment. The system can either warn the user of the imminent collision or take longitudinal and lateral control of the vehicle in order to avoid the collision.

A robust and reliable obstacle detection system is a crucial element for CAS. Obstacle detection is achieved by processing the data provided by environmental sensors (such as radars, cameras, LIDARs etc.) using detection and classification algorithms. Current high-performance detection systems use multiple or high-cost sensors to achieve their functionality. This study aims to develop a vehicle detection system based around a single low-cost sensor, in this case a monocular camera.

The process of detection using a camera-based system consists of two stages: Hypothesis Generation (HG) and Hypothesis Verification (HV). This chapter discusses each of the vehicle detection systems that comprise the detection process. First, the Hypothesis Generation (HG) stage is introduced. This sub-system is responsible for generating the object candidate locations in an image (initial detection). The second stage presented, Hypothesis Verification (HV) is responsible for verifying the existence of an object in the image and classifying it as vehicle, pedestrian or other object (Sun, Miller, et al., 2002; Li and Guo, 2013; Kanjee and Carroll, 2015; Mukhtar et al., 2015).

The main methods used for each stage of the detection process are presented here, with the focus given on those used for the proposed detection systems. The remainder of the review is structured as follows: Section 2.2 presents the method used to generate potential vehicle locations (HG) while Section 2.3 focuses on the verification of those potential locations (HV). Section 2.4 explores the topic of



feature extraction from images while Section 2.5 focuses on the use of HOG features in vehicle detection. Section 2.6 presents the architecture of Convolutional Neural Networks and their use in vehicle detection and Section 2.7 is a short review of range estimation techniques. Finally, Section 2.8 summarises this review and identifies the gap in research.

## **2.2 Initial vehicle detection (Hypothesis Generation)**

This first stage of the detection process involves identifying potential vehicle locations in the captured images. Potential location or regions of interest (a ROI is a portion of an image on which an operation is performed; multiple ROIs can exist in an image) in images can be determined using two methods: motion-based techniques that analyse a sequence of image frames to detect moving objects based on their optical flows or appearance-based (also known as knowledge-based) techniques that analyse single image frames to find visual cues that indicate vehicle existence (Sun, Miller, et al., 2002; Khammari et al., 2005; Sivaraman and Trivedi, 2013; Mukhtar et al., 2015).

### **2.2.1 Motion-based approaches**

In motion-based approaches, optical flow fields for moving vehicles are calculated by matching feature points or specific pixels between consecutive image frames.

Vehicle corners is the tracked feature by Smith (1995), while Heisele and Ritter (1995) track the colour blobs of the vehicles. Jazayeri et al. (2011) track multiple low-level features such as corners, horizontal line segments and intensity peaks. To create the optical flow fields in motion-based approaches, it is necessary to track the selected features across several frames. Usually, a fixed number of frames are selected. In Jazayeri et al. (2011) the minimum tracking duration is 50 frames. Yanpeng et al. (2008) use optical flow optimisation to track overtaking vehicles. Their findings show that detection accuracy depends on the relative speed between host and overtaking vehicle, with vehicles overtaking the host vehicle with small relative speed (less than 10km/h) proving difficult to detect (with a 69.1% detection rate). Kuo et al. (2011) use an appearance-based method and motion flow on an

embedded system to detect and track vehicles on a highway. Preceding and overtaking vehicles are detected at a rate of 95.8% and 90.6% respectively, with large vehicles causing false negatives in the detection.

Motion-based monocular camera detection is less common than appearance-based methods as it requires the analysis of several frames to detect moving objects. Since monocular vision lacks direct depth measurements that are important for motion flow methods, it is difficult to use such methods without introducing significant inaccuracies. Instead, for a monocular camera-based detection, it is simpler to analyse single frames to find visual cues. For motion-based detection, using a stereo vision system is more appropriate, as it is more accurate in calculating distance to objects.

### **2.2.2 Appearance-based approaches**

In appearance-based approaches, specific characteristics of a vehicle or of its adjacent environment are sought by the image processing algorithm. In this way regions of interest are created and are further examined for the presence of vehicles. Usually, a vehicle detection system looks for a combination of features in an image, as one feature on its own is unreliable and not sufficient to detect a vehicle (Chan et al., 2012). The main features that appear in the literature are:

- i. Shadows
- ii. Edges
- iii. Corners
- iv. Symmetry
- v. Texture
- vi. Colour
- vii. Vehicle back lights

**i) Shadows** on a paved road hint to the existence of a vehicle on the road. Cheon et al. (2012) detect the boundaries of the road region by outlining the lowest homogenous region in the lower part of an image. Areas with colour intensities under a specific threshold are declared shadow regions, their edges are detected and in that way,

possible vehicle locations are declared. Li and Guo (2013) segment the shadows underneath vehicles using histogram analysis and combine the horizontal and vertical edge features of the shadows to generate the possible vehicle locations. (Baek et al., 2014; Yu et al., 2016; Di and He, 2016) follow a similar approach to exploit shadows generated under the vehicle for ROI generation and combine them with other visual cues to enhance detection performance.

Figure 2-1b is the output of shadow region detection process performed on a sample image (Figure 2-1a). The shadows underneath the vehicles are visible in white, separating them from the surrounding environment:



**Figure 2-1: Original image (a), shadow segmentation (b). Source: Li and Guo (2013)**

Shadow region detection does not come without limitations. It is greatly affected by the illumination conditions of the environment and by the shadows cast by nearby objects.

**ii) Detecting the edges** of a vehicle is one of the most common methods for generating a ROI. The reason is that the rear view of a vehicle usually forms a rectangular shape with horizontal and vertical edges and specific aspect ratios, ranging between 0.4 and 1.6, depending on the size of the vehicle (Teoh, 2011). This common characteristic allows for efficient vehicle detection, minimising the probability for missed detections due to an irregular shape.

Edge detection aims to identify points in a digital image where there are sharp changes in image brightness. Such changes in an image usually correspond to discontinuities in depth or surface orientation, changes in material properties or variations in local illumination. The boundaries (edges) of an object are examples of such change in image brightness.

The most common ways to perform edge detection is to use filters allowing the detection of object boundaries. Such filters (e.g. Sobel filter, Canny edge detector) are not only used for the particular application but are widely used in image processing to extract useful structural properties of objects.

The Sobel operator (or filter) is a discrete differentiation operator, used to compute an approximation of the gradient of the image intensity function. The operator makes use of 2 3X3 sized kernels (one for horizontal and one for vertical changes) which are convoluted with the input greyscale image to calculate approximations of the two derivatives (Sobel, 1990; Fisher et al., 2003a). The two kernels are:

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \text{ in the } x \text{ (horizontal) direction, and}$$

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \text{ in the } y \text{ (vertical) direction.}$$

After the convoluting the image with the kernels, the resulting components  $G_x, G_y$  can be used to compute the gradient magnitude and direction:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * A \quad (2.1)$$

$$G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A \quad (2.2)$$

Where \* here denotes the 2-dimensional convolution operation.

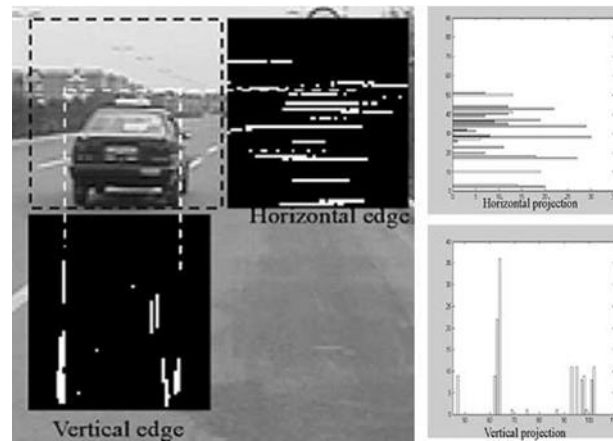
$$G = \sqrt{G_x^2 + G_y^2} \quad (2.3)$$

$$\theta = \tan^{-1}\left(\frac{G_y}{G_x}\right) \quad (2.4)$$

The Canny edge detection filter takes the Sobel operator and improves it to produce better results (Canny, 1986). The Canny edge filter is a multi-stage algorithm:

- i. First, the input image is smoothed using a 5X5 Gaussian filter to remove noise.
- ii. The second step is to find the intensity gradients of the smoothed image, using the Sobel operator described above.
- iii. After computing the gradient magnitudes and orientations, a process called non-maximum suppression is used to remove any unwanted pixels which may not be part of an edge. Every pixel is checked if it is a local maximum in its neighbourhood in the direction of the gradient. If so, it is stored for consideration in the next step, otherwise it is suppressed (its value is set to 0). This operation results in an image with “thin” edges.
- iv. This final stage decides which edges detected during the previous stages are really edges or not. During this stage, a process called hysteresis thresholding takes place. Two threshold values, one minimum and one maximum are used to detect true edges. Any edges with intensity gradient higher than the maximum threshold values are considered “sure” edges and are retained. Edges below the minimum value are labelled “false edges” and are discarded. The edges that have intensity gradients in between the threshold values are retained if they are connected to “sure” edges above the maximum threshold value, otherwise they are discarded as well. Apart from classifying edges, this stage also removes individual pixels (noise) assuming that edges are long lines.

Vertical and horizontal edge structures are detected and processed in Sun, Miller, et al.(2002) in order to generate ROIs. Teoh (2011) uses a Canny edge detector to generate an edge image, while Baek and Lee (2014) also use a Canny filter along with shadow region detection for the HG stage. Deng et al. (2014) use edges to detect vehicles in the same lane as the ego-vehicle, while (Yu et al., 2016; Di and He, 2016) perform ROI generation by utilising Sobel filters along with shadow region detection. Figure 2-2 below presents edge information on an image of a vehicle’s rear view:



**Figure 2-2: Edge detection** Source: Teoh (2011)

Edge detection is not robust enough to be used on its own. It is limited by interference from outlier edges while it is difficult to select an optimum threshold for the process itself (Mukhtar et al., 2015).

**iii)** In general, the shape of a vehicle is rectangular with four **corners**. Corners can be detected by identifying edge pixels at the positions corresponding to vehicle's sides. Detected corners can be clustered based on their type and location. These clusters can be used as inputs for a classifier (identifies objects) to determine whether the corners belong to a vehicle (Jinhui and Meng, 2010). Corner detection fails to perform in complex and cluttered environments (e.g. urban) and thus, is severely limited in its application.

**iv)** Most vehicles' front and rear views are symmetrical over a vertical centre line. With that in mind, the estimation of the location of a vehicle in an image by detecting regions with high horizontal symmetry is possible. A symmetry value is calculated based on pixel characteristics including grey colour values, gradients, colour and feature points. To locate a vehicle, it is necessary to determine the symmetry axis (centreline) of the vehicle, which can be found using grey level symmetry, contour and horizontal line symmetry (Kuehnle, 1991). Bensrhair et al. (2001) experiment with both monocular and stereo vision setups for vehicle detection and use symmetry as the main visual cue. Grey level symmetry is exploited initially, before symmetry properties are computed in horizontal and vertical edges are computed in order to enhance detection robustness. Dai et al. (2007) exploit vehicles' symmetric

properties in multi-scale windows for same lane vehicle detection (in highway environment), while Teoh (2011) use symmetry in a generated edge image for vehicle detection, after reducing the effective search space of the image to improve performance.

The drawback of Symmetry detection as a visual cue is that it is processor intensive and highly dependent on the vehicle's surrounding environment.

**v and vi) Colour and texture** of a vehicle. Most vehicles have a homogenous body colour that is different from road surface or a background object. The same applies for the texture of a vehicle, which is different from its surroundings. This information can be used to segment vehicles from the images acquired by a camera system. The limitation of using colour as a visual cue is its poor performance in a background of matching colours and its dependence on good illumination conditions.

**vii) Detection of head or rear lights** of a vehicle is usually performed to detect a vehicle in low-light conditions, where other feature detection techniques have low reliability. Chen et al. (2006) segment and cluster bright objects in an image, and the target regions are verified using symmetric properties (shape of lights, texture and relative position). Schamm et al. (2010) use a perspective blob filter to separate front from rear lights and in this way distinguish vehicles going the same or opposite way. Despite various night time detection techniques, such as the one described above, using an IR (infrared) camera is a more efficient way to detect vehicles and extract their features. Figure 2-3 below presents the result of rear light detection, where the lights are identified and stand out from the rest of the image using rectangular bounding boxes:



**Figure 2-3: Rear light detection** Source: Schamm et al. (2010)

Vehicle detection using vehicles' lights is inefficient when other sources of light such as street lights or shop bulbs exist in a scene.

This section presents the dominant visual features used for appearance-based vehicle detection and their limitations. The literature indicates that all of them are used with varying levels of effectiveness. The most common ones appear to be edge detection (which is easy to implement due to the existence of various filters from other image processing applications), shadow region detection (usually rectangular-shaped shadows are formed underneath vehicles) and symmetry (due to the specific rectangular shape of vehicles' rear view).

To increase robustness and ensure improved detection performance, a combination of visual cues is usually used to generate the sub-images necessary for the next part of the detection process.

### **2.3 Hypothesis Verification**

Hypothesis Verification (HV) is the second part of the detection process, tasked with validating the identified image areas as objects of interest (e.g. vehicle, pedestrian) or not.

There are two types of verification techniques:

- Template matching (correlation based approaches)



- Object classifier methods, probability distribution methods (learning based approaches)

### 2.3.1 Template matching

Template matching involves measuring the similarity between the ROI extracted from an image and a predefined template by calculating their correlation. Since vehicles come in different models with various appearances, a general template with features common to all vehicles is used. Common features may include rear window and plate, rectangular-shaped box with specific aspect ratio and an object with one horizontal and two vertical edges (U-shape) (Parodi and Piccioli, 1995; Handmann et al., 1998; Bensrhair et al., 2001).

A dynamic template that updates the initial template increases the robustness and of the matching technique. After the initial verification using a generic template described above, a cropped image of the detected vehicle becomes the new template and the updated template to accurately verify the vehicle across different frames (Betke et al., 1997). A dynamic template was also used by Lin and Xu, (2006) and its reliability was measured using edges, area and aspect ratio of the target to match the result to a vehicle, with a 95.7% performance rate reported for tracking.

While template matching has been used in vehicle detection, the method appears to have serious limitations that affect its usefulness. Detection failure when there are changes in scene illumination as well as problems when an object is rotated (resulting in low correlation coefficient) indicate limited robustness. Another problem to be considered is the need to generate a large number of templates to cover all vehicle cases on the road (Nath and Deb, 2010).

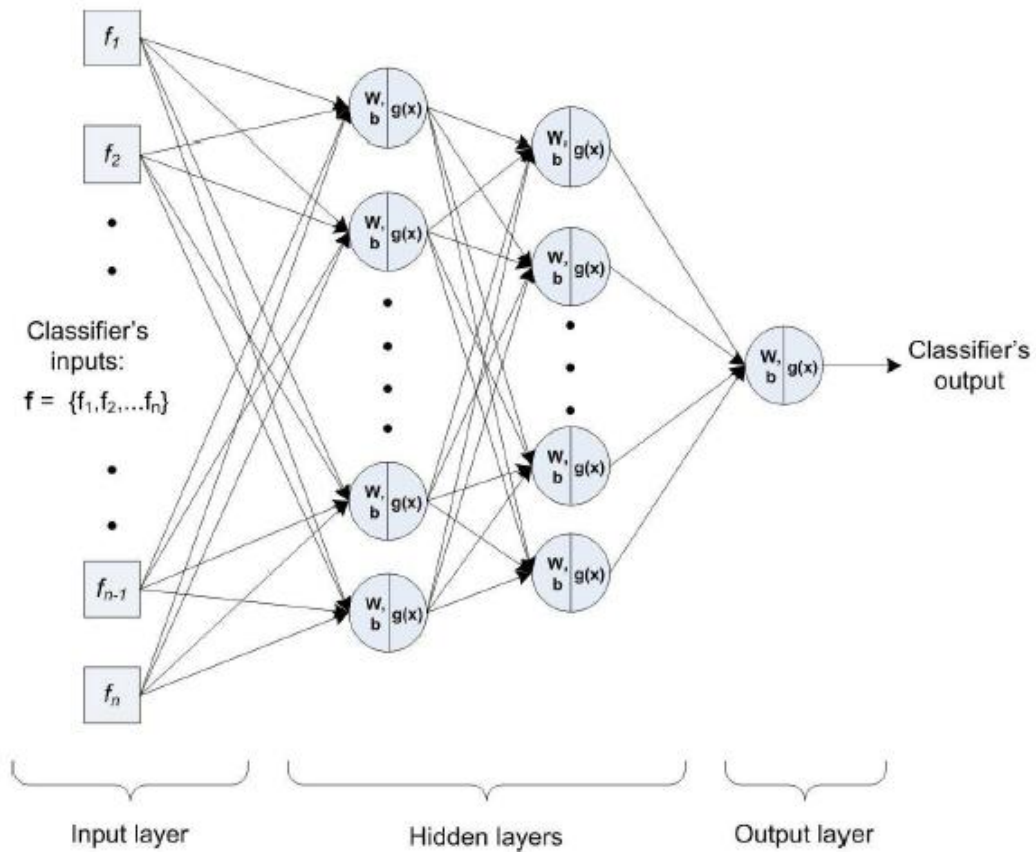
### 2.3.2 Object classifier methods

**Object classifiers** use two classes of images to discern vehicles from non-vehicle objects. A classifier learns the characteristics of a vehicle's appearance from a set of training images that includes images of both vehicles and non-vehicles. The classifier training is a supervised learning approach and the larger the set of images used, the better the classifier performs. The most common classifiers include **Artificial Neural**

**Networks (ANN), Support Vector Machines (SVM), AdaBoost and Mahalanobis distance.** Another way to classify objects is to model the probability distribution of features belonging in each class (methods using the Bayes rule assuming Gaussian distribution such as a Dynamic Bayesian Network) (Sun, Miller, et al., 2002; Khammari et al., 2005; Sun et al., 2006; ; Haselhoff et al., 2007; Sivaraman and Trivedi, 2013; Kanjee and Carroll, 2015; ; Mukhtar et al., 2015; Chavez-garcia and Aycard, 2015).

**i) Artificial Neural Networks** (Schmidhuber, 2015; Hornik et al., 1989) are a family of models inspired by biological neural networks and are used to estimate or approximate functions that can depend on a large number of inputs (generally unknown). They are presented as a series of interconnected nodes (neurons) which exchange information with each other. The connectors between the nodes have numeric weights assigned to them, which can be adjusted, making the network adaptive to inputs and capable of learning.

The basic structure of an ANN consists of one input layer, one output layer and one or more hidden layers between them. The number of inputs is equal to the number of features used for the classification and the number of outputs is the number of classes to be classified. For example, if there were 3 potential vehicle classes for objects to be classified into (e.g. car, truck, motorcycle), then the output layer would have 3 outputs. The hidden layers between the input and output layers is where the learning process is taking place. There is no specific rule to determine the number of hidden layers. A small number may result in inaccurate classification while a large one increases classification accuracy but increases computational load as well. In practice, the optimal number of hidden layers/nodes is determined through extensive experimentation. An example of an ANN with simple topology with  $n$  inputs (features), 1 output and 2 hidden layers (processing layers) is given in Figure 2-4:



**Figure 2-4: ANN (n inputs, 2 hidden layers, 1 output)** Source: Teoh (2011)

Each connector is associated with a weight,  $w$  and a bias,  $b$ . These values hold the ‘knowledge’ of the network and they are acquired through learning. In each node, the weighted sum of each input from the previous layer plus the bias term is calculated. The result is then transformed to the output using an activation function,  $g(x)$ . The learning process aims to minimise the output error. After an initial output and its error are calculated using the various inputs, the weights of the connectors are adjusted so that the classification error is reduced. This process is repeated for a fixed number of iterations or until the desired minimum error is achieved.

**ii) Support Vector Machines (SVMs)** (Vapnik, 1998; Müller et al., 2001) are supervised learning models with associated learning algorithms used to analyse and recognise patterns. They can be used to solve both classification and regression problems.

In supervised learning, models are taught what conclusions or predictions they should come up with. This is possible by providing the model with labelled prior knowledge (known output). The supervised learning model then uses the training data to “learn” a link between the input and outputs. By comparison, unsupervised learning models are left on their own to model the hidden structure or underlying structure in the data in order to learn more about the data. In this case, there is no labelled prior knowledge. A common example of unsupervised models is clustering methods.

The Support Vector Machine is a two-class classifier and its aim is to find the frontier which best segregates the two classes. It maps the training data of two object classes from the input space into a higher dimensional feature space, using a mapping function,  $\phi$ . Then an optimal separating hyperplane with maximum margin is constructed in the feature space to separate the two classes. After the optimal hyperplane is determined, new data samples are assigned into one category or the other. The coordinates of each data item are called Support Vectors. A simple example of classification using a linear SVM is given in Figure 2-5 below:

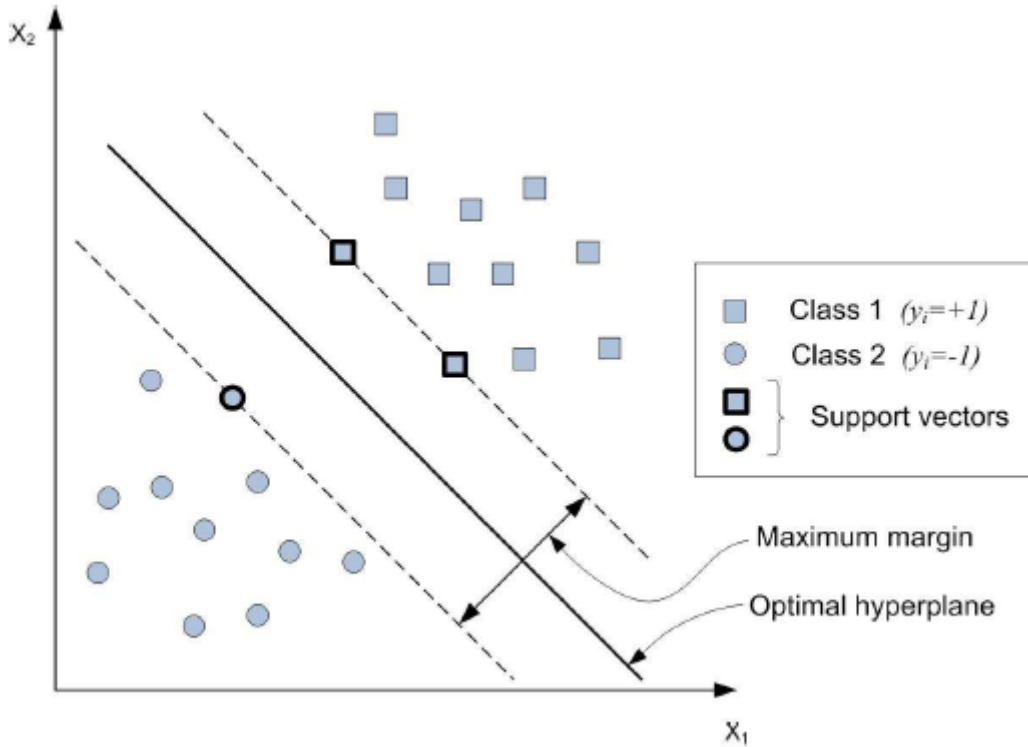


Figure 2-5: Example of hyperplane separating two feature classes Source: Teoh (2011)

Given a set of  $l$  labelled training samples (input – output pairs):

$$(x_i, y_i), i = 1, 2, \dots, l$$

Where:

$x_i \in R^N$  are the N-dimensional input feature vectors and  $y_i \in \{-1, +1\}$  are the labels for Class 1 and Class 2.

The decision function is:

$$f(x) = \sum_1^l y_i a_i k(x, x_i) + b \quad (2.5)$$

For an unknown data  $x$ , it can be classified into:

Class 1 if  $f(x) > 0$  or Class 2 if  $f(x) < 0$

The coefficients,  $a_i$  and bias,  $b$  are estimated from the training data, by solving the constrained optimisation problem with the aim of finding a separating hyperplane with maximum margin. The support vectors from Classes 1 and 2 are the training data that sit on the boundary in the hyperspace ( $a_i \neq 0$ ), as can be seen in Figure 2-5.

$k(x, x_i)$  is the kernel function that we use in order to avoid calculating the mapping function  $\phi$  which, in many cases, is not an easy task. The kernel function may be linear, polynomial, sigmoid etc. (Teoh, 2011; Chen et al., 2013;) :

Linear: 
$$K(x_i, x_j) = x_i^T x_j \quad (2.6)$$

(Gaussian) Radial Basis Function: 
$$K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2), \gamma > 0 \quad (2.7)$$

Polynomial: 
$$K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0 \quad (2.8)$$

Sigmoid: 
$$K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r) \quad (2.9)$$

Where:  $\gamma, r$  and  $d$  are the kernel parameters,  $T$  is the transposed matrix.

**iii) Mahalanobis distance** (Mahalanobis, 1936) is a measure that is used to assess the dissimilarity between two sets of variables. It is different than Euclidean distance in that it considers the correlation between the variables when calculating the distance. That is a useful characteristic because most of the variables used for classification are dependant to each another. It is used as a minimum distance classifier where the distances between an unknown sample and several object's classes are calculated. The sample is then classified into a class with the shortest distance.

The Mahalanobis distance,  $D_{Mahalanobis}$  between an N-dimensional vector  $x = (x_1, x_2, \dots, x_n)^T$  and a group of vectors with mean  $\mu = (\mu_1, \mu_2, \dots, \mu_n)^T$  and covariance matrix S is:

$$D_{Mahalanobis} = \sqrt{(x - \mu)^T S^{-1} (x - \mu)} \quad (2.10)$$

$\mu$  and  $S$  represent the vehicle's class distribution. In order to classify a test image, the  $D_{Mahalanobis}$  between the image and the  $\mu$  of each vehicle class is calculated. If the distance is below a certain threshold, the image is classified as a vehicle. By using varying thresholds, different values of true detection and false detection rates can be calculated.

**iv) AdaBoost** (Freund and Schapire, 1999) (short for Adaptive Boosting) is a boosting method used to improve the performance of several “weak” classifiers by combining them into a “strong” classifier. A weak classifier is a classifier that performs poorly, but still better than random guessing (over 50% correct classification). The output of these classifiers is combined into a weighted sum that represents the output of the final boosted classifier.

The equation for the boosted (strong) classifier is:

$$H(x) = \text{sign}(\sum_1^T a_t h_t(x)) \quad (2.11)$$

Where  $T$  is the number of weak classifiers,  $h_t(x)$  is the output of the weak classifier  $t$  and  $a_t$  the weight applied to classifier  $t$  by AdaBoost. The final output is a linear combination of all the weak classifiers and the classification decision is made by looking at the sign of this sum.

The output weight  $a_t$  for the first classifier is given by:  $a_t = \frac{1}{2} \ln \frac{(1-e_t)}{e_t}$  where  $e_t$  is the classifier's error rate. After computing the first alpha, the training example weights are calculated again using the following formula:  $D_{t+1}(i) = \frac{D_t(i) \exp(-a_t y_i h_t(x_i))}{Z_t}$  where  $D_t$  is a vector of weights, with one weight for each training example.  $D_t$  is a distribution which means that each weight  $D$  represents the probability that each training example  $i$  will be selected as part of the training set.  $Z_t$  is the sum of all weights and it is used in the division so that the weights are normalised and the probabilities all add up to 1 (Viola and Jones, 2001; Haselhoff et al., 2007; Matas and Sochman, 2009).

Classification accuracy depends on many factors so determining which classifier performs the best is not a trivial task. It is a combination of application requirements, balancing performance and processing speed and the quality of data.

For example, SVM classification performance depends on the selection of kernel function and its parameters, which are empirically determined, after experimentation and validation of produced results. A Neural Network's discrimination ability is largely dependent on the topology of the network, with the number of hidden layers and the number of training cycles affecting performance. Similarly, empirical evaluation determines the optimal number of hidden layers and training cycles. In the same way, AdaBoost performance is affected by the number of "weak" classifiers that make up the system (Sun et al., 2006; Teoh, 2011).

The quality and size of input data is equally important for successful classification results. Extracted images features affect the classifier's discriminative ability and the size of training data (positive and negative training samples) is also of significant importance.

Detailed information on the use and performance of classifiers in vehicle detection follows in section 2.5.2.

## **2.4 Image feature extraction**

Instead of feeding a classifier raw image data, an image processing stage is involved (feature extraction). The purpose of this processing is to remove irrelevant and redundant data, which would make the classification process harder and more computationally intensive. In order to obtain those specific information that will be used as input to the classifier, training images are processed to extract descriptive features of the object to be classified. A good selection of features is important to capture the variability in the appearance of a vehicle and achieve good classification results. The most common features used for classification are:

- Gabor features
- Principal Component Analysis (PCA) features



- Haar wavelets
- Histogram of Oriented Gradients (HOG)

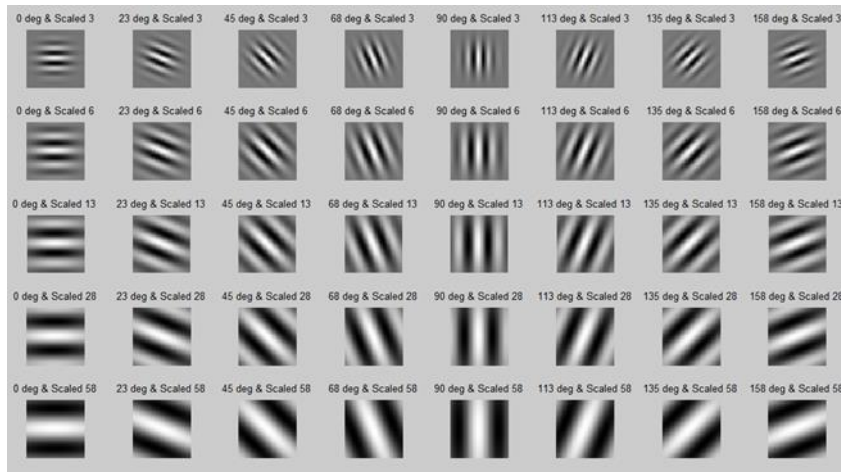
i) The **Gabor** filter is a linear filter used for edge detection in image processing. Representations of Gabor filters are reminiscent of the human visual system and they are used for texture analysis, object segmentation and classification. A 2D Gabor filter is a Gaussian function that can be viewed as a sinusoidal plane of particular frequency and orientation. Gabor filters respond to lines or edges with different widths and orientations depending on the filters' parameters, so in order to obtain good descriptive from an image, using different orientation and scales of the filter is required (Sun, Bebis, et al., 2002; Teoh, 2011).

An example of a Gabor filter (sample image (a), features -5 scales/8 orientations-(b) and output (c) can be seen in the figures below. Figure 2-6 is a sample image, Figure 2-7 is the features used (5 scales/8 orientations) and finally, Figure 2-8 is the output when the filters are applied. It can be observed that each of the filters produces a different output, highlighting the varying orientations:



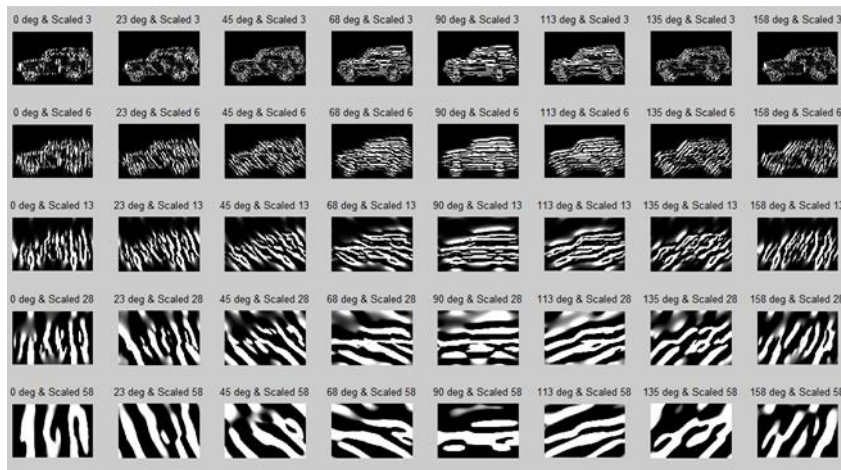
**Figure 2-6: Sample image**

**Source:** Stackoverflow (2016)



**Figure 2-7: Gabor features (5 scales/8 orientations)**

**Source: Stackoverflow (2016)**



**Figure 2-8: Gabor output**

**Source: Stackoverflow (2016)**

**ii) Principal Component Analysis (PCA)** (Jolliffe, 2002; Fodor, 2002) is a common technique used to reduce features' dimension. It is a statistical procedure that converts a set of observations of possibly correlated variables into uncorrelated values. At first, the covariance matrix  $C$  of the  $n$ -dimensional feature set is calculated. After that, all the eigenvectors and eigenvalues of the  $C$  matrix are calculated and sorted. The eigenvectors with the highest eigenvalues are called Principal Components. Each of the Principal Components consists of  $n$  coefficients and each coefficient is associated with a feature of the original feature space (Truong and Lee, 2009; Teoh, 2011).

**iii) Haar-like** features consider adjacent regions in an image, sum the pixel intensities in each region and calculate the difference between the sums of these

regions. The difference is used to categorise those sections of the image based on a threshold that separates objects from non-objects. Haar features are “weak” classifiers, so a large number are required to accurately describe an object (Viola and Jones, 2001).

**iv)** The **Histogram of Oriented Gradients (HOG)** (Dalal and Triggs, 2005) is a feature descriptor used in image processing with the purpose of identifying objects in an image. The primary idea behind HOG features is that they can be used to describe object appearance and shape by their distribution of intensity gradients or edge directions (in image processing, an edge is considered a point where image brightness changes abruptly) (Zhiqian Chen et al., 2013). More details on HOG and HOG based-vehicle detection are presented in the next section.

## **2.5 Histogram of Oriented Gradients (HOG) and vehicle detection**

### **2.5.1 HOG feature extraction**

HOG operates similarly to other descriptors such as edge orientation histograms (Freeman and Roth, 1995), SIFT (Scale-Invariant Feature Transform) descriptors (Lowe, 1999) and shape contexts (Belongie and Malik, 2000), the difference being that HOG describes a whole image and produces a single feature vector used to describe an object compared to other methods that operate locally around specific interest points and produce a collection of feature vectors to represent the same object. Compared to HOG, edge orientation histograms only take into consideration the gradient of pixels that correspond to edges, while SIFT and shape context descriptors measure shape similarity (by identifying interesting points in an object and measuring the relative position between them) (Lowe, 1999; Belongie and Malik, 2000).

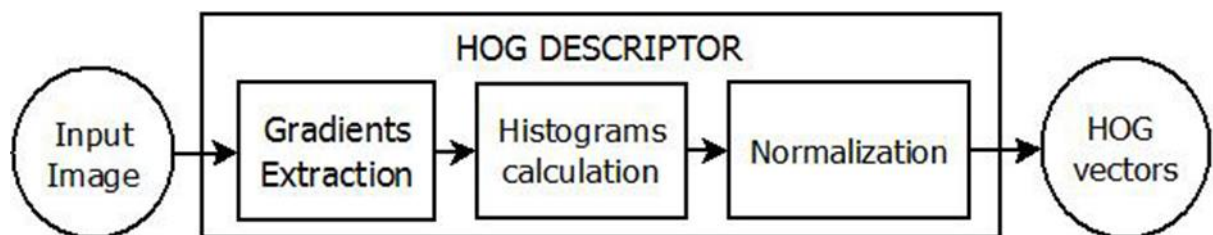
The descriptive power of HOG comes from calculating gradients over a dense grid of small image areas (cells) and contrast-normalising them in larger groups (blocks) (Dalal and Triggs, 2005).

**Cells** are small connected regions that contain the pixels that make up the image. Cells are rectangular in traditional HOG with a number of pixels that can be defined (e.g. 8X8, 16X16 pixels cells.). A **block** is the name given to the superset of cells upon which normalisation takes place.

HOG features were described and used for the first time in Dalal and Triggs (2005), in which they were used to detect pedestrians, outperforming Haar wavelets, SIFT descriptors and shape context descriptors.. Research with HOG features has since expanded to detecting other objects, including vehicles.

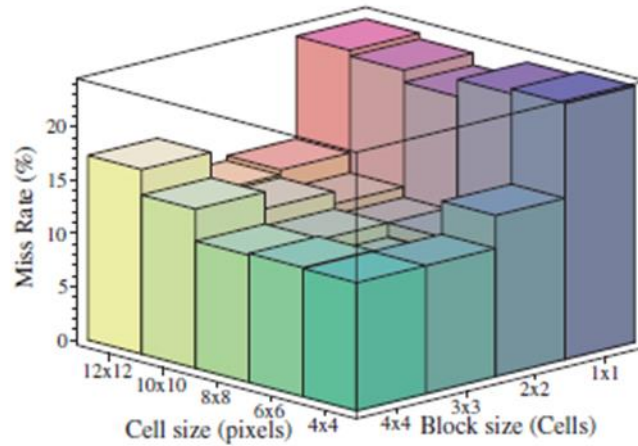
The general flow of calculating the HOG feature vector is the following (Figure 2-9):

1. Gradient computation for each pixel in a small area (cell)
2. Spatial/orientation binning
3. Normalisation and descriptor blocks



**Figure 2-9: HOG descriptor generation** Source: Ballesteros and Salgado (2014)

Since HOG is calculated over small image areas called cells, determining the cell size is the first step. Dalal and Triggs (2005) experimented with various cell sizes and concluded that there is a trade-off between detection accuracy and computational cost when deciding on cell and block size. The results of their experimentation are presented in Figure 2-10:



**Figure 2-10: HOG cell/block performance**

**Source:** Dalal and Triggs (2005)

They determined that rectangular 6x6 pixel cells, organised in 3x3 blocks perform best, with a misdetection rate of 10.4%. However, this was not the combination they used in their research, instead using 8x8 pixel cells in 2x2 blocks. This option was selected based on its performance and it is a close second in terms of minimum miss rate.

Gradient vectors are then computed for every pixel within each cell. In image processing, a **gradient** is a directional change in the intensity or colour in an image and is measured by the change in pixel values along each direction (x and y). Pixel value is a number that indicates the brightness of the pixel. For greyscale images, pixel value ranges between 0 and 255; this is the size of a byte (8 bits). 0 is the value representing black, while 255 represents white. Colour images have three separate components (RGB – Red, Green, and Blue) each component taking a value from 0-255.

The gradient is given by the formula:

$$\nabla f = \begin{bmatrix} f_x \\ f_y \end{bmatrix} \quad (2.12)$$

Where  $f_x$  and  $f_y$  are the derivatives with respect to x and y directions respectively:

$$f_x = I(x + 1, y) - I(x - 1, y) \quad (2.13)$$

$$f_y = I(x, y + 1) - I(x, y - 1) \quad (2.14)$$

It is necessary to compute the magnitude and the angle of the vector. Magnitude represents the strength of the edge:

$$|\nabla f(x, y)| = \sqrt{f_x^2 + f_y^2} \quad (2.15)$$

The gradient orientation  $\theta$  represents the direction of the edge for each pixel and is given by:

$$\theta(x, y) = \tan^{-1}\left(\frac{f_y}{f_x}\right) \quad (2.16)$$

The next step is the creation of the histograms for each cell. Each pixel calculates a weighted vote for an orientation-based histogram, based on the orientation of the gradient of that specific pixel. The cells can either be rectangular or radial in shape (rectangular cells offered better performance according to Dalal and Triggs (2005), and are better suited for vehicle detection due to the vehicle geometry) and the histogram bins are evenly spread over 0-180 degrees or 0-360 degrees, depending on whether “unsigned” or “signed” orientation has been selected. In the original paper, unsigned (0-180) orientation and 9 bins were found to perform the best for the specific task (human detection). Fine orientation is essential for good performance, but not at the expense of computational time. Performance improved by increasing the number of bins, but made little difference after 9 bins. Figures 2-11, 2-12 and 2-18 show the effect of bin size on accuracy and computational time from various researchers experimenting with HOG parameters, with miss rate decreasing as the number of bins increased. Computing time also increases significantly. (Dalal and Triggs, 2005; Tae Young Lee et al., 2015; Seung Hyun Lee et al., 2015). It is also argued that signed orientation might offer better performance in other tasks such as vehicle detection. However, other researchers have found that is not the case (Teoh, 2011).

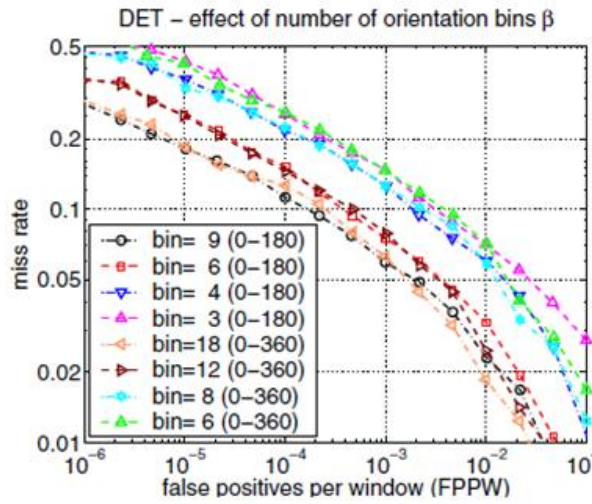


Figure 2-11: Bin size effect on miss rate Source: Dalal and Triggs (2005)

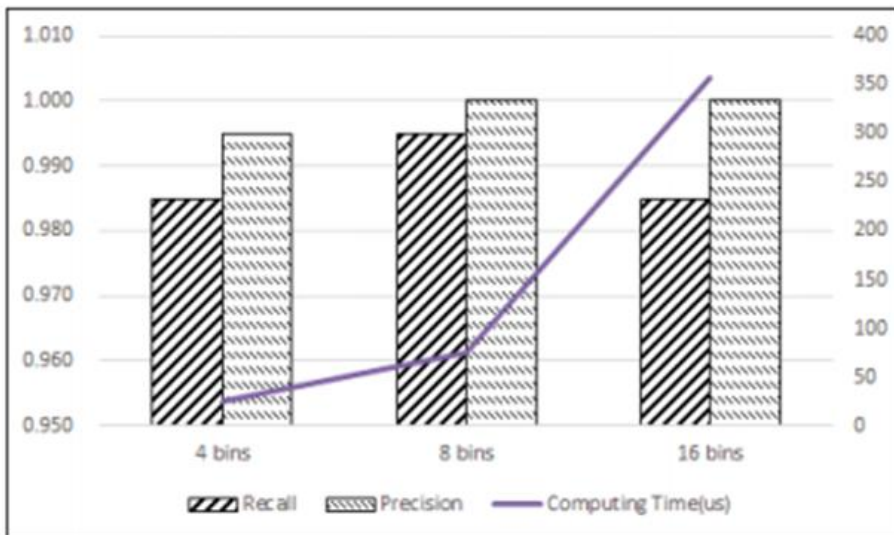
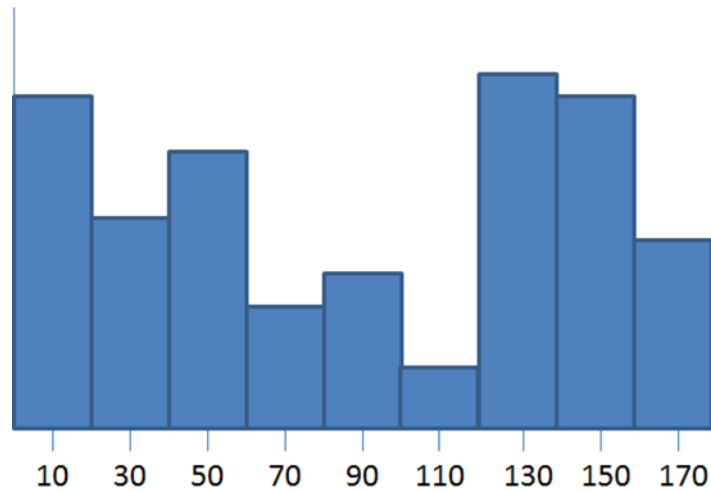


Figure 2-12: Bin size effect on accuracy/computational time Source: Lee et al. (2015a)



**Figure 2-13: Example of a 9-bin HOG histogram for a cell** Source: McCormick (2003)

This histogram doesn't actually represent the frequency distribution of the vectors' orientation. Instead, the magnitudes of pixels are added and binned.

A weighted vote means that the contribution of each pixel to the histogram via the gradient's magnitude is split between the two closest bins it falls in. That happens because it is very rare for a gradient to fall exactly at the centre of a bin (e.g. orientation angle 110 degrees). The vote is bilinearly interpolated between the centres of the two closest bins.

For example, if the angle of a gradient vector is 85 degrees, then we add 1/4th of its magnitude to the bin centred at 70 degrees, and 3/4ths of its magnitude to the bin centred at 90 degrees (distance to bin 70 is 15 degrees, distance to bin 90 is 5 degrees). It would be possible to change the way votes are cast using another method (Gaussian for example, to downweight the effect of pixels near the edges).

After this step, comes the normalisation of the histograms. Gradients are grouped in blocks and normalised locally to make them invariant to illumination changes (e.g. shadowing, contrast/brightness etc.). Dalal and Triggs (2005) evaluated 4 normalisation schemes:

- L2 norm:  $v \rightarrow \frac{v}{\sqrt{|v| + \epsilon^2}}$  (2.17)



- L2-Hys: L-2 norm followed by clipping (limiting the maximum values of  $v$  to 0.2) and renormalizing

- L1 norm:  $v \rightarrow \frac{v}{||v||+\epsilon}$  (2.18)

- L1-sqrt:  $v \rightarrow \sqrt{\frac{v}{||v||+\epsilon}}$  (2.19)

where  $v$  is the unnormalised descriptor vector,  $||v||$  its k-norm (magnitude) and  $\epsilon$  a small constant.

Results show that L2 norm, L2-Hys and L1-sqrt offer comparable performance while L1 norm is slightly worse by about 5%. All normalisation schemes offered better performance compared to non-normalised data. The results can be seen in the figure below, where it is obvious that miss rate and False Positives decrease when normalisation is used:

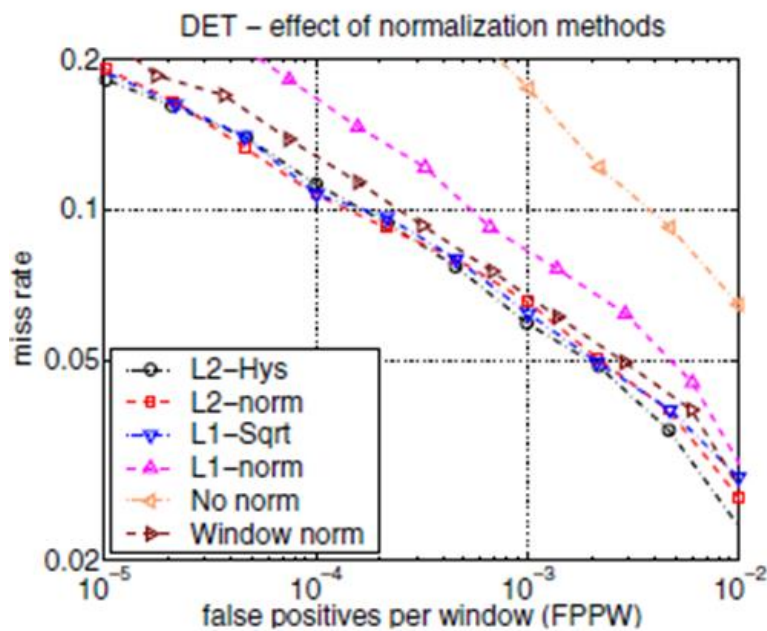


Figure 2-14: Effect of normalisation schemes Source: Dalal and Triggs (2005)

The key characteristic of the HOG descriptor is the block overlap in normalisation that creates redundancy and improves the detection performance. In their paper, Dalal and Triggs (2005) experimented with various levels of overlap before implementing a 50% overlap on the blocks. The effect of overlap is that each cell

appears multiple times in the final descriptor, though each time normalised by a different set of neighbouring cells. In 2x2 cell blocks with 50% overlap, corner cells appear once, other edge cells appear twice, while interior cells appear four times each. This leads to a final vector file (descriptor) that is much longer compared to a non-overlap descriptor. This leads to increased processing time as a side-effect of higher detection accuracy.

In the original implementation of HOG for human detection, a very dense detection window was created due to overlapping, and this redundancy led to improved performance by 5%. This dense window was beneficial for detecting the shape of humans, but may not be necessary for vehicles, where shape presents smaller variation (usually a rectangular shape). There was no comparison in performance between overlapping and non-overlapping HOG detection in the original study. Another study by Ma et al., (2011) (again, for pedestrian detection) showed that non-overlapping HOG had a 3% increased miss rate compared to overlapping HOG but around 40% increase in processing speed.

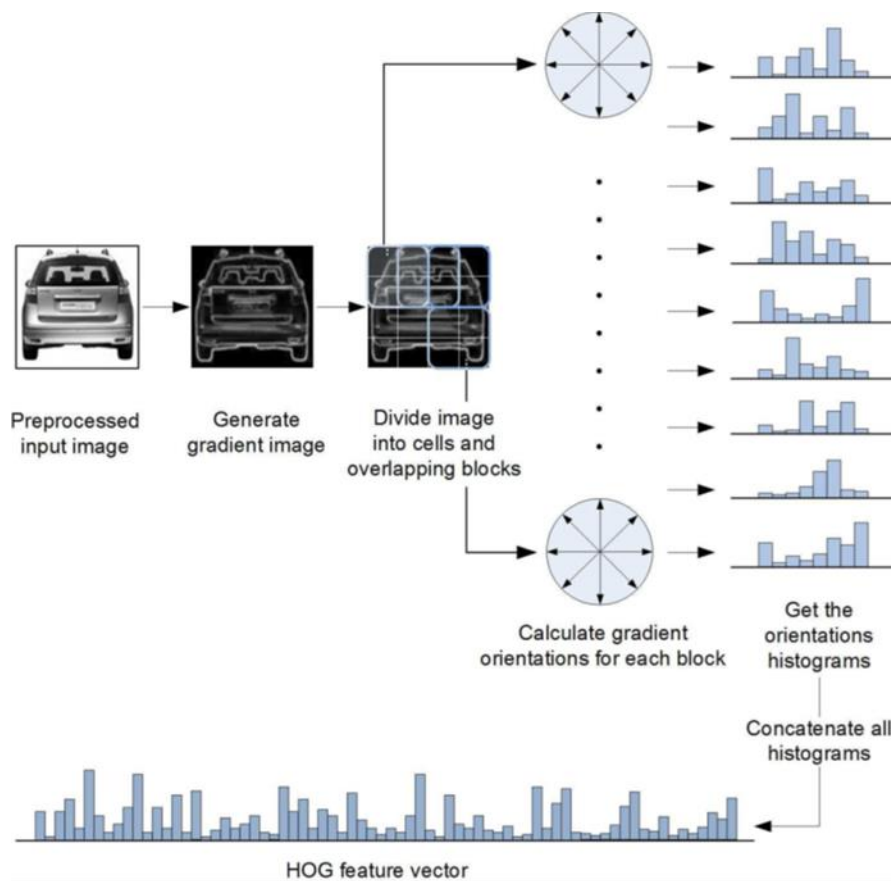
While it is not clear why block normalisation was chosen in the original research as opposed to normalising across the whole image, it is probably due to the fact that it is more probable for changes in contrast to occur over small regions within the image. So instead of normalising across the whole image, normalisation happens in a small region around the cell.

The final HOG descriptor is the concatenation of all normalised cell histograms from all blocks in the image in a vector. So, the final vector has a size:

Descriptor size = Total no. of blocks \* cells per block \* no. of bins per histogram.

So for example, a 32X32 pixel image produces a 324 long vector (9 bins, 9 blocks in total, 4 cells per block).

The 64x128 images used by Dalal and Triggs (2005), produce 3,780 values (7 blocks horizontally, 15 blocks vertically, 9 bins, 4 cells per block). Figure 2-15 below summarises and visualises the steps for HOG feature extraction:



**Figure 2-15: Summary of steps for HOG extraction**      **Source:** Teoh (2011)

### 2.5.2 HOG-based vehicle detection

HOG and its variations are widely used in image processing for object identification. While its original application was pedestrian detection, it has been used to detect other objects that share common characteristics with each other, such as road vehicles.

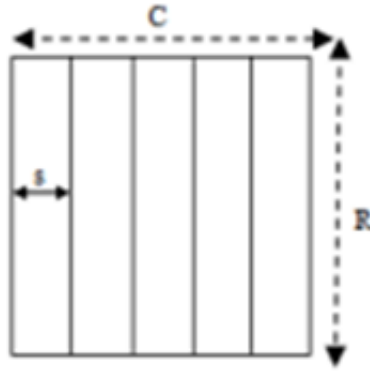
In his research on vehicle detection, Teoh (2011) experimented with HOG parameters (i.e. bin size, signed/unsigned orientation, gradient computation) and concluded that HOG outperforms Gabor features in detection accuracy as well as computational efficiency. The results of this extensive experimenting with HOG features can be seen in Figure 2-16 below:

Number of histogram bins	Detection rate	Area under the ROC curve	Processing time (ms)
4	86.1%	0.9875	13.35
8	90.8%	0.9921	22.61
16	96.7%	0.9969	61.02
32	97.1%	0.9973	276.22

Number of orientations	Combine opposite orientation?	Number of histogram bins	Detection rate	Area under the ROC curve	Processing time (ms)
8	No	8	90.8%	0.9921	22.61
8	Yes	4	92.3%	0.9928	12.92
16	No	16	96.7%	0.9969	61.02
16	Yes	8	97.8%	0.9980	22.78
32	No	32	97.1%	0.9973	276.22
32	Yes	16	98.1%	0.9982	59.83

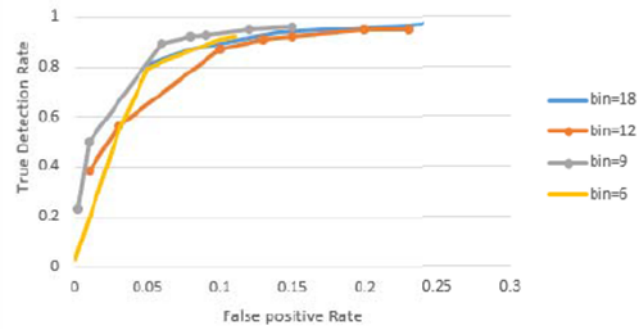
**Figure 2-16: Effect of bin size and signed/unsigned orientation** Source: Teoh (2011)

Li and Guo (2013) and Chen et al. (2013) use a combination of HOG coupled with a SVM classifier to detect vehicles. A shadow edge detection technique and a brute force approach are used respectively to achieve their results, with the first one claiming 96.87% detection rate on very low resolution images (24X24 pixels) without commenting on real-time performance, while the second claims real-time operation though without providing detection rates. Hu et al.(2013)use HOG features for verifying vehicle existence while utilising Haar wavelets and AdaBoost for ROI generation, achieving 93.3% classification accuracy. Yc Chen et al. (2013) use perspective geometry to solve the problem of slow sliding window approach and their adaptive scan approach results in 7 times faster vehicle search compared to the brute force approach. Similarly, Kim et al. (2013) use adaptive window but their overall system performance fails to meet real-time requirements. Laopracha et al. (2014) use a modified version of HOG called v-HOG, which is less accurate but faster than traditional HOG and experiment with HOG parameters and kernel functions. Their method results in up to 100% classification accuracy, though using only a set of low-resolution vehicle images (64x64 pixels). The structure of v-HOG can be seen in Figure 2-17 below:



**Figure 2-17: Vertical HOG (v-HOG) structure** Source: Laopracha et al. (2014)

Kim et al. (2014) use SVM and HOG combined with a method to determine the vehicle size, in order to reduce the size of ROIs for classification and the total computational time due to the sliding window approach. Shadow and edge detection are used by Baek and Lee (2014) to determine ROI and a HOG+SVM verification process. Their paper reports 15ms candidate generation and 60ms vehicle verification time. Deng et al. (2014) use lane detection to limit the search space and improve performance. Their system is only able to detect vehicles directly ahead of the ego-vehicle. Ballesteros and Salgado (2014) experiment with HOG parameters for a pose-based vehicle verification system. By removing interpolation and HOG overlap and using only specific cells in the descriptor, it is claimed that computational cost is reduced by 60% while retaining the same level of accuracy as traditional HOG. Alencar et al. (2015) and Chavez-garcia and Aycard (2015) use a multi-sensor fusion system (radar and radar/LIDAR respectively) to improve detection performance, though both systems exhibit false detections in many cases. Kim et al. (2015) use  $\pi$ HOG (a variation of traditional HOG) to include position and intensity information and increase its descriptive ability. The resulting feature vector is longer than HOG and reduced search space is required to reduce computational time. Lee et al. (2015) detect shadow region for the HG part of the process, reporting 92% detection rate but the system fails completely when the shadow area is not properly detected.



**Figure 2-18: Effect of bin size on detection rate** Source: Lee et al. (2015b)

Sun and Watada (2015) use boosted HOG+SVM classifier to detect pedestrians and vehicles in static images. HOG features are boosted using Adaboost and shadow detection is used for ROI generation. Vehicle detection accuracy is not reported, though pedestrian detection is reported lower than traditional HOG (75% to 86%). The detection process is also limited to daytime operation. Yu et al. (2016) detail a vision-based lane marking and vehicle detection system with shadows and vertical edges used for ROI generation. SVM+HOG are used for verification, with no quantitative results provided, though problems with the accuracy of the classifier and computation time are reported. Finally, Di (2016) use HOG and an Adaboost classifier, with edges and shadow providing ROIs. Fine direction separation is used (20 directions per histogram) for every cell. To reduce running time, feature values are grouped before they are used by AdaBoost’s “weak” classifiers (for example, 4 adjacent directions are grouped into 1 for the first “weak” classifier). Detection accuracy is 91.37% with a false positive rate of 3.09%, although the Adaboost classifier proves to be too computationally expensive for real-time applications.

### **Classifier performance using other feature descriptors**

The performance of various combinations of classifiers and feature sets has been examined in the literature before. In the majority of cases, performance is measured by the detection rate (DR) metric, which indicates the proportion of correct vehicle detections out of the total number of vehicle instances. The results of the most relevant classification methods are presented in Table 2-1 below:

**Table 2-1: Features and classifiers in the literature**

Methodology	Results	Image resolution / additional information
AdaBoost + Haar like features (Haselhoff <i>et al.</i> , 2007)	88.5-93% DR (detection rate)	24x18 resolution images
AdaBoost + Haar like features (Wu <i>et al.</i> , 2009)	71% DR 38 fps	30x25 / 40x40 images Detecting occluded vehicles
AdaBoost + HOG features (Chavez-garcia and Aycard, 2015)	90% DR	No info on resolution Performance varies depending on traffic scenario / (high FP)
Random Decision Forest + HOG features (Alencar <i>et al.</i> , 2015)	392 FP/430 FN (1922 cars)	128x64 images General detector
SVM classifier + Haar wavelets (Papageorgiou, 2000)	90% DR 28 fps	32x32 images Low refresh rate
SVM classifier + Haar wavelets (Sun, Miller, <i>et al.</i> , 2002)	10 Hz sampling rate	64x64 images
SVM classifier + Gabor features (Sun, Bebis, <i>et al.</i> , 2002)	94.8% DR 30 fps	64x64 images
NN + PCA features SVM + PCA features NN + Gabor features SVM + Gabor features NN + wavelet features SVM + wavelet features NN + combined Gabor/wavelets SVM + combined Gabor/wavelets (Sun <i>et al.</i> , 2006)	18% average error 9.09% error 16% average error 6% average error 16.4% error 8.5% average error 11.54% error 4% average error	32x32 images
SVM classifier + PCA analysis (Truong and Lee, 2009)	95% DR	64x64 images
SVM classifier + HOG features (Rybski <i>et al.</i> , 2010)	88% DR	50x60 images Orientation detection
Mahalanobis distance + Gabor features Mahalanobis distance + HOG features ANN + Gabor features ANN + HOG features SVM + Gabor features SVM + HOG features (Teoh, 2011)	65.2% DR 78.1% DR 89.4% DR 96.5% DR 96.3% DR 98.4% DR	32x32 images Gabor high processing time
SVM classifier + HOG features (Li and Guo, 2013)	96.87% DR	24x24 images
AdaBoost + Haar(HG)/HOG (HV) (Hu <i>et al.</i> , 2013)	93.3% DR	64x64 images

Methodology	Results	Image resolution / Additional information
SVM classifier + HOG features (Baek <i>et al.</i> , 2014)	98.75% DR	No resolution reported 15ms ROI generation 60ms verification time
SVM + adapted HOG features (Ballesteros and Salgado, 2014)	98.69% DR on average	64x64 resolution images Static images
SVM classifier + HOG features (Tae Young Lee <i>et al.</i> , 2015)	92.09% DR	64x64 images
AdaBoost + HOG features (Di and He, 2016)	91.37% DR 3.09% FP	64x64 images No real-time performance
ELM + HOG features ELM + v.HOG features SVM + HOG features SVM + v.HOG features (Laopracha <i>et al.</i> , 2014)	98.76% DR 98.90% DR 77-100% DR 96.87-100% DR	64x64 images Static images No processing time reported

A camera/radar fusion system to perform vehicle detection is described in Haselhoff et al. (2007), where a method to use mutual information from the two sensors and eliminate redundant features is used. The radar here is used to generate ROIs while the camera verifies the existence of obstacles by extracting rectangular Haar features and classifying using AdaBoost. Subimage resolution is normalised to 24x18 and contrast/variance normalisation is used on the images to improve detection results. The detection rates (88.5-93%) presented vary depending on the level of data elimination. The AdaBoost detection system in Wu et al. (2009) scans for partially occluded cars and buses using images of different resolution (very low resolution up to 40x40 pixels) for the two classes. The detection of occluded vehicles using Haar-like features here requires significant pre-processing to produce sample images, while the system can only detect specific cases of occluded vehicles. Results show good performance in static images but detection rates in video lag behind.

In Sun et al. (2002b) and Sun et al. (2002a) a SVM classifier with different feature sets is used to perform classification. The Gabor features used in Sun et al. (2002a) give improved results compared to Haar wavelets in Sun et al. (2002b) although the classification process only takes place in subimages in Sun et al. (2002a). In the second study, the performance in complex scenes is explored, with a multi-scale edge detection process taking place to generate ROIs. Sun et al. (2006) test a series



of classifiers to determine which performs better at vehicle classification. The same multi-scale edge detection process is used, where the original image is downsampled to increase process speed. Results display the superiority of using SVM combined with HOG features, compared with other classifiers and features sets (Neural Network, Mahalanobis distance / Gabor features). Truong and Lee (2009) use long horizontal and vertical line detection for HG and PCA features combined with a SVM classifier. The most important features (i.e. Principal Components) that describe a vehicle are chosen manually and the image resolution is reduced in this study as well. Rybski et al. (2010) fuse camera and LIDAR data to detect vehicle orientation and a classifier with different image resolution to achieve their results. Their system is trained on 8 possible orientations and is able to correctly predict vehicle orientation with an accuracy of 88%, though testing concluded that using an orientation-independent classifier produces better results than attempting to determine individual vehicles through their orientation.

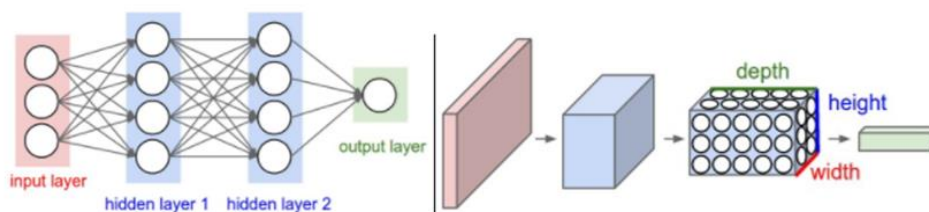
## **2.6 Convolutional Neural Networks and vehicle detection**

This section discusses the use of Convolutional Neural Networks (CNNs) as vehicle detectors in detail. The section begins by providing an introduction to the CNN, its architecture and building blocks. Different classification/detection algorithms based on CNN are presented and finally, the differences between traditional approaches to vehicle detection and CNN-based methods are discussed.

### **2.6.1 Architecture of a Convolutional Neural Network (CNN)**

A CNN (or ConvNN) is a type of Neural Network that has successfully been used to analyse visual information. It is a feed-forward network, meaning that information always moves towards one direction in the network; it never moves backwards as in recurrent neural networks. While CNNs are similar to ordinary Neural Networks (they are made up of neurons with learnable weights and biases), they make the assumption that the input data are images and therefore their architecture is arranged accordingly, with their parameters set to make the learning process more efficient (LeCun et al., 2010; Karpathy and Li, 2019b).

Unlike regular NNs, neurons in CNNs are arranged in 3 dimensions: width, height and depth (with depth referring to the third dimension of an activation volume/number of filters used to produce the feature maps) with each neuron connected to a small region of the layer before it, called receptive field. An example of the difference between a regular and a ConvNN can be seen in Figure 2-19 below. In the CNN, each layer transforms the 3D input image (width x height x 3 (for RGB images)) to a 3D output volume:



**Figure 2-19: Regular NN (left) - CNN (right)** Source: Karpathy and Li (2019b)

As described earlier, a Neural Network is a series of layers, one after the other, where the output of one layer is transformed in the next through a differentiable function. In a CNN, there are four main operations taking place, each in its own layer:

1. Convolution (Convolution Layer)
2. Introducing non-linearity
3. Pooling or sub-sampling (Pooling Layer)
4. Classification (Fully Connected Layer)

The four main types of layers are supplemented by others, performing other functions such as normalisation or dropout (Krizhevsky and Hinton, 2012; LeCun et al., 2010; Bishop, 2007; Plemakova, 2018; Karpathy and Li, 2019b). Additionally, a loss layer at the end of the network is used to calculate probability scores for different object classes. The sum of the probability scores for all mutually exclusive object classes is 1.

The structure and operation of a CNN is detailed extensively in the literature (LeCun et al., 1989; LeCun et al., 1998; Bishop, 2007; Plemakova, 2018; Karpathy and Li, 2019b) :

## Convolution layer

The convolution layer is the core building block of the CNN and the whole network takes its name after the specific operation. The primary purpose of convolution is the extraction of features from the input image. Essentially, it serves the same purpose as other feature descriptors used in image processing (HOG, SIFT, Haar wavelets etc.) (Zhao et al., 2019).

As discussed before, every image is a matrix of values, which indicate the brightness of a pixel. For greyscale images, pixel values range from 0 to 255 (with 0 indicating black colour, 255 white colour and values in between them different scales of grey). For RGB colour image, values range from 0-255 for each of the colour channels (Red, Green and Blue).

To perform convolution, small sized 2D matrices (e.g. 3x3 pixel size) called filters (or alternatively, kernels) are used on the 2D image matrix to compute the **dot product**:

- i. The filter is slid across the input (image) matrix by a fixed number of pixels (e.g. 1 pixel) which is called stride.
- ii. For every position, an element-wise multiplication is performed between the two matrices
- iii. The multiplication outputs are summed to get the final integer, which forms a single element of the output matrix.

Since the filter is limited in size compared to the input matrix, the filter only “covers” a part of the image in each stride (receptive field). The final product of this process is called **convoluted feature** or **feature map**. The output feature map varies, depending on the filter used for convolution. In image processing, different filters are used for different operations, such as edge detection, sharpening or blurring an image etc.

Before training a CNN, parameters such as number of filters, filter size and stride need to be specified for the convolution layer:

- i. The number of filters determines the “depth” of the convolution layer, i.e. the number of feature maps produced through the convolution operation. For example, if the number of filters is set to 5, 5 different maps containing different information will be produced.
- ii. Filter size, which is the size of the filter matrix (3x3, 5x5 etc.).
- iii. Stride, which is the number of pixels by which the filter matrix slides over the input matrix. A larger stride will produce a smaller feature map, making the training process faster, but running the risk of missing useful information from the image.

Additionally, it is possible to add zero values around the input matrix (zero-padding), which allows the filter to be applied on border pixels of the input image as well.

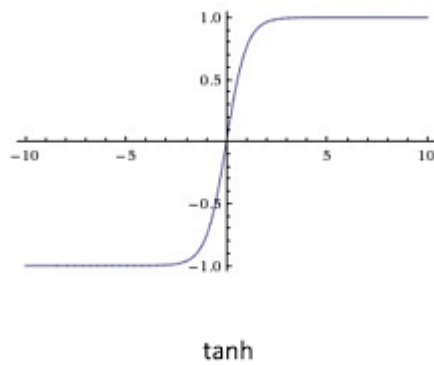
### **Introducing non-linearity to the network**

The convolution layer is followed by a layer tasked with introducing non-linearity in the CNN. Since convolution is a linear operation (matrix multiplication and addition) and most real-world data is non-linear, it is necessary to introduce non-linearity to the network. Non-linear means that the output cannot be reproduced from a linear combination of the inputs. Modern neural network models use non-linear activation functions that allow the model to create complex mappings between the network’s inputs and outputs. That way, the network can learn and model complex data such as images, videos or other datasets which are non-linear or have high dimensionality.

Linear functions are avoided in neural networks, because network models with linear activation functions are effectively only one layer deep, regardless of how complex their architecture is (the last layer of the network becomes a linear function of the first layer). Essentially, a neural network with linear activations is a linear regression model, with limited power and ability to handle complex input data.

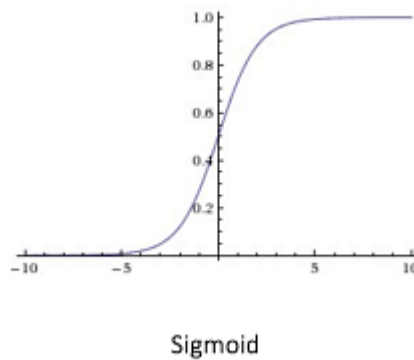
The most common non-linear functions used in CNNs are:

The hyperbolic tangent  $f(x) = \tanh(x)$  function,



**Figure 2-20: Hyperbolic tangent function**

The sigmoid  $f(x) = (1 + e^{-x})^{-1}$  function,

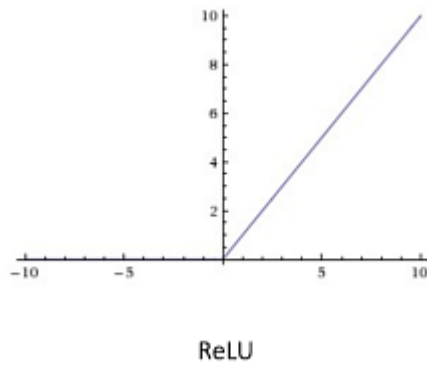


**Figure 2-21: Sigmoid function**

or some other function such as the ReLU function.

The function usually used in convolutional networks is the **ReLU** function or **Rectified Linear Units** function (Nair and Hinton, 2010):

$$f(x) = \max(0, x) \tag{2.20}$$



**Figure 2-22: Rectified Linear Unit function**

which replaces all negative values in the feature map with zero.

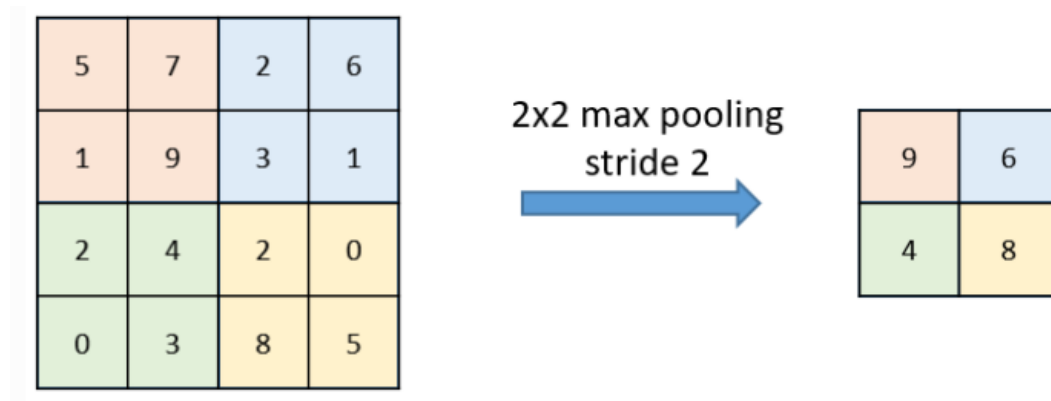
The ReLU function is preferred over others, because it has been found to train the neural network faster, with no significant penalty to accuracy. Its convergence rate is approximately 6 times faster compared to the *tanh* function (Krizhevsky and Hinton, 2012).

### **Pooling layer**

The pooling function (also known as subsampling or downsampling) aims to reduce the size of the feature map after the convolution and ReLU layers, while at the same time retaining the most important information. It is also helpful to avoid overfitting while training the network by reducing the number of parameters and computations and provides an almost scale invariant representation of the image (making detection of objects in an image easier, no matter where the object may be located). Among the various types, such as average, max, sum etc., **Max** pooling is the most common function to perform subsampling as it has proven to work better in practice (Krizhevsky and Hinton, 2012; Simonyan and Zisserman, 2014; Girshick, 2015).

In the case of Max pooling, the rectified feature map is partitioned into non-overlapping rectangles, and for each such rectangle, the maximum value is taken as representative for that region. Pooling operates along both width and height of the feature map and for every depth slice (Nagi et al., 2011).

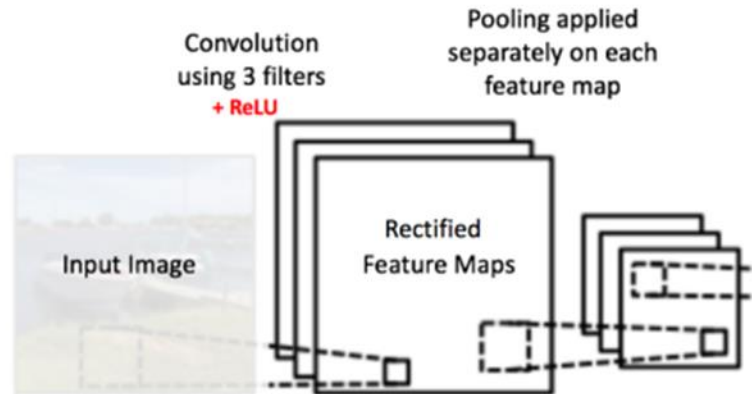
The size of the pooling window is defined beforehand. An example of a max pooling operation on a 4X4 input feature map using a 2X2 window can be seen in the figure below. The **max** operation is performed over 4 values with only 1 remaining, so in this case, 75% of the information is discarded:



**Figure 2-23: Example of max pooling operation**

The pooling windows are usually kept small in size (usually 2X2) because of their destructive nature (larger receptive fields discard a large amount of possibly useful data).

Figure 2-24 below is indicative of how pooling operates within the structure of a Neural Network, where the depth of the convolution layer is 3 (3 feature maps produced):



**Figure 2-24: Pooling operation**

### **Classification and Fully Connected Layers**

The Fully Connected Layer acts as the high-level reasoning part of the Neural Network and is responsible for classifying the input image into various object classes based on the data the network was trained on.

This layer is a traditional neural network layer that uses an activation function to produce its output and takes its name from the fact that every neuron in the previous layer is connected to every neuron in this one. Similarly to neurons in regular neural networks, the connections here have associated weight and bias values.

In comparison, other layers in the CNN such as convolutional or pooling layers are only partially connected, with each neuron in a convolutional layer only connected to a few local neurons in the previous layer. Classic neural network architecture (where all layers are fully connected) was found to be inefficient for computer vision tasks. Images represent such a large input for a neural network that would require a huge number of connections and network parameters. A CNN addresses this problem by considering the type of input data (images) and adapting to it, by having layers that are used for the extraction of useful features (convolutional, pooling) that are only partially connected. The fact that an image is composed of smaller details or features that can be processed individually to reach a decision about the image as a whole makes this possible.



The FC layer uses as input the output from all previous network layers before it (convolutional, ReLU and pooling). This input, in addition to the connection weights and the bias value are used to produce the output probabilities for each of the object classes in the dataset.

The FC layer is followed by an output layer. In classification problems, the softmax function is commonly used and is tasked with predicting a single class out of  $K$  mutually exclusive classes (it is multinomial logistic regression used for multi-class classification). Other classifiers, such as SVM, can also be used for this task with their performance being comparable. Softmax is usually preferred because it converts an output of arbitrary real-valued scores into probabilities that add up to 1, which is more intuitive. In comparison, SVM treats the output as uncalibrated scores that are difficult to interpret (Qi et al., 2017).

The output softmax function is:

$$y_r(x) = \frac{\exp(a_r(x))}{\sum_{j=1}^k \exp(a_j(x))} \quad (2.21)$$

Where  $0 \leq y_r \leq 1$ ,  $\sum_{j=1}^k y_j = 1$  and  $a_r$  is the conditional probability of the sample class  $r$ .

For multi-class classification problems, the softmax output function is:

$$P(c_r|x, \theta) = \frac{P(x, \theta|c_r)P(c_r)}{\sum_{j=1}^k P(x, \theta|c_j)P(c_j)} = \frac{\exp(a_r(x, \theta))}{\sum_{j=1}^k \exp(a_j(x, \theta))} \quad (2.22)$$

Where  $0 \leq P(c_r|x, \theta) \leq 1$  and  $\sum_{j=1}^k P(c_j|x, \theta) = 1$ . Moreover,  $a_r = \ln(P(x, \theta|c_r)P(c_r))$ ,  $P(x, \theta|c_r)$  is the conditional probability of the sample given class  $r$  and  $P(c_r)$  is the class posterior probability.

The softmax classifier attempts to minimise the classification error (target probability – output probability) by minimising the *cross-entropy* between the “true” distribution  $p$  and an estimated distribution  $q$ :

$$H(p, q) = -\sum_x p(x) \log q(x) \quad (2.23)$$

More information on the softmax function can be found in Bishop (2007).

To summarise, the training of a Convolutional Neural Network is as follows:

- The network structure is defined and all filters and parameters initialised with random values.
- The network takes the first training image as input, goes through the complete process and calculates the output probabilities for each object class (since weights are randomly assigned in the beginning, output probabilities are also random).
- The total error at the output layer is calculated.
- Using back propagation (LeCun et al., 1989; Bishop, 2007), the gradients of the error are calculated and gradient descent is used to update all filter values and weights to minimise the output error (the weights are adjusted in proportion to their contribution to the total error).

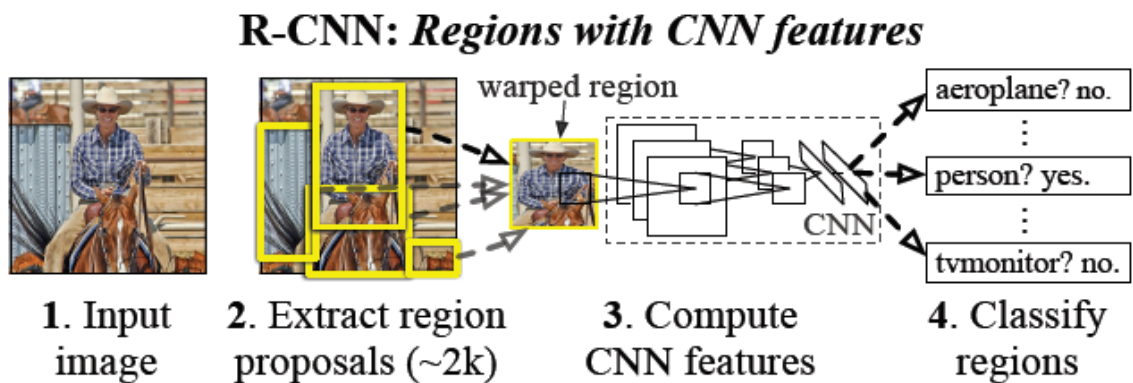
Now, if the same image is used again, the output probabilities will be closer to the target, as the weights and parameters of the network have been optimised to correctly classify that image. The same process (except initialising the parameters) is used for every image in the training set and the end product is a Neural Network that, given a large enough training dataset (so that its parameters are well adjusted), is accurate enough to correctly classify new images.

### **2.6.2 CNN-based vehicle detection**

Vehicle detection using deep convolutional networks is currently being researched extensively. Inspired by the success of CNNs in image classification (Krizhevsky and Hinton, 2012), several detection models have emerged. Most CNN-based detectors are based on the R-CNN (or Regions with Convolutional Neural Network) (Girshick et al., 2014) and its evolutions, Fast R-CNN (Girshick, 2015) and Faster R-CNN (Ren et al., 2015). Others, such as R-FCN (Region-based Fully Convolutional Networks) (Dai et al., 2016) , SSD (Single Shot Multibox Detector) (Liu et al., 2016) and YOLO and its variations (Redmon et al., 2016; Redmon and Farhadi, 2017)

brought changes in the architecture with the goal of optimising real-time performance.

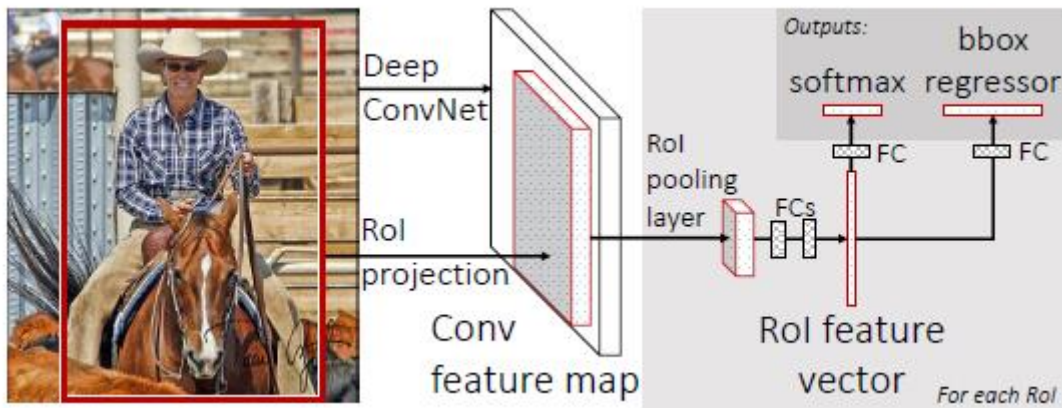
When it was initially introduced in 2014, R-CNN produced state of the art performance by combining the classification method Alexnet (Krizhevsky and Hinton, 2012) with an external Region Proposal method to generate candidate object locations. RP methods such as Selective Search (SS, used in the original implementation of R-CNN) (Uijlings et al., 2013), EdgeBoxes (Zitnick and Dollár, 2014) and others could be used to generate the region proposals which were then fed to the Alexnet network. On the final layer of the CNN, a SVM classifier (used to determine the object class) and a linear regression model (used to improve the bounding box coordinates) were added. While outperforming traditional detection methods, R-CNN was very slow (SS produced around 2,000 region proposals that each had to pass through Alexnet) and it was impossible to achieve real-time performance. Training the network is also expensive in memory space, as the extracted features from all region proposals need to be stored. The operation flowchart of R-CNN can be seen in Figure 2-25 below:



**Figure 2-25: R-CNN flowchart**    **Source:** Girshick et al. (2014)

Fast R-CNN significantly reduced the computational cost by sharing the feature map generated for the entire image for the region proposals. Now the feature map is only calculated once in the beginning. A fixed-length feature vector is extracted from each region proposal with a ROI pooling layer. Each vector is fed into the FC layers before branching into two output layers. The two output layers, a Softmax layer

producing probability scores, and a bounding box regressor layer are now incorporated into the model instead of being separate as before (Girshick, 2015). The operation flowchart for the Fast-RCNN model can be seen in Figure 2-26 below:



**Figure 2-26: Fast R-CNN flowchart** Source: Girshick (2015)

Faster R-CNN further improved computational speed by including an RPN (Region Proposal Network) into the Fast R-CNN model, instead of relying on an external process. The region proposals are now generated straight from the convolution map, effectively minimising computations. Region proposals are generated in different sizes and scales and each of the boxes is given an objectness score (Ren et al., 2015).

The RPN is a fully convolutional sub-network, able to predict object bounds and probability scores at each position simultaneously. Using one convolutional layer's output (conv feature map), object proposal boxes are generated in different sizes and scales, which then serve as additional input data to the classification layer. A sliding window moves across the feature map and generates 9 region proposals in every position (3 scales and 3 sizes). The region proposals are called anchor boxes, since they are centred on a fixed point. The produced vectors are fed into two sibling FC layers, the box-classification (cls) layer and box-regression (reg) layer, which score and generate object bounds respectively. The RPN in Faster R-CNN can be seen in Figure 2-27 below:

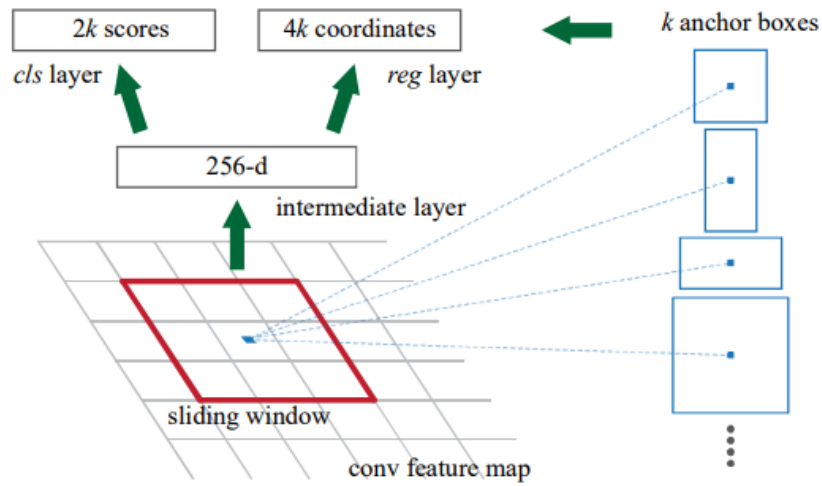


Figure 2-27: The RPN in Faster R-CNN Source: Ren et al. (2015)

The output of the RPN then returns to main network for classification. The structure of the Faster R-CNN model can be seen in Figure 2-28 below:

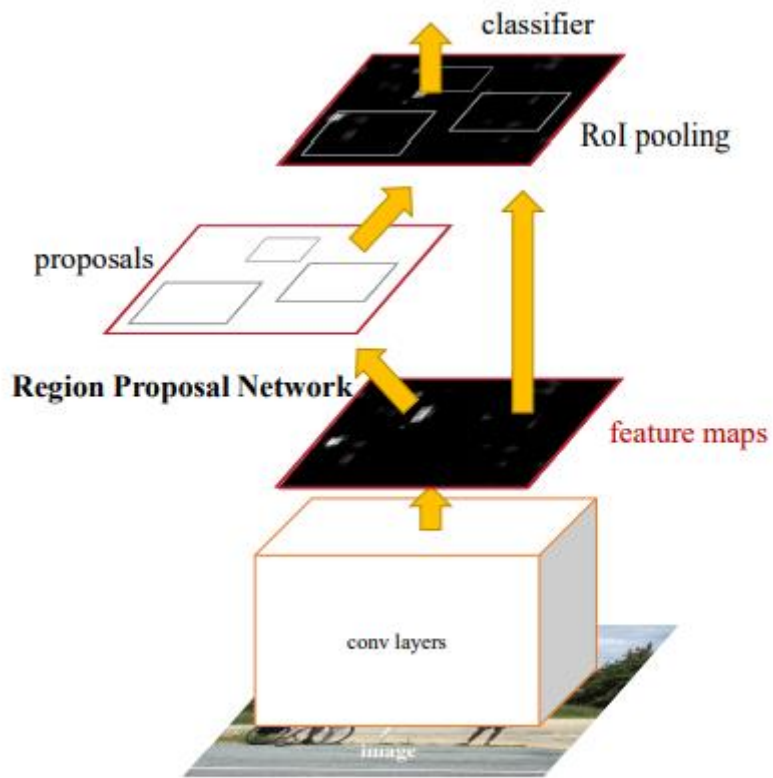


Figure 2-28: Faster R-CNN flowchart Source: Ren et al. (2015)

With the inclusion of a built-in region proposal unit, the CNN can be trained without the need for external region proposal processes. Training takes place in four stages. The first 2 train the region proposal network (RPN) and CNN while the other 2 combine the output of the first 2 stages and fine-tune the network:

Stage 1: Training a Region Proposal Network (RPN)

Stage 2: Training a Fast RCNN Network using the RPN from stage 1

Stage 3: Re-training RPN using weight sharing with Fast RCNN

Stage 4: Re-training Fast RCNN using updated RPN

Faster R-CNN was originally tested on the Pascal VOC 2007/2012 (Everingham et al., 2010) and MS COCO (Lin et al., 2014) datasets for detection and achieved very good detection accuracy at 5FPS when using the deep VGG-16 model (Simonyan and Zisserman, 2014) for feature extraction.

The R-FCN model, developed by Dai et al. (2016), is similar to Faster R-CNN using a RPN but eschews Fully Connected layers completely to become a fully convolutional network. Figure 2-29 shows the architecture of the system. In R-FCN, the last convolutional layer produces position-sensitive scores for every object class. The ROIs produced by the RPN are applied and a final score is calculated for each generated ROI. In the end, a softmax classifier assigns classes to the objects.

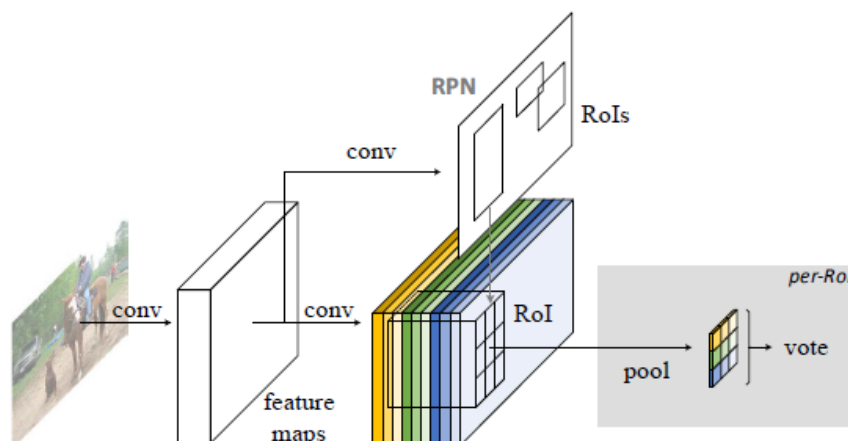
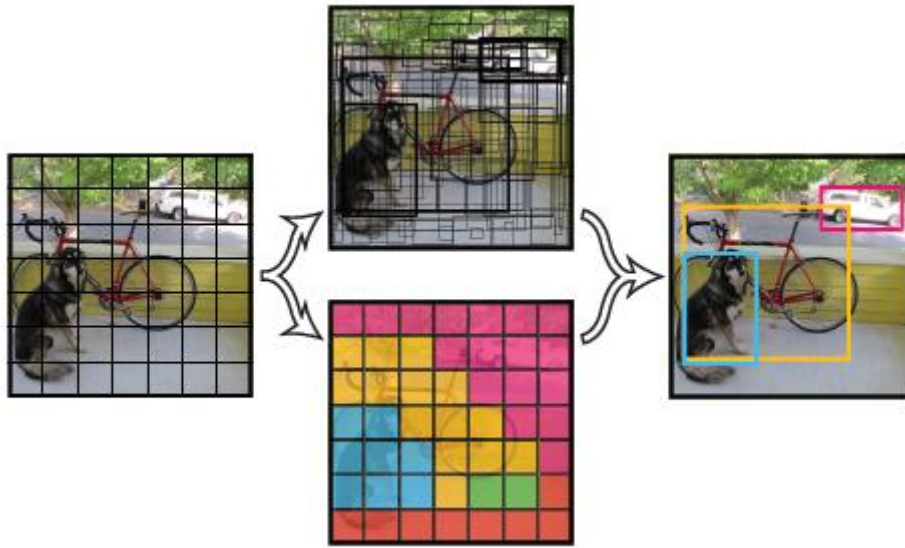


Figure 2-29: R-FCN flowchart

Source: Dai et al. (2016)

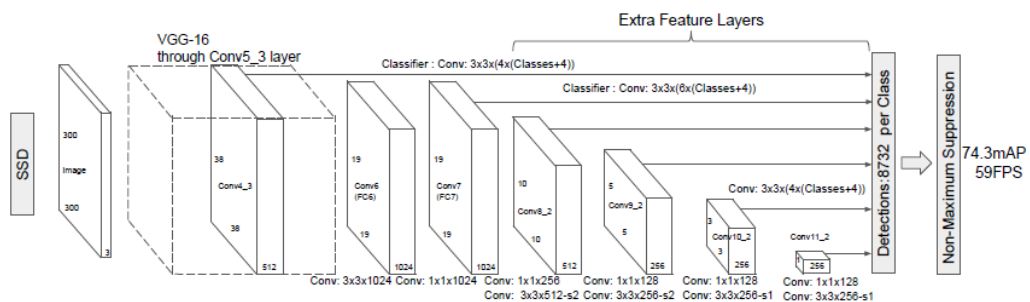
The methods described above all follow the traditional two-stage object detection pipeline, where region proposals are generated (Hypothesis Generation – HG) and then proposals are verified (Hypothesis Verification – HV) by assigning probability scores and bounding boxes. Other methods treat object detection as a regression/classification problem, adopting a unified framework to achieve their results (class categories and locations) directly, without the use of generated region proposals. The best-known methods following this approach are YOLO and its variations (Redmon et al., 2016; Redmon and Farhadi, 2017) and SSD (Liu et al., 2016).

With YOLO, Redmon et al. (2015) proposed a novel framework for object detection which makes use of the whole feature map produced for an image to predict confidence scores for multiple categories and bounding boxes. The basic idea behind YOLO is to divide the input image into a grid and predict the object that exists in each grid cell. Each grid cell predicts multiple bounding boxes and their corresponding confidence scores, along with conditional class probabilities. The highest scoring boxes are retained as the final detection results. The YOLO network consists of 24 convolutional layers and 2 Fully Connected layers and in its first iteration managed to process 45FPS. A subsequent iteration, YOLOv2 by Redmon and Farhadi (2017) added some improvements such as batch normalisation, anchor boxes and multi-scale training to improve the detection result. An inherent problem with YOLO is their issue with detecting small objects or objects close to the camera, as it is constrained by the one object/grid cell rule.



**Figure 2-30: YOLO detection system** Source: Redmon et al. (2015)

Similarly to YOLO, SSD (Liu et al., 2016) uses the entire feature map for its predictions but forgoes grid cells, instead using a set of default anchor boxes. To handle objects of various sizes, the network fuses predictions from multiple feature maps with different resolutions. SSD uses the VGG-16 (Simonyan and Zisserman, 2014) network for feature extraction but adds several layers to the end of the network to calculate confidence scores and predict the bounding boxes. The SSD network achieves 59FPS using 300x300 resolution images but still suffers when dealing with small objects, requiring further modifications. The SSD architecture can be seen in Figure 2-31 below:



**Figure 2-31: SSD network architecture** Source: Liu et al. (2016)



The models presented above spurred research in vehicle detection using CNNs, with advances in vehicle detection using both static and moving cameras. In the majority of cases, it is opted to use pre-trained network models as feature extractors as a starting point for vehicle detection. This approach is called **transfer learning**; it is essentially the re-purposing of a model trained to perform one task to work on a different task. It is possible to either use a CNN as is (as a feature extractor) and only replace the final FC layer with one better suited to the new task or fine-tune the network fully (all layers) or partially (the last layers that should be specific to the new task) with a new dataset. The most common CNN architectures used for transfer learning are AlexNet (Krizhevsky and Hinton, 2012), VGG (Simonyan and Zisserman, 2014), ZF-net (Zeiler and Fergus, 2013), ResNet (He et al., 2016) and GoogleNet (Inception) (Smoluk, 2015).

A CNN (Alexnet) is used in Yao et al. (2017) as a classifier to identify vehicles in traffic videos. First, region proposals are generated by combining three visual cues (multiscale saliency to distinguish vehicles from the background, edge density and colour contrast) into a Bayesian classifier. A pre-trained Alexnet model then classifies the vehicles. A detection rate of 93% on average is reported, with 5sec of processing required for each frame. Hsu (2018) use a sliding window approach to generate ROIs and Fast R-CNN as a classifier for vehicle detection. High precision and recall rates are reported, though run-time performance is not discussed. Prabhakar et al. (2017) use Faster R-CNN with a pre-trained ZF-net (Zeiler and Fergus, 2013) on a Titan X GPU to achieve a 71.7% mAP (mean Average Precision) on the KITTI dataset (Geiger et al., 2012). Several parameters of the Faster R-CNN model are tested in Fan et al. (2016), where the effects of image size, number of proposals and additional training stages are explored. Accuracy ranges from 52-83% on the KITTI dataset, with a run-time of just 2FPS when tested on an 1800x543 resolution image.

The RPN in Faster R-CNN is modified in Gao et al. (2017) to enhance small object detection. By adding additional scales for the anchor boxes (for a total of 15 instead of the default 9) and modifying the CNN architecture, precision is improved. However, no run-time performance is reported, although the modifications should

bring an extra computational load that impacts performance. Zhang et al. (2017) attempt to implement a real-time vehicle detection and tracking system based on Faster R-CNN. The KITTI dataset is used to train the VGG-based network while a combination of Camshift and Kalman filter is used for tracking. The reported computation time for target detection and tracking is 0.935s and 0.0244s respectively, making it unsuitable for real-time application. He and Li (2018) fuse camera and radar data to detect vehicles in traffic videos. The results of the YOLOv2 model are improved by fusing radar detections and the final system has an accuracy of 96% at 38FPS. A SSD based model, fine-tuned to identify small objects more efficiently is used by Kim et al. (2016) to detect vehicles, pedestrians and cyclists. Tested on the KITTI dataset, the best-performing model tested manages 70.7-86.7% accuracy for the vehicle class. Gu et al. (2017) develop a CNN that includes Inception modules, inspired by GoogleNet (Smoluk, 2015) for vehicle detection. The system is tested on the VOC 2007+2012 dataset (Everingham et al., 2010) and achieves 63.5% mAP at 46FPS. In He and Lam (2018), a deep residual network is used to extract image features while a lateral network is used to improve localisation of the final output. The proposed LateralCNN is evaluated on the DETRAC (Wen et al., 2015) traffic camera benchmark and achieves an accuracy of 67.25% on average at 28FPS.

## **2.7 Range estimation using a monocular camera**

Accurate range estimation is a crucial task for safety-critical applications such as ACC and CAS. Systems using active sensors (radar, LIDAR) can accurately measure distances due to the way they operate (measuring the reflection of emitted signals for radar or laser beams for LIDAR systems) without requiring advanced processing methods. Range estimation using a monocular camera however, is challenging since the camera image is subject to perspective distortions and limited accuracy (Joglekar et al., 2011; Eskandarian, 2012; Mukhtar et al., 2015; Huang et al., 2019). To overcome those limitations, the main approaches to produce accurate measurements are the use of filters (e.g. Kalman) to update measurements and avoid large deviations or the introduction of methods to reduce the associated errors.

Also known as linear quadratic estimation (LQE), a Kalman Filter (Kalman, 1960) is an algorithm widely used for guidance, navigation and control of vehicles. It uses a

series of measurements observed over time, containing noise and other inaccuracies, and produces estimates of unknown variables that tend to be more precise than those based on a single measurement alone. The version of the filter used for non-linear processes is called Extended Kalman Filter.

In Stein et al. (2003), range is measured using the camera parameters and perspective. To increase the accuracy of the measurement, two cues are used: size of the vehicle in the image and position of the bottom of the vehicle. The estimate is more accurate when the geometry of the road and the range rate are considered in the calculation. Finally, the output of a radar sensor is used as ground truth. In Han et al., (2016), the distance to a detected vehicle can be estimated by calculating the width of the vehicle and using lane markings as reference. The method is accurate on the assumption that lane width remains constant but fails if the road environment changes or roads are not structured. In Salari and Ouyang (2013), a SVM is utilised to estimate the position of a vehicle. By considering the width or height of the vehicle known, it is possible to estimate the distance and calculate TTC by measuring the vehicles width change in a sequence of images. The system is not accurate when the vehicle is far from the camera and the detected change in width is small. Moreover, the SVM is slow and not suitable for real-time application. An estimation based on camera height position, focal length, coordinates of the vehicle bottom and horizon is used in Park and Hwang (2014) but the calculation is not accurate when there is any variance in the horizon position. Lim et al. (2019) propose the use of CNN-based detector (YOLO) and a nested Kalman filter to first stabilise distance data from the camera and then use this filtered data in the Kalman filter again to calculate relative velocity. This way, TTC is calculated based on distance over relative velocity. Joglekar et al. (2011) estimate depth using a monocular camera by applying perspective geometry and correcting errors in the calculation of in-path and oblique distances to an object. Their approach to error calculation gives accurate distance measurements up to 70m from the camera. Christiansen et al. (2018) use a CNN to produce bounding boxes for vehicles and a Kalman filter to correct the distance measurements for the bounding box width and the distance to the ground. A LIDAR system is used as ground truth for the proposed system. Huang et al. (2019) utilise a CNN-based approach to detect vehicles and segment them from the surrounding environment. A

different NN calculates the vehicles' actual dimensions and pose. A geometric model then estimates the distance to the vehicle.

## **2.8 Knowledge gap**

This chapter reviews the most common methods used for vehicle detection based on a monocular camera. The two approaches (traditional image processing and CNN-based detection) are explored and the main findings are discussed in this section:

### **Image processing approach**

The section about Hypothesis Generation focuses on appearance-based methods, as the proposed system is based on a monocular camera, identifying the prominent visual cues used for ROI generation and highlighting their shortcomings. Object classifiers are the focus of the Hypothesis Verification section, given that they are more adaptable compared to template matching methods.

As seen in the literature, the Support Vector Machine (SVM) and AdaBoost classifiers are the most common methods for image segment classification. Paired with the appropriate feature set, they appear to be the most efficient and produce the best results (Khammari et al., 2005; Teoh, 2011; Burlet and Dalla Fontana, 2012; Chavez-garcia and Aycard, 2015).

Compared to other descriptors, HOG features are computationally expensive but they are highly adaptable and descriptive and well suited for this application, providing good detection results (Teoh, 2011; Li and Guo, 2013; Zhiqian Chen et al., 2013). Another advantage of using an SVM-HOG combination is its versatility in that it can be employed for pedestrian detection as well. After reviewing the some of the most recent literature on vehicle detection using HOG features, some conclusions can be drawn:

The main problem to be addressed is the minimising of computational time so that camera-based detection systems are implemented in real-time. The accuracy level in most cases is very good, the reason being that computationally complex methods

(classifiers combined with feature descriptors) are used for detection. It is obvious that the problem's solution lies with efficiently balancing detection accuracy and speed, or the implementation of a new, intelligent method.

In most cases, the run-time problem is attempted to be solved either with experimenting with feature descriptor parameters (in the case of HOG, bin size, block size etc.) altering the descriptive power of the feature descriptor or by reducing the search space in Hypothesis Verification, so that the total number of calculations is reduced.

The common theme across all relevant research is that traditional feature descriptors (and their variations) are used for classification purposes (combined with a classifier function). They are all focused on the second part of the detection process (HV), while for the first part (HG), other methods are used. A new way of using existing tools, such as classifiers and feature descriptors could lead to improved results, as far as detection performance is concerned. This study will attempt to optimise performance by modifying the detection pipeline.

### **CNN-based detection**

The review of different network models that can be used for vehicle detection (Huang et al., 2017; Zhao et al., 2019) has led to some interesting findings:

- CNNs have managed to combine every aspect of object detection (ROI generation, feature extraction and verification) into one unified pipeline, as opposed to traditional image processing object detection where each stage is largely separate.
- Region proposal based methods, such as Faster R-CNN and R-FCN generally perform better compared to regression/classification based approaches (YOLO, SSD etc.) due to the fact that regression approaches produce increased localisation errors.
- Regression/classification methods have trouble locating small objects, which may be an issue when detecting vehicles that are far away from the camera.

- R-FCN and SSD models are faster on average but Faster R-CNN is more accurate.
- Input image resolution impacts accuracy. Low resolutions hurt accuracy significantly, though at the same time inference time is reduced.
- Training complex and deeper networks used for feature extraction such as ResNet (He et al., 2016) takes more time, but this time consumption can be reduced by adding as many layers into shared fully convolutional layers as possible.
- Region proposal based models can be modified to improve run-time and achieve real-time performance with the introduction of “tricks” such as batch normalisation (Ioffe and Szegedy, 2015) or modifying the detection parameters (for example, reducing the number of generated proposals).

The common theme in CNN-based detection is the reliance on deep pre-trained network structures, especially for feature extraction. The potential to achieve the same level of performance using simpler and more efficient structures is not explored sufficiently well; this study will explore the detection performance of a simpler network structure.

In search of a robust vehicle detector built around the capabilities of a monocular camera, two separate detectors can be developed and their performance explored:

- i) A detector following the traditional image processing approach using HOG feature extraction and SVM classification, modified in an attempt to improve run-time operation.
- ii) A CNN-based detector built on the Faster R-CNN model, with the goal to improve its performance and achieve real-time detection. The particular model (Faster R-CNN) was selected as, even though it is not the fastest compared to others, it is high-performance and with no inherent disadvantages. Real-time operation can be achieved by modifying the network structure as to find a good balance between accuracy and speed.

## **3 Methodology**

### **3.1 Introduction**

The previous section described the main approaches used in a camera-based vehicle detection system. This chapter presents the two methods used for vehicle detection in this project. The two methods will be tested on a vehicle dataset to determine which is better in terms of detection performance and whether they are suitable for real time application.

The end goal is to develop a vehicle detection system with:

- High detection rate, minimising false positive (FP) and false negative (FN) detections
- Real-time performance (computationally efficient for use in a practical application)
- Versatility, so that additional functionality can be added without seriously impacting performance
- Ability to perform under different conditions (urban environment, motorway, varying weather conditions)

The system is built around a NIR (Near Infrared) monocular camera that provides the data feeding into the detector. It is necessary to maximise the detection system's performance, as the camera is the only available environmental sensor and its capabilities are limited compared to a system utilising an array of sensors.

### **3.2 Vehicle detection I – Histogram of Oriented Gradients (HOG) and Support Vector Machine (SVM) classification**

The first method of vehicle detection is based on a combination of HOG features and SVM classification. The traditional image processing method is modified so that HOG features are also used to generate Regions of Interest (ROIs), instead of using a sliding window approach or using HOG solely for classification. It will be examined

whether this approach (using information not coming directly from the raw image data) is suitable for ROI generation.

The flow chart below (Figure 3-1) presents the detection process that is the centre of the detection system. The complete process is presented in more detail in the next section:

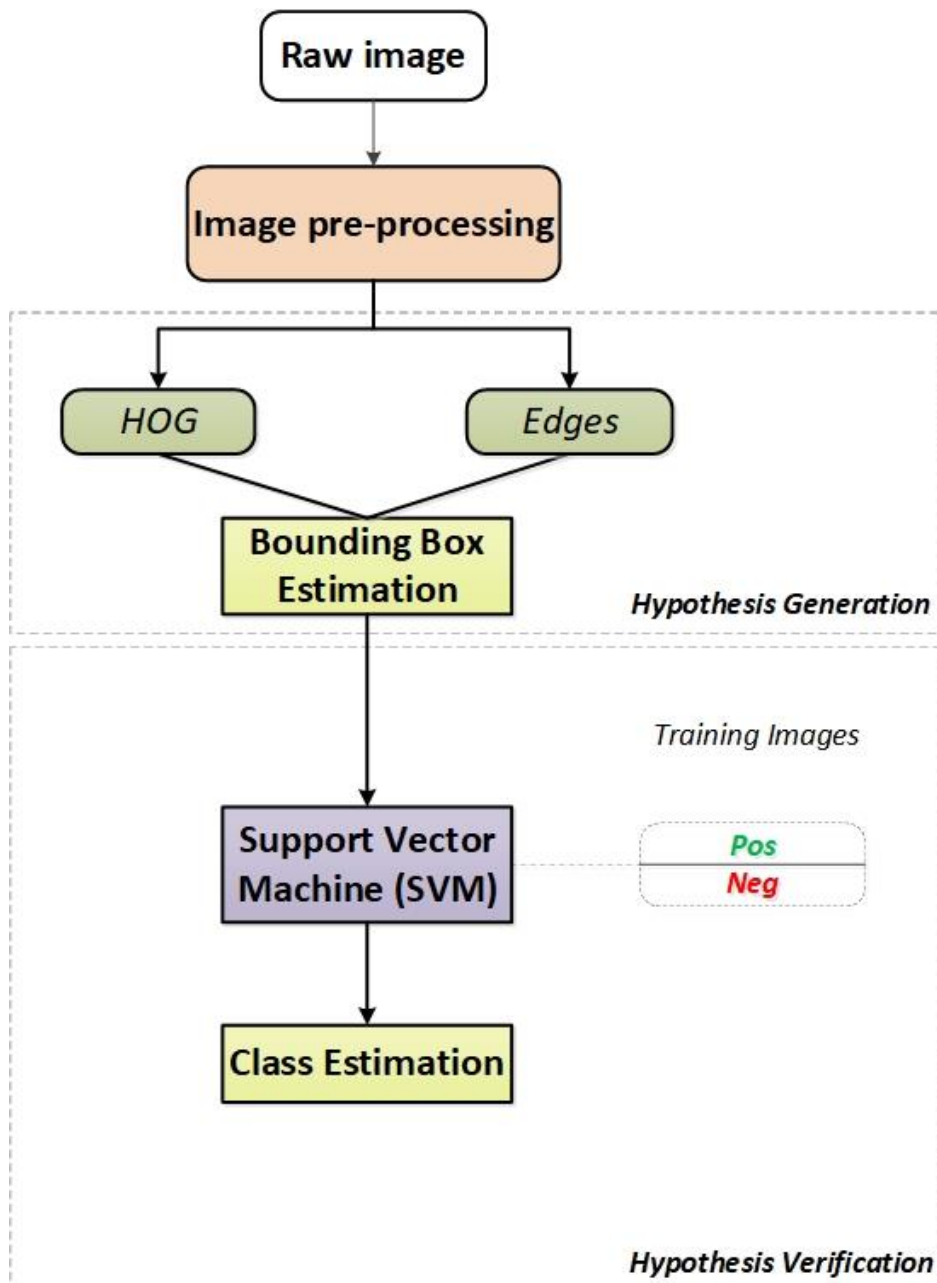


Figure 3-1: Detection system



The detection process is divided in two parts:

- Hypothesis Generation (HG)
- Hypothesis Verification (HV)

### **3.2.1 Hypothesis Generation (HG)**

The first part of the detection process is Hypothesis Generation that follows an appearance-based approach. Specific visual characteristics are sought for in an image in order to detect a potential object. This approach is preferred over a motion-based approach since a monocular camera is used. While there are methods to compensate for the lack of depth measurements for cameras of this type, motion-based approaches are better suited to stereo vision cameras and are more resource intensive. Additionally, it would be highly desirable to detect slow moving objects as well; something a motion-based system has issues with (Mukhtar et al., 2015).

The proposed detection algorithm exploits the feature vector generated from HOG extraction to detect strong horizontal elements in an image. Instead of using filters that are most commonly used to detect edges such as Canny (Canny, 1986) or Sobel (Sobel, 1990), HOG gradients can be used to indicate horizontality in an object. Due to their shape, vehicles exhibit areas with strong horizontal elements (edges), mainly but not exclusively, their roof and bottom.

A sample image, taken using the instrumented vehicle's NIR camera can be seen below. The same image, after the application of Sobel and Canny filters follows next.



**Figure 3-2: Sample image**



**Figure 3-3: Sample image with Sobel filter applied**



**Figure 3-4: Sample image with Canny filter applied**

Instead of using traditional visual cues (edges using filters and shadow detection being the most common, this method proposes that the HOG feature extraction process, usually employed for verifying the existence of objects is used to generate vehicle candidate locations. The reason for choosing the particular detection cue is that it may lead to shorter processing time. The potential benefit from using HOG as a ROI estimator lies with the ability to reduce the different functions required for the detection process. Instead of using shadows, corners etc., the same process used for classification is also utilised for initial detection.

The goal is to maximise detection accuracy using this HOG-based detector. To support the main HOG-based detector, another visual feature can be implemented, if the required performance levels (detection rate, false positive rate) are not met. The system is flexible enough so that other visual cues can be added if necessary. The main considerations for selecting a particular visual cue (for example edges, symmetry etc.) are the limitations imposed by the data and the processing time they require (Teoh, 2011; Mukhtar et al., 2015). For this study, edge detection using the Canny filter was utilised to improve object separation in one of the image processing steps.

The HG stage results in image parts (sub-images) that are smaller than the original image recorded by the camera. Ideally, the sub-images contain the vehicle with as little unnecessary information (surrounding environment) as possible. The potential vehicle is highlighted in the image by a rectangular box.

The sub-images are used as input information for the next stage of the vehicle detection process, where the system confirms the presence of a vehicle in the image or not. The process of generating the bounding boxes for candidate vehicle locations can be divided in several steps:

- i. Image (raw data) pre-processing
- ii. HOG feature extraction
- iii. Feature vector processing
- iv. Clustering/Segmentation
- v. Effective vehicle-environment separation

vi. Bounding box generation

The proposed Hypothesis Generation (HG) system is presented in Figure 3-5 below. First, the input colour image extracted from videos recorded during data collection runs is converted to grayscale and scaled down to a lower resolution. HOG features are extracted from the reduced resolution and the strong horizontal edges are detected. The resulting horizontal edge image is further processed to separate objects of interest from clutter. Finally, the bounding box around each hypothesized location is estimated and drawn on the original processed image. The final output is a ROI for which the HOG features required for classification have already been extracted.

Each part of the Hypothesis Generation process will be presented in more detail in Figure 3-5 below:

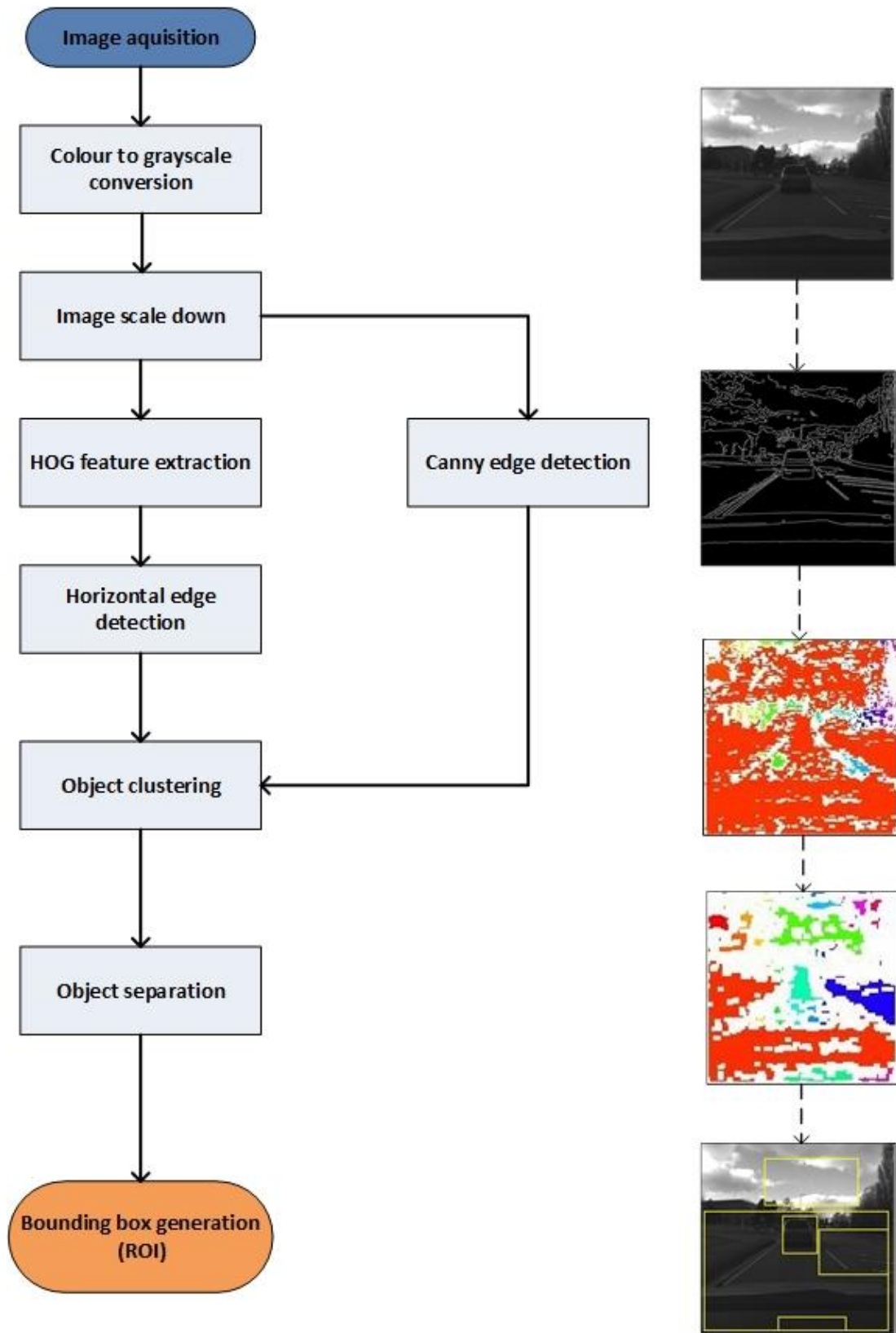


Figure 3-5: Proposed HG system

### **Image pre-processing**

The collection of raw data and its features is described extensively in Chapter 4. The first step after acquiring the raw data from the camera (video file that is transformed into individual frames) is to convert each frame from colour to grayscale. The NIR camera installed in the instrumented vehicle produces video and images in the RGB colour space, though the image properties indicate that each pixel carries the same value across all colour channels. The image is converted from RGB to grayscale using the following formula, which is a weighted sum of the 3 colour channels. Each pixel in an image has a numeric value (0-255) for each of the colour channels. Every value is multiplied with the corresponding coefficient:

$$0.2989 * R + 0.5870 * G + 0.1140 * B \quad (3.1)$$

Where the coefficients 0.2989, 0.5870 and 0.1140 are the weights for converting from RGB to NTSC, according to the Rec. 601 format (International Telecommunication Union, 2011).

Converting the image from RGB to grayscale reduces the computational load, as many functions in the detection process perform their operation individually for each colour channel. For example, HOG feature extraction is performed for every colour. Having only one colour to consider, the whole operation becomes much faster. In addition, since each pixel carries the same value for all three colours, considering colour as an additional visual cue in the current vehicle detector and benefiting from it, is not possible.

Next, the images are resized into a lower resolution, another operation that takes place to reduce the computational cost. The original size of the images recorded is 2048x2048 pixels (4 MPixels). Using a lower resolution image, the size of the feature vector file when producing HOG features is reduced significantly:

**Table 3-1: Vector file size comparison**

<b>Image size</b>	<b>Feature vector size</b>	<b>Size reduction</b>	<b>Avg. processing time</b>
2048x2048	2,340,900	-	0.46 sec
1024x1024	580,644	75.20%	0.19 sec
512x512	142,884	93.90%	0.1 sec

The size of the original image's produced feature vector prohibits any thought of real-time application. Using a lower resolution image is required. No significant descriptive information is lost, while the resulting feature vector becomes much more manageable. Additionally, the required time to extract HOG features is reduced significantly when the image is downscaled. The produced vector sizes in Table 3-1 are based on the default HOG extraction options (8x8 pixel cell size, 2x2 cell blocks, 50% block overlap, 9 orientation bins). Any configuration adjustment modifies the final size accordingly.

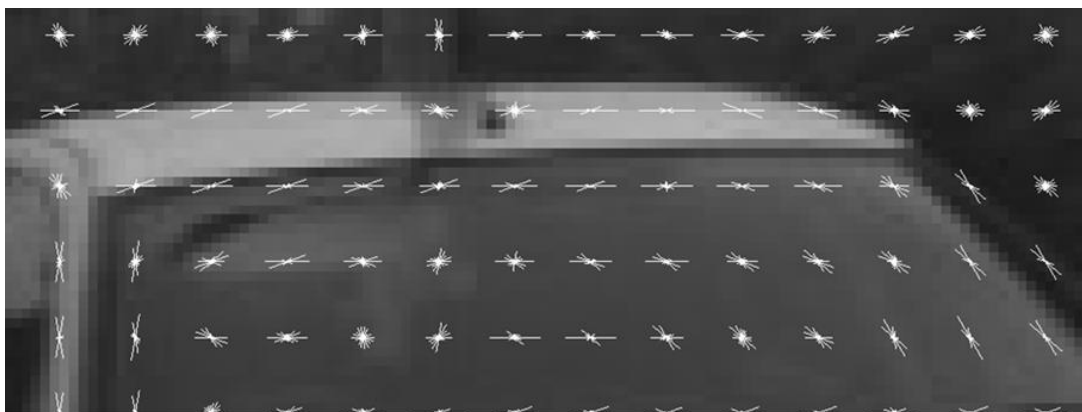
To summarise, the conversion to grayscale and downscaling the image to a lower resolution results in a reduced computational load, which is a requirement for real-time operation. At the same time, there is no loss of useful information in the resulting image compared to the original as the resolution remains high enough to ensure this (every object remains easily discernible, no blurring occurring in the image).

### **HOG feature extraction**

The second part of the process involves the extraction of HOG features from the images. The idea behind HOG features is that they can be used to describe object appearance and shape by their distribution of intensity gradients or edge directions (Dalal and Triggs, 2005). The descriptive power of HOG is entirely dependent on the level of detail desired. Even though there is a trade-off between accuracy and computational load, it has been reported that after a certain point, there is no gain in descriptive power (only an increase in computational requirements) (Seung Hyun Lee et al., 2015; Teoh, 2011). The extraction process has been detailed extensively in section 2.5.1 of the literature review.

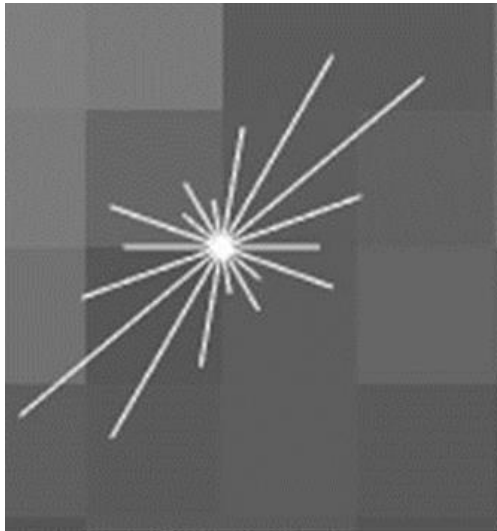
The MATLAB software was used for the extraction of HOG features from the sequence of images. The extraction process allows for a certain level of parameterisation depending on the level of detail required. It is possible to change the size of the HOG cell (default size: 8x8 pixels), the number of cells in a block (default size: 2x2 cells), the number of overlapping cells between adjacent blocks (default option: block size/2), the number of orientation histogram bins (default value: 9) and finally, the selection of using signed orientation or not (the default option is to use unsigned orientation from 0-180°).

Apart from the produced feature vector, it is possible to produce the visualisation of the extracted features for presentation purposes. Figures 3-6 and 3-7 below present the visualisation of HOG features. The visualisation is a grid of uniformly spaced rose plots. The grid dimensions (number of rose plots) are determined by the defined cell size and the image resolution. Each of the rose plots shows the distribution of gradient orientations within a HOG cell. The length of each petal is scaled to indicate the contribution each orientation makes to the cell histogram (magnitude of the vectors, where stronger gradients have a bigger impact on the histogram). The plot displays the edge directions, which are normal to the gradient directions. When viewing the plot with the edge directions it is possible to better understand the shape and contours encoded by HOG. Figure 3-6 presents a grid of such rose plots in an image area, while Figure 3-7 is an individual rose plot zoomed in. The number of petals in each of the rose plots is twice the number of bins selected.



**Figure 3-6: HOG visualisation**





**Figure 3-7: Individual rose plot**

In order to perform some initial testing, the default parameters were used for the extraction of HOG features. Since the default settings were used for pedestrian detection for the first time, the possibility of tweaking the parameters to improve performance was explored. This was necessary, since the nature of this application (ROI generation based on horizontality) may require a different parameter set.

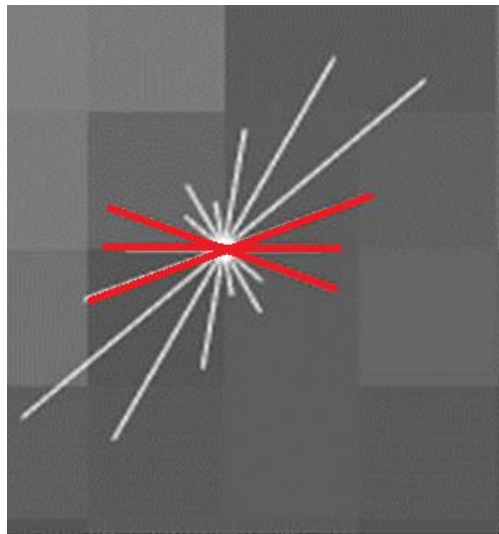
In the end, the default extraction parameters were selected due to the balance between descriptive ability and computational cost. The detection system has been tested with modified parameters as well.

- A larger cell size reduced the size of the feature vector. This may result into lower computational load, but the loss of significant descriptive information is a serious side effect.
- Removing the overlap between blocks completely also reduced the descriptive information significantly, rendering the info useless.
- Increasing the number of orientation bins from 9 to 12 did not offer any performance advantages, with the end result not significantly different.
- Using signed orientation makes no difference, as the direction of the edges is of no interest.

### **Feature vector processing**

The next step is to process the resulting vector in a way that produces the desired feature. From the HOG feature vector, the useful information required is the gradient values that indicate areas with strong horizontality. A process is implemented to determine which areas of an image contain long horizontal elements (in vehicles, such parts would be bumpers, windshields, roof etc.).

Processing takes place at the cell level. The values contained in the feature vector are re-arranged and assigned to the correct cell/block. That way, the HOG values are formatted from a 1D vector into an array that resembles the original image. The largest value (magnitude) corresponds to the strongest gradient. This information is isolated and selected to represent the specific cell. The cells exhibiting strong horizontality are this way identified. A cell is considered to exhibit strong horizontality is one where the strongest gradient is horizontal or close to horizontal ( $\pm 20^\circ$ ), as in Figure 3-8:



**Figure 3-8: Cell with strong horizontal elements**

An example of the produced output, a logical matrix where white (value = 1) indicates strong horizontality can be seen in Figure 3-9 below, beside the original sample image:



**Figure 3-9: Example of cell processing**

It is expected that this process will produce several erroneous results, due to other objects in the background exhibiting similar characteristics. Identifying the correct objects and extracting them from background noise is required before the classification process (Hypothesis Verification).

### **Clustering / segmentation**

The output of the previous step is a logical matrix (containing 0 and 1) used as input in a clustering process to form the objects that exhibit strong horizontal edges. A value of 1 indicates a cell of pixels in which the horizontal gradient is the strongest ( $\pm 20^\circ$ ). A large number of cells in sequence with said value indicate areas of the image with strong horizontality. The length of each horizontal segment is measured and only the continuous segments are retained.

The method used to cluster and label the areas is called connected components (CC) (He et al., 2017). CC groups and labels areas based on connectivity between the elements of the matrix. It is able to function with both images (which are represented by matrices containing each pixel's intensity value) and binary matrices the ones produced by processing the feature vector.

Other methods, such as superpixel segmentation (Achanta et al., 2010) and K-means clustering for image segmentation (Dhanachandra et al., 2015) were also implemented in an attempt to find the optimal clustering method.

Superpixel segmentation groups neighbouring pixels in an image by similarity in colour or gray scales. The advantages of grouping pixels into larger areas in an image are:

- To compute features in more meaningful regions compared to computing features for individual pixels
- To reduce the number of inputs for any subsequent process (i.e. classification)

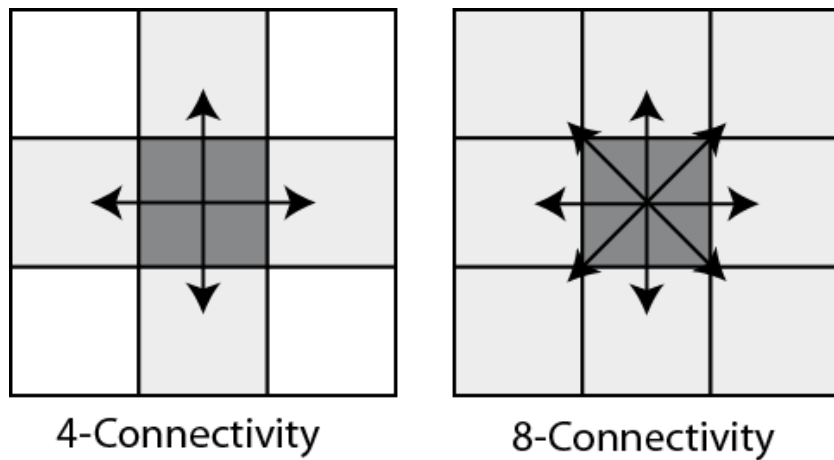
Using Superpixels did not produce any valuable results in terms of clustering into interesting areas. The reason behind this failure is that it is practically impossible to define the number of interesting objects for every image containing vehicles and also, because the size of vehicles in every image varies, depending on the distance from the camera.

K-means clustering is a popular type of unsupervised learning, where data points in a dataset are clustered together based on feature similarity. The algorithm works iteratively to assign each data point to one of K clusters based on the features provided. K, the number of clusters is defined beforehand, based on the number of groups desired.

The use of k-means clustering failed for the same reasons as superpixel clustering. It is very difficult to define the number of k interesting objects in an image from the start given that every image is different, containing one or more vehicles. Applying k-means to the original image to distinguish objects also proved impossible due to the image pixels exhibiting little variation in intensity levels.

The segmentation method producing the most promising results was the Connected Components (CC) method. CC scans the matrix (or image) and checks if the value of a matrix cell is shared by its neighbours as well (intensity values for pixels in an image, respectively). If yes, the cells are labelled as belonging to the same cluster.

For a 2D matrix, connectivity scans can be 4-way or 8-way (Fisher et al., 2003b), as can be seen in Figure 3-10 below:



**Figure 3-10: 4 and 8-way connectivity**

In 4-way connectivity, pixels are connected if their edges touch. The pixels are part of the same object if they are both on and connected along the horizontal or vertical direction. In 8-way connectivity, pixels can be connected if their edges or corners touch. They are part of the same object if they are on and connected along through any of the horizontal, vertical or diagonal directions.

Both configurations were considered during testing, though no meaningful differences in performance were observed. In the end, 4-way connectivity was used in the segmentation/labelling process.

The output of the CC clustering is matrix of region labels. The properties of all labelled regions, such as area size, centroid, find the region's extreme points, orientation and perimeter can be measured. At this stage, it is also possible to draw a bounding box around each area as a way to distinguish between different regions.

Finally, it is useful to remove all objects (connected components) that measure less than a specified size from the output. In that way, very small objects that cannot be part of a vehicle in the image are removed and are not considered for further processing. All objects measuring less than 50 pixels in size were removed from the output.

Figure 3-11 below presents the output of CC clustering, with colour-coded clusters in RGB for visualisation purposes. The resulting output is a very dense image, where discerning individual objects is extremely difficult. The object of interest in this case is the vehicle in the centre of the image. To improve object separation, additional processing is required:



**Figure 3-11: CC clustering output**

### **Vehicle – environment separation**

An enhancing process is required in order to reduce the number of errors in the initial detection of horizontal elements and the segmentation of objects.

Its aim is to improve object-environment separation. The problem with the whole process up to this point is that sometimes, the object (vehicle) is not effectively separated from its surrounding environment. The reasons for this are several:

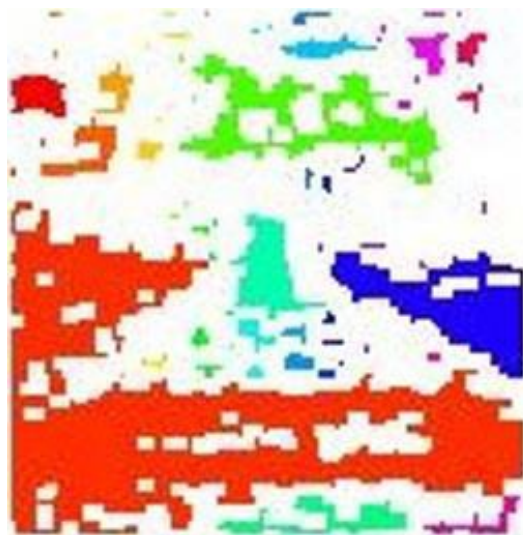
- Another vehicle in close proximity, creating long horizontal edges that cannot be separated
- Vehicle's horizontal elements aligned with detail in the background such as signs, horizontal lines on the road, horizon line
- Random error, due to the quality of the input image

One solution to the above problems would be to use a higher resolution image, so that more fine detail would be retained. However, that would introduce performance

issues as the computational cost to process the images would be significantly higher. That would lead to increased run time as well.

To improve object separation, an averaging operation is introduced. The 2D 3x3 filter  $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$  is applied, calculating averages along horizontal lines and then a threshold is applied to retain only the strongest horizontal elements in every image area. When the values in the resulting matrix are clustered again, the improvement in object segmentation is significant.

Figure 3-12 below presents the improved output of the separation process, after re-clustering to form new objects. The improvement from the previous step (in Figure 3-11) is evident in Figure 3-12 below:



**Figure 3-12: Improved object separation output**

Similarly to the previous step where another connectivity type was used, another configuration using a larger 5x5 filter was tested. Between the different sized filters, the smaller 3x3 one proved to be more effective as the larger 5x5 filter led to loss of useful information. After this step, the vehicle in the centre of the image is more clearly discernible.

To further enhance the output and ensure proper object/background separation, additional visual cues were incorporated into the ROI generation algorithm. Out of

the possible options for enhancing the result in appearance-based approaches, some had to be excluded due to the constraints imposed by the use of the specific input data.

- The option of using the useful colour cue had to be excluded due to the image data being grayscale.
- Using shadows as an additional visual cue would be ineffective, due to the quality of the input images (with intensity values close in many image areas where the vehicle was located) and its sensitivity to illumination conditions.
- Vehicle back lights could not be considered due to the images being grayscale
- Texture-based segmentation was also ineffective, incapable of distinguishing between vehicle and other objects in the background.
- Advanced symmetry detection methods would be not as cost-effective as simpler image processing methods, in addition to symmetry being sensitive to illumination, occlusion and other symmetric objects in the image.

The Sobel (Sobel, 1990) and Canny (Canny, 1986) edge detection filters were implemented in separate occasions as additional methods to separate a vehicle from its surrounding environment. The filters were applied on the original downscaled images. The output of the edge filters is an output binary image (logical array of 0-1 values) of the same size as the input image matrix, where the presence of an edge is signified with a 1, with 0 elsewhere in the image. In the end, the Canny filter was selected to improve vehicle-environment separation.

The logical array (edge filter output) is superimposed with the output of the clustering process. The end result is an array containing labelled areas that are considered as Regions of Interest (ROIs).

### **Bounding box generation**

Objects and areas of interest are highlighted in the image by drawing a rectangular box (known as a bounding box) around them. The bounding box drawn is the smallest rectangle containing the labelled region that is used as input for



classification. Along with the bounding box, useful information such as area size, coordinates of the bounding box, region centre etc. are retained.

Figures 3-13 and 3-14 below present the generated bounding box for the sample image, before and after small object removal (objects containing less than 50 pixels). All generated bounding boxes that do not resemble vehicles (due to small/very large size, irregular aspect ratio) should be removed, so that the number of candidate locations is as low as possible and therefore making the classification process faster:



**Figure 3-13: Original bounding boxes**



**Figure 3-14: Bounding boxes after small object removal**

### 3.2.2 Hypothesis Verification

For the second phase of the detection process, which involves verifying the existence of an object in an image and classifying it as a vehicle, a pedestrian or an object, a SVM classifier is employed. The sub-images generated by the HG phase (bounding boxes – ROIs) are utilised in this second stage of the detection algorithm.

Before the SVM is able to distinguish between vehicles and non-vehicles in real-time data streams, it is necessary to train it to identify what visual characteristics are distinct to vehicles. The training process utilises a dataset of positive (vehicle) and negative (non-vehicle) images, from which the describing elements of objects (feature vectors) are extracted. The image feature extraction process is presented in Figure 3-15 below:



**Figure 3-15: Image feature extraction process**

The samples images contained in the training dataset are processed before the feature extraction. A histogram equalisation process takes place first; to mitigate the effect of differing lightning and contrast conditions have in performance. Next, the images are scaled to a fixed size resolution, common for all, so that all extracted feature vectors are equal in size. This is a necessary process, as the SVM classifiers can only be trained with images of the same size. After that, the HOG extraction process takes place, using parameters determined after experimentation with HOG extraction settings. Finally, the resulting feature vectors are stored in a feature file, ready to be used for classifier training.

The same process takes place for input data in real-time conditions. Using the extracted feature vectors, sub-images are classified into vehicles or non-vehicles. A successful detection, where only the vehicle is verified and other ROIs are discarded, can be seen in Figure 3-16 below (the sample image in Figure 3-2 was used):



**Figure 3-16: Example of successful vehicle detection**

As mentioned before, SVMs rely on a kernel function to perform classification. There are several kernel functions that may be used and there is only one restriction as to which one to use. The kernel must satisfy the Mercer condition (it must be a continuous symmetrical kernel of a positive integral operator).

According to the Mercer theorem in Mathematics, if  $K: [a, b]^2 \rightarrow R$  (where  $K$  is the kernel function) is a symmetric, non-negative definite function, then there exists a countable sequence of functions  $\{\varphi_i\}_{i \in N}$  (mapping function  $\varphi$ , as in section 2.3.2 on SVM) and a sequence of positive real numbers  $\{\lambda_i\}_{i \in N}$  such that,

$$K(s, t) = \sum_{i=1}^{\infty} \lambda_i \varphi_i(s) \varphi_i(t) \quad (3.2)$$

Cortes and Vapnik, (1995) has shown that if the kernel function  $k$  is positive definite, the existence of  $\varphi$  is guaranteed. This allows for the kernel to be used instead of calculating the mapping function  $\varphi$ , as mentioned in section 2.3.2.

Some initial testing was required in order to determine which kernel is more appropriate for the particular task. Dalal and Triggs, 2005, in their work on human detection, experimented with two kernels: a linear and a Fine Gaussian kernel and determined that the slightly improved performance for the F. Gaussian SVM came at the expense of much higher computational cost and therefore longer run-time.

200 vehicle images, a small sample size, were used to perform the kernel and parameter test. HOG features in different resolutions (64x64, 128x128, and 256x256) were extracted using the default parameters in MATLAB. During the training process, a portion of the data (20%) is used for validating the results.

In the end, the linear kernel and the Fine Gaussian kernel were selected for training the SVM classifier with the complete training dataset. Maximum classification accuracy was 88.7% for the F.Gaussian kernel and 74.6% for the linear one. The trade-off between them was a significant increase in training time.

The SVM classifier was trained on the training dataset on 2 different resolutions (128x128 and 256x256) to determine whether there are any differences in classification performance.

### **3.3 Vehicle detection II – Convolutional Neural Network (CNN)**

This second method focuses on the detection of vehicles using a variant of CNN, called Faster R-CNN (also known as Region-Based CNN). Faster R-CNN is detection and classification method that combines a CNN and a Region Proposal Network (RPN), which is a separate convolutional network designed to generate potential Regions of Interest (ROI) for objects. Instead of using a pre-trained multi-layer network, as is common in most classifiers, the Neural Network is trained from scratch, using own collected image data. Vehicle instances were manually annotated and used to train the network to detect the vehicle object class. The goal is to explore the feasibility of using a CNN with few convolutional layers as a vehicle detector, using a limited amount of collected data instead of readily available datasets. The performance of the examined network models will offer insights as to what kind of conditions such networks can operate in, given there are data size and image quality constraints and whether they can be used in safety critical applications. The performance in terms of detection accuracy and run-time will be compared to the traditional HOG+SVM method proposed in the above section.

For the purposes of this project, and to examine what kind of optimisations are required to produce an efficient CNN-based vehicle detector, six different network

models are examined. All of them are based on the Faster R-CNN algorithm, which was selected for its high performance compared to other methods (Huang et al., 2017), but are modified so as to determine which network topology and which parameters are the most efficient for detecting vehicles in the dataset. The modifications included are changes to the RPN network, introducing batch normalisation, increasing the learn rate of the network and increasing the network depth, all of which affect detection performance. The examined networks were developed, trained and tested using Matlab R2018a.

All the basic building blocks of a CNN are present (Convolution layer, non-linear conversion layer, pooling layer, classification layer) with the addition of dedicated normalisation layer in one of the models. Dropout layers are not utilised and the task of regularisation and avoiding overfitting is handled by dedicated normalisation layers (where applicable), L2 regularisation and the size of the model itself.

The CNN vehicle detectors are all trained from scratch, without the use of pre-trained networks (e.g. Alexnet, Google Net etc.) and their pre-calculated weight values. Initially, all networks are trained using a base 5,000 image samples and additional training data (obtained through data augmentation) are added in 5,000 image batches until a maximum of 20,000 training images. Compared to pre-trained networks trained on large databases (e.g. ImageNet containing 374,000 images for the vehicle class), this is a small amount of data. The process of acquiring, processing and augmenting the image data for the CNN-based detector is detailed in Chapter 4.

Due to the number of available training images, the use of a deep network structure would lead to overfitting problems. The model would be well adjusted to the training dataset (essentially memorising the dataset features) but would not generalise well with new data. With that in mind, it was essential that the networks used would be relatively shallow (compared to pre-trained classification networks such as ResNet, GoogleNet etc.) in addition to using regularisation as an additional measure to reduce potential overfitting problems.

The six convolutional networks are the following:

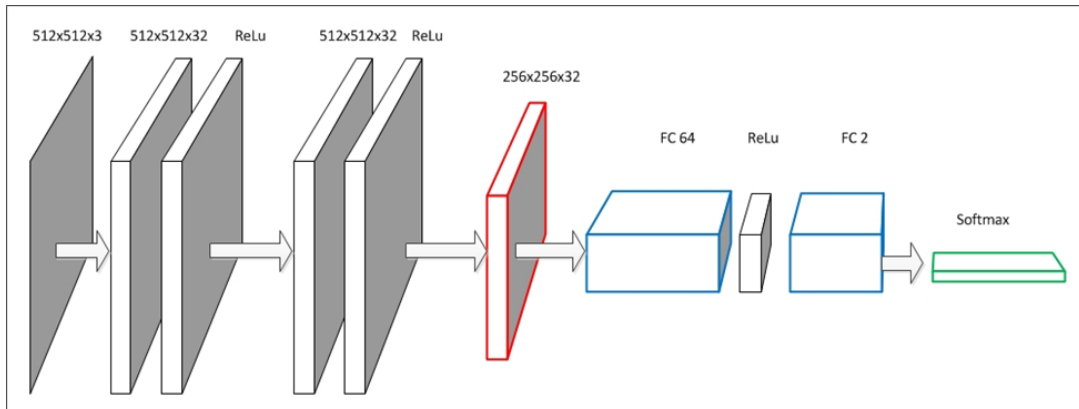
Layer no.	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6
1	Input layer	Input layer	Input layer	Input layer	Input layer	Input layer
2	Conv layer	Conv layer	Conv layer	Conv layer	Conv layer	Conv layer
3	ReLU	ReLU	Batch normalisation	ReLU	ReLU	ReLU
4	Conv layer	Conv layer	ReLU	Conv layer	Conv layer	Conv layer
5	ReLU	ReLU	Conv layer	ReLU	ReLU	ReLU
6	Max pooling	Conv layer	Batch normalisation	Max pooling	Conv layer	Conv layer
7	Fully Connected layer	ReLU	ReLU	Fully Connected layer	ReLU	ReLU
8	ReLU	Max pooling	Max pooling	ReLU	Max pooling	Max pooling
9	Fully Connected layer	Fully Connected layer	Fully Connected layer	Fully Connected layer	Fully Connected layer	Conv layer
10	Softmax	ReLU	ReLU	Softmax	ReLU	ReLU
11	Classification layer	Fully Connected layer	Fully Connected layer	Classification layer	Fully Connected layer	Max pooling
12		Softmax	Softmax		Softmax	Fully Connected layer
13		Classification layer	Classification layer		Classification layer	ReLU
14						Fully Connected layer
15						Softmax
16						Classification layer

Figure 3-17: Summary of developed CNNs

**Model 1: A reference network structure using the following 11 layers:**

- i. Input layer (32x32x3 images)
- ii. Convolution layer (32 3x3 convolutions with stride [1 1] and padding [1 1 1 1])
- iii. ReLU (Rectified Linear Unit)
- iv. Convolution layer (32 3x3 convolutions with stride [1 1] and padding [1 1 1 1])
- v. ReLU
- vi. Max pooling layer (3x3 max pooling with stride [2 2] and padding [0 0 0 0])
- vii. Fully Connected layer 64 fully connected layer
- viii. ReLU
- ix. Fully Connected layer 2 fully connected layer
- x. Softmax layer
- xi. Classification output Crossentropyex

The architecture of the reference network can be seen in Figure 3-18 below:



**Figure 3-18: Reference network architecture**

Additional details regarding each of the network's layers are given below:

**Image input layer:** The image input layer introduced the image to the neural network and is able to normalise the image data before any further processing takes place. The original input image is segmented into smaller parts and this is the layer where the size of these image patches is set. For object detection tasks, this size is set as approximately the size of the smallest object that needs to be detected. Since vehicle sizes in an image can vary and also depend on the distance from the ego-vehicle, a relatively low value (32x32 pixels) is used. Zero-centre normalisation (Karpathy and Li, 2019a) is applied in this layer, so that the mean of the image data lies on zero (mathematically, this is achieved by calculating the mean in the data, and subtracting each data item with that mean). Regarding image patch size, it is possible to reduce the minimum size of the patch, that would however, increase the number of sub-images that need to be processed by the CNN and so lead to an increase in processing time. In purely classification tasks, this size is set as the fixed size of a training image.

**Convolution layer:** Initially, the size and number of filters is set. This determines the size of the produced feature maps as well as the depth of the convolution layer. In

this case, the layer has a depth of 32 (32 different feature maps produced), while the size of the convolution matrix is set to 3x3.

In both convolution layers in this CNN, the step size for traversing the input image (stride) is set to 1 pixel in both x and y direction, while additionally, 1 row of padding is added to all sides of the input matrix (top, bottom, left, right sides). Stride is the number of pixels the filter shifts over the input image. Its value affects the size of the resulting feature map (and the encoded features), with a smaller stride resulting in a larger feature map (and more useful information) and a larger stride in a smaller feature map but faster processing.

Adding padding to the image increases its size and gives the opportunity to border pixels to better interact with the filters, as they can now be at the centre of the filter. This results in more features to be detected by the filter and an output feature map that has the same shape as the input image.

It is possible to manually set additional parameters for each of the convolution layers such as weight learn factor and bias learn factor, though in this case the final trained network determines weights and biases for each layer based on the training dataset without any manual initialisation.

**ReLU layer:** The function used to introduce non-linearity to the networks is the ReLU function (used widely in CNNs) which replaces all negative pixel values in the feature map produced by the convolutional layer with 0:

$$f(x) = \max(0, x) \tag{2.20}$$

**Pooling layer:** The options available for the pooling layer are similar to those for the convolution layers. The size of the pooling window is determined along with step size (stride) and possible padding around the convoluted feature maps.

In this case, a max pooling layer is used to downsample the output of the convolutional layers and reduce the number of connections to the following layers. The pooling layer comes after 2 convolution + ReLU operations and uses a 3x3 size window with a stride of 2 across the x and y directions. Note that the stride size is smaller than the actual pooling window. That indicates that there is some overlap



between pooling regions. No padding is added to input borders in any direction as it is not required.

**Fully Connected Layer:** The Fully Connected layer is a traditional Neural Network layer, in that it uses an activation function to produce its output. Every neuron in this layer is connected to every neuron of the previous layer and each of these connections has an associated weight and bias factor.

Here, it is possible to define the number of the output neurons and also initialise weight, bias and learn factor bias values. Usually, the network initialises those values by assigning random values for the first training example and then auto-adjusts them based on the error of the output probability for the expected object class.

The output size of the first fully connected layer is 64 neurons while for the second layer of this type, the network needs to produce a number of outputs equal to the number of object classes and background. Since, this a single-class vehicle detector, the number of output neurons is 2: one for the vehicle object class and one for the background.

**Softmax layer:** The Softmax loss function is used as the classifier (others, such as SVM can be used) to predict a class out of K mutually exclusive classes. The output softmax function is:

$$y_r(x) = \frac{\exp(a_r(x))}{\sum_{j=1}^k \exp(a_j(x))} \quad (3.3)$$

Where  $0 \leq y_r \leq 1$ ,  $\sum_{j=1}^k y_j = 1$  and  $a_r$  is the conditional probability of the sample class  $r$ .

**Classification layer:** The classification layer is the output layer of the neural network and takes its name from the loss function used for training the network and calibrating the weight and bias values.

This layer uses the cross entropy loss function.

**Model 2: The second network model is a 13 layer structure, adding an additional set of Conv + ReLU layers on the reference model and increasing the number of convolution kernels to 64.**

- |       |                              |   |
|-------|------------------------------|---|
| i.    | Input layer                  | (32x32x3 images)  |
| ii.   | Convolution layer            | (64 3x3 convolutions with stride [1 1] and padding [1 1 1 1]) |
| iii.  | ReLU (Rectified Linear Unit) |   |
| iv.   | Convolution layer            | (64 3x3 convolutions with stride [1 1] and padding [1 1 1 1]) |
| v.    | ReLU                         |   |
| vi.   | Convolution layer            | (64 3x3 convolutions with stride [1 1] and padding [1 1 1 1]) |
| vii.  | ReLU                         |   |
| viii. | Max pooling layer            | (3x3 max pooling with stride [2 2] and padding [0 0 0 0])     |
| ix.   | Fully Connected layer        | 64 fully connected layer                                      |
| x.    | ReLU                         |   |
| xi.   | Fully Connected layer        | 2 fully connected layer                                       |
| xii.  | Softmax layer                |   |
| xiii. | Classification output        | Crossentropyex  |

Adding additional convolution layers (which is the core building block of a CNN) increases the depth size and the generalisation ability of the network. The network is able to recognise more complex image features and therefore better understand the relationship between the input image and the class it belongs to. However, there is a trade-off. Increasing the network depth results in a network that is difficult and slow to train and test. Additionally, a very deep network is more prone to overfitting on the dataset it was trained on. A balance between network size and runtime speed is required. The increased number of kernels (64) used in this network model means a larger number of feature maps will be produced from the convolution process, resulting in additional salient features that benefit the detection/classification process.

**Model 3: A network structure containing the original 11 layers, with the addition of batch normalisation layers to reduce network sensitivity during initialisation.**

- |       |                           |   |
|-------|---------------------------|---|
| i.    | Input layer               | (32x32x3 images)  |
| ii.   | Convolution layer         | (32 3x3 convolutions with stride [1 1] and padding [1 1 1 1]) |
| iii.  | Batch normalisation layer |   |
| iv.   | ReLU                      | (Rectified Linear Unit)                                       |
| v.    | Convolution layer         | (32 3x3 convolutions with stride [1 1] and padding [1 1 1 1]) |
| vi.   | Batch normalisation layer |   |
| vii.  | ReLU                      |   |
| viii. | Max pooling layer         | (3x3 max pooling with stride [2 2] and padding [0 0 0 0])     |
| ix.   | Fully Connected layer     | 64 fully connected layer                                      |
| x.    | ReLU                      |   |
| xi.   | Fully Connected layer     | 2 fully connected layer                                       |
| xii.  | Softmax layer             |   |
| xiii. | Classification output     | Crossentropyex  |

A batch normalisation layer normalises the activations of each input channel across the mini-batches of images used for training, speeding up the training process and reducing the sensitivity to network initialisation. The output of the convolution layer is normalised by subtracting the mini-batch mean and dividing by the mini-batch standard deviation. The effect of the batch normalisation layer on performance will be examined.

**Model 4: A reference structure containing the original 11 layers, with a modification in the RPN producing a maximum of 1000 ROIs per image.**

- |       |                       |   |
|-------|-----------------------|---|
| i.    | Input layer           | (32x32x3 images)  |
| ii.   | Convolution layer     | (32 3x3 convolutions with stride [1 1] and padding [1 1 1 1]) |
| iii.  | ReLU                  | (Rectified Linear Unit)                                       |
| iv.   | Convolution layer     | (32 3x3 convolutions with stride [1 1] and padding [1 1 1 1]) |
| v.    | ReLU                  |   |
| vi.   | Max pooling layer     | (3x3 max pooling with stride [2 2] and padding [0 0 0 0])     |
| vii.  | Fully Connected layer | 64 fully connected layer                                      |
| viii. | ReLU                  |   |
| ix.   | Fully Connected layer | 2 fully connected layer                                       |
| x.    | Softmax layer         |   |
| xi.   | Classification output | Crossentropyex  |

The Region Proposal Network (RPN) of Faster R-CNN network uses a maximum of 2000 proposals to generate the training samples for the network. By reducing this number, the aim is to speed up training and testing. It is expected that a small impact on detection accuracy will occur.

**Model 5: A network structure, based on the second model (13 layers and 64 convolution kernels) with a modified learning rate for each training step.**

- |       |                       |   |
|-------|-----------------------|---|
| i.    | Input layer           | (32x32x3 images)  |
| ii.   | Convolution layer     | (64 3x3 convolutions with stride [1 1] and padding [1 1 1 1]) |
| iii.  | ReLU                  | (Rectified Linear Unit)                                       |
| iv.   | Convolution layer     | (64 3x3 convolutions with stride [1 1] and padding [1 1 1 1]) |
| v.    | ReLU                  |   |
| vi.   | Convolution layer     | (64 3x3 convolutions with stride [1 1] and padding [1 1 1 1]) |
| vii.  | ReLU                  |   |
| viii. | Max pooling layer     | (3x3 max pooling with stride [2 2] and padding [0 0 0 0])     |
| ix.   | Fully Connected layer | 64 fully connected layer                                      |
| x.    | ReLU                  |   |
| xi.   | Fully Connected layer | 2 fully connected layer                                       |
| xii.  | Softmax layer         |   |
| xiii. | Classification output | Crossentropyex  |

For this model, the learning rates for each of the training stages of the CNN (training and fine-tuning the Region Proposal Network – RPN and the CNN) have been modified to have the network parameters change faster compared to the other configurations. The rationale behind this modification is to explore the possibility of the network converging faster at an optimal solution, and even achieving a superior result after the end of the training epochs.

The learning rate for the first two training stages is set at  $5 \times 10^{-4}$  (0.0005) and for the fine-tuning stages at  $1 \times 10^{-5}$  (0.00001). The values have been selected after experimenting with different rates, where even higher values would make it impossible for the CNN to finish training and converge to a solution.

**Model 6: A deeper network structure, where an additional Conv + ReLU set of layers is added, and a variable number of filters is used.**

- i. Input layer (32x32x3 images)
- ii. Convolution layer (32 3x3 convolutions with stride [1 1] and padding [1 1 1 1])
- iii. ReLU (Rectified Linear Unit)
- iv. Convolution layer (32 3x3 convolutions with stride [1 1] and padding [1 1 1 1])
- v. ReLU
- vi. Convolution layer (32 3x3 convolutions with stride [1 1] and padding [1 1 1 1])
- vii. ReLU
- viii. Max pooling layer (3x3 max pooling with stride [2 2] and padding [0 0 0 0])
- ix. Convolution layer (64 3x3 convolutions with stride [1 1] and padding [1 1 1 1])
- x. ReLU
- xi. Max pooling layer (3x3 max pooling with stride [2 2] and padding [0 0 0 0])
- xii. Fully Connected layer 64 fully connected layer
- xiii. ReLU
- xiv. Fully Connected layer 2 fully connected layer
- xv. Softmax layer
- xvi. Classification output Crossentropy

For this model, the depth of the network has been increased with the addition of extra layers. It is expected that there will be some performance gain in terms of accuracy, at the expense of training and testing time. To offset the increased processing time, the number of convolution kernels has been reduced to 32 for the first 3 convolution layers, with the last convolution layer retaining the full 64 kernels.

## **Region Proposal Network (RPN)**

The Region Proposal Network (RPN) is common across all examined models as part of the Faster R-CNN algorithm (Ren et al., 2015). It is tasked with generating the Regions of Interest (ROIs) for the CNN, without the need for an external function. Essentially, it operates as a shallow NN accepting the feature map produced by the main network as input. ROIs are produced in various scales on the feature map before a classification and a regression layer assign probability scores to each ROI and generate bounding boxes for them respectively. The output of the RPN is returned to the main CNN for classification.

## **Training and testing the CNN detector**

The vehicle detector training process can be separated in 4 distinct stages. The first two train the Region Proposal Network (RPN) while the last two combine the output of the first two stages and fine-tune the network.

Stage 1: Training a Region Proposal Network (RPN)

Stage 2: Training a Fast RCNN network using the RPN from stage 1

Stage 3: Re-training RPN using weight sharing with Fast RCNN

Stage 4: Re-training Fast RCNN using updated RPN

The parameters used to train the different models are the following:

**Max epochs:** the number of epochs used for training. An epoch is a full pass of the training algorithm over the training set. An epoch is divided into iterations, which are the steps taken in the gradient descent algorithm towards minimising the loss function using a mini batch. The number of epochs used for every training stage is 10.

**Minibatch size:** is a subset of the training set that is used to evaluate the gradient of the loss function and update the weights. It is used in each training iteration. The size is set 128 for the first and third training stages and 64 for stages two and four (the

reference model uses 128 images for all four training stages – the change was necessary to avoid running into memory problems for the larger models).

**Initial Learning Rate:** The learning rate determines how quickly the network parameters change. For the first two training stages, the learning rate is  $1e^{-5}$  (0.00001) while for the final two,  $1e^{-6}$  (0.000001). The learning rate is lower in fine-tuning stages so that smaller adjustments are made.

**L2 regularisation:** Regularisation (or weight decay) is added to reduce overfitting. A regularisation term for the weights is added to the loss function  $E(\theta)$ . The loss function with the added term takes the form:

$$E_R(\theta) = E(\theta) + \lambda\Omega(w) \quad (3.4)$$

Where  $w$  is the weight vector,  $\lambda$  is the regularisation coefficient and the regularisation function  $\Omega(w)$  is  $\Omega(w) = \frac{1}{2}w^T w$ .

The L2 regularisation coefficient is  $1e^{-4}$  (0.0001).

The detection parameters are set as follows:

**Positive overlap range:** Bounding box overlap ratio for positive training samples. The anchor boxes that are generated by the RPN for an image have varying levels of overlap with the ground truth objects in the particular image. Region proposals that overlap with ground truth bounding boxes within the specified range are used as positive training samples. Other boxes with lower values can be used as negative samples or even discarded completely if they fall between the ranges specified for positive/negative samples. This process is necessary for the detector to learn what a correct detection is and what is not. The choice to have ‘neutral’ proposals that do not contribute to the training process (proposals between the positive/negative ranges) and disregard them was due to the fact that a large number of negative samples exist already in an image and it is not necessary to process them all. The value used for positive samples for the vehicle detector was [0.6-1]. The overlap ratio used is defined as union (area of intersection between two bounding boxes divided by the area of the union of the two).



**Negative overlap range:** Bounding box overlap ratio for negative training samples. Similar to the positive range, values range from 0-1 and the value used was [0-0.3].

**Number of strongest regions:** Maximum number of strongest region proposals to use for generating training samples. The default value for this option is 2000 proposals. Reducing this value leads to reduced processing time at the cost of training accuracy.

**Smallest image dimension:** This option sets the smallest dimension (either width or height) when it is necessary to resize training images. Resizing images into smaller sizes reduces computational cost. For all the models tested, training images retain their original size (512x512).

To summarise, the training parameters used for training and testing the CNN detector are as follows:

- Original image size is 512x512. The image is segmented into smaller parts for detection purposes, with the segment size set at 32x32 pixels.
- The convolution matrix is 3x3 in size with 1 pixel stride. The number of convolution kernels is 32, except for the models, where the number of convolution kernels is 64.
- The pooling matrix has a size of 3x3 and stride 2 across both dimensions.
- The output size of the first Fully Connected layer is 64, while the output for the second one is 2, since there are only 2 object classes (vehicle and background).
- The network is trained for 10 epochs for each of the Faster R-CNN training stages (RPN, Neural Network, fine-tuning). The number of training epochs was selected to be 10 and not higher, to avoid overfitting issues in the trained model. When a model is trained many for many cycles, it tends to over-fit to the training data and loses its generalisation ability. As a rule, training a CNN model should stop when the error rate in the validation data is minimised. After that point, the model starts to over-fit.
- The initial learning rate is set at  $1e-5$  and for the fine-tuning process at  $1e-6$ .

- The Region Proposal Network (RPN) generates 2,000 candidate regions for each input image by default.

The performance of the two vehicle detectors will be assessed in terms of accuracy and precision in a complex dataset, containing vehicle instances of various sizes in different types of environment. The desired outcome is a vehicle detector that is able to detect all vehicle instances in a set of images or video while at the same time producing the lowest number of False Positive (FP) detections possible. The run-time performance of the most accurate detector will also be assessed, in order to determine whether it is suitable for application in the real world.

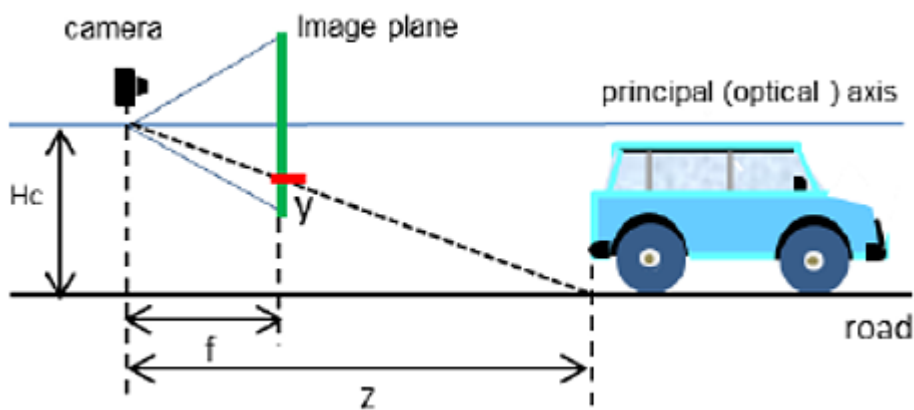
### **3.4 Range estimation using perspective geometry**

This section describes the method used to estimate distance from a vehicle ahead. The method uses simple perspective geometry using the monocular camera's parameters to estimate the distance to the centre of a bounding box generated by the vehicle detector. Because of the simple geometric calculation, the bounding box is considered to be on the surface of the road. Accurate computation of a location in 3D space is not possible and would require a stereo camera system or an active sensor (radar, LIDAR). Additionally, there is no consideration for error compensation so the system cannot be considered accurate for use in safety critical application. This merely serves as a test to determine the inaccuracies of the measurement and determine whether the developed detector produces workable results. The method is based on the work in (Stein et al., 2003).

Figure 3-18 is a schematic diagram of the imaging geometry. The camera is mounted on a vehicle at height  $H_c$ . The rear of the vehicle is at a distance  $Z$  from the camera while  $f$  is the focal length of the camera. The point of contact between the vehicle and the road projects onto the image plane at a position  $y$ . The focal length and point  $y$  are typically in pixels and are not drawn to scale here. The distance  $Z$  is derived directly from the similarity of triangles:

$$Z = \frac{f H_c}{y} \quad (3.5)$$

All required values are available: the height  $H_c$ , focal length  $f$  and principal point of the camera are known and determined through camera calibration. The pitch of the camera that specifies the angle (tilt) from the horizontal position and is a potential source of error is also determined during the calibration process. The bounding box and the location of the centre of the bottom side are produced by the vehicle detector.



**Figure 3-19: Range estimation based on bottom of b. box** Source: Christiansen et al. (2018)

This simple calculation only gives a rough estimate of the distance between the ego and target vehicle, with the limitations being numerous:

- The surface of the road is considered flat
- The distance is calculated to the surface of the road and not the actual vehicle
- The distance is calculated from a single image. An approach using information from a sequence of images would reduce the associated distance error and provide more accurate distance measurements.

## **4 Data collection and pre-processing**

### **4.1 Introduction**

The performance of applications such as camera-based Collision Avoidance rely on robust detection algorithms that have been trained on datasets containing large amounts of quality data which in this case is image samples. The data need to be collected and pre-processed before any part of the detection process takes place.

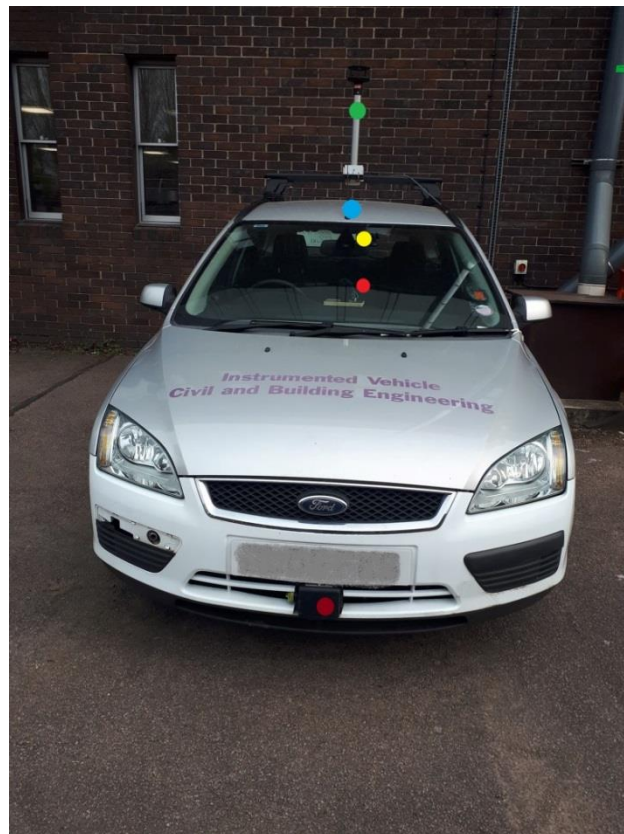
This chapter describes the process of collection, the features and limitations of the datasets used in this project. Pre-processing for each of the detection methods will be described in its own discrete section, as different processes need to be employed.

### **4.2 Data collection**

The most common data collection method for images is recording video sequences from a moving vehicle in the desired environment and then processing the video files to extract the image dataset. All the data utilised in this project were collected using an instrumented vehicle belonging to the School of Architecture, Building and Civil Engineering of Loughborough University. The vehicle is equipped with the following set of sensors:

- A PointGrey Grasshopper3 –U3-41C6NIR-C Near Infrared (NIR) camera (mono sensor, 4.1MP resolutions)
- A Continental ARS 308-2 77GHz long range automotive radar
- A U-blox NEO M8L GNSS and 3D Dead Reckoning system
- A Mobileye 560 forward collision warning and lane departure unit
- A weather station collecting wind speed, humidity and other environmental data
- An Arduino microcontroller connected to the CAN bus exporting information about the status of the vehicle.

The instrumented vehicle along with the installed sensors can be seen below:



● Weather station ● GPS ● Mobileye ● NIR monocular camera ● Automotive radar

Figure 4-1: Loughborough University instrumented vehicle

For this particular project, the NIR camera was used to record the videos required, while the ARS radar was used to provide the readings used as ground truth for range estimation.

The camera records video at 4MP resolution (2048x2048) while the frame rate was set at 15FPS (Frames per second). The recording frame rate was set at 15FPS to reduce the size of the produced video files. The resulting file from this frame rate setting is around 2 GB in size for every 10 minutes of recorded video. Using a higher frame rate (e.g. 30FPS) would result in even larger files that would require additional time to process. The images collected include various vehicle types (private cars, light duty vehicles, buses) and were taken in various operational environments (urban roads, rural roads, motorways) as well as different weather conditions (sunny, overcast, rain) so as to ensure representative samples. The same level of variability is used for both training and testing datasets. The videos were recorded around the town of Loughborough in Leicestershire (UK), the rural areas around it, the M1 motorway connecting London and Leeds in the UK and the city of Nottingham (UK).



**Figure 4-2: Locations of M1 motorway, Loughborough and Nottingham**

In total, several hours' worth of data was recorded, resulting in a large pool of images from which the training and validation datasets were extracted. Samples of the images collected can be seen in Figure 4-3 below:

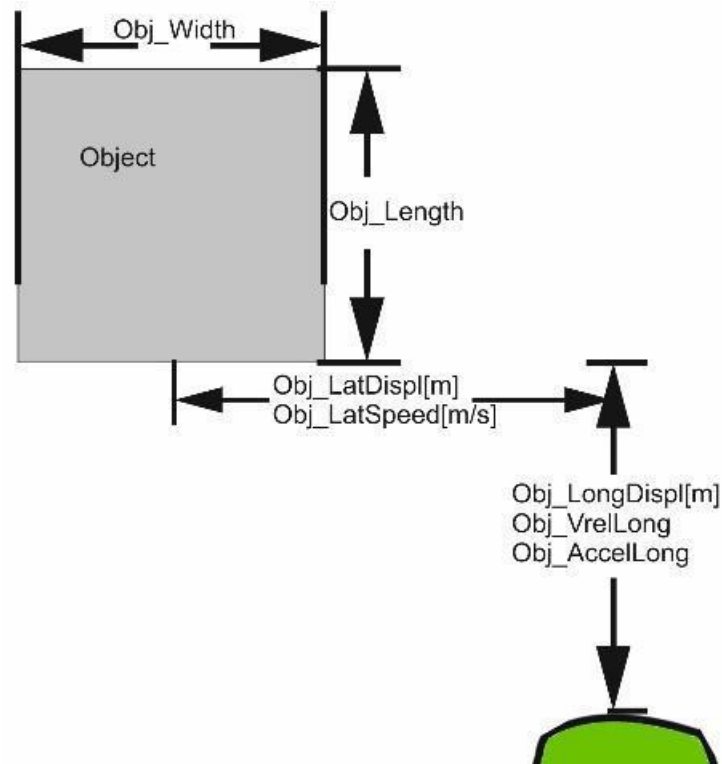


**Figure 4-3: Image dataset samples**

The radar sensor detects objects that reflect radar waves up to 200m (long range, 17° Field of View) and 60m for short range (56° Field of View). The radar is able to distinguish between targets (objects that reflect radio waves) and objects (targets that are detected more than once and tracked).

The variables of interest from the radar output are:

- Number of objects traced in this measurement cycle (*NoOfObjectsY* and *NoOfObjectsTime*)
- Dynamic property of each object (*Obj\_DynPropTime* and *Obj\_DynPropY*) that categorises each object's movement; 0: unclassified, 1: standing, 2: stopped (never moved before) and 4: oncoming.
- Longitudinal displacement of an object (*Obj\_LongDisp [m]*) which will provide the ground truth measurement of the distance between the ego-vehicle and a vehicle ahead.
- Lateral displacement of an object (*Obj\_LatDisp [m]*) which will help with identifying different vehicles in an image. Negative value means that the object is located to the right of the camera, while positive value means that the object is on the left.



**Figure 4-4: Variables measured by the radar sensor**

**Source:** Schnieder (2017)



### **4.3 HOG based detector training data**

This section describes the type of data and processing required for the development of the HOG based detector described extensively in section 3.2 of the Methodology chapter.

#### **4.3.1 Training set**

Due to the large variability of vehicles (size and type) on the road, a large training sample was required in order to train and evaluate the SVM classifier. To train a classifier, the dataset needs to be separated into positive and negative samples. Positive samples need to exhibit enough variation so that the classifier is able to correctly identify all types of vehicles, while negative samples need to cover as many as possible of non-vehicle objects that appear on the road.

The training set consists of 2,135 positive image samples and 2,779 negative samples. Positive samples are appropriately cropped vehicle's front and rear view images, straight on or slightly angled. Vehicles include not only passenger vehicles but also light trucks, lorries and buses. Negative samples include various objects such as traffic signs and lights, poles, lane markings, railings, trees and other vegetation, buildings or parts of buildings and other random objects.

The number of negative samples was augmented by flipping the images horizontally and using the produced symmetric images as additional training samples (for a total of 5,558 images). The total size of the dataset is 7,693 images for a ratio of 1:2.6 positive/negative samples. The dataset was tested and no issues regarding imbalanced dataset were observed. Using this dataset produced slightly better classification results compared to using the more balanced dataset.

Examples of positive and negative training images are shown in Figures 4-5, 4-6 and 4-7 below:



Figure 4-5: Examples of positive training images



Figure 4-6: Examples of negative training samples (a)



**Figure 4-7: Examples of negative training images (b)**

### **4.3.2 Validation set**

An early indication of the performance of the trained SVM classifier is given by performing cross-validation on the training data. However, a smaller independent (not used in training) dataset was used to verify the results of the cross-validation process and ensure the generated classifier performs as expected on unseen data. The size of this smaller dataset is 300 images, with 100 being positive (vehicle) samples and 200 negative (non-vehicle) image samples.

## **4.4 CNN-based detector training data**

In this section, the processing and augmentation of data required for the development of the CNN-based detector (described extensively in section 3.3 of the Methodology chapter) is presented.

The most common issue with training CNNs is the generation of large amounts of data required for this particular task. The amount of data required to effectively train a CNN is highly dependent on the complexity/depth of the network and the task it performs. Structures containing many layers require large amounts of data to perform, as is evident by image classification networks such as VGGNet, which was trained on 1.2 million images (with assigned label for each of the 1000 classes) (Jia Deng et al., 2009).

Simpler tasks such as single-class object detection which is performed here, do not require this amount of data, although a combination of small dataset and deep network would result in overfitting problems (the network memorising features of the training dataset and does not generalise well in unknown data). A balance is required between network complexity and dataset size, especially when not using pre-trained networks where many of the learnable parameters have fixed values and are not trained on new data.

In order to generate the data required for a vehicle detector, it is necessary to manually annotate a large number of images and provide ground truth labels for them. Although the task is simple it is very time-consuming given the vast amount of data required. To reduce annotating time, various tools such as automatic image labellers can be used to automate the process, though not without limitations and with the human presence still being essential to validate the generated labels. Another way to overcome the data collection and annotating issue is the use of a virtual environment in which, it is possible to generate a vast amount of data, label them using custom-made tools and then develop a CNN detector. However, the applicability of virtual data in real-world simulations and the transferability of the method have not been proven yet and remain to be examined (Filipowicz et al., 2017; Martinez et al., 2018).

In this study, 5,000 images were selected and annotated manually. Even though the detector is focused on vehicles and other objects are disregarded, the number of images is still far from the many thousands of images usually used to train a CNN. In order to reduce the time and cost required to collect and label more images, it is necessary to increase the size of the training dataset artificially or, “augment” the dataset.

Data augmentation is a common technique used to increase the amount of relevant data with samples that differ, even if the differences are minor in many cases. It is usually performed to reduce overfitting in models, using information already in the existing dataset. When using images as input data, common ways to augment the data is by using operations such as resizing, rotation, reflection etc. These operations produce instances of the same object (albeit slightly altered) that can be used as

additional input data. However, they are linear transformations of the same image and do not bring any new visual features that could improve the learning abilities of the model (Mikołajczyk and Grochowski, 2018). For example, reflection essentially retains the same pixel values in a sample image albeit in a mirror image, generating the same features. Therefore, the dataset size would increase without significant gains in the learning ability of the CNN detector. They are also more appropriate for classification purposes, i.e. when the object of interest occupies the whole or a large part of the image (Wang and Perez, 2017).

The network operates as a detector in this case and its input data are scene images containing vehicles in various locations on the road, along with their bounding box coordinates. To augment the data and increase the vehicle instances in the dataset, the following approach called *translation* was utilised:

- The size (width, height) of the bounding box in the original annotated image is kept constant
- To generate new vehicle instances, the bounding box is offset by a relatively small number of pixels in both  $x$  and  $y$  directions.
- New bounding boxes are generated for the image, containing the object of interest (vehicle)

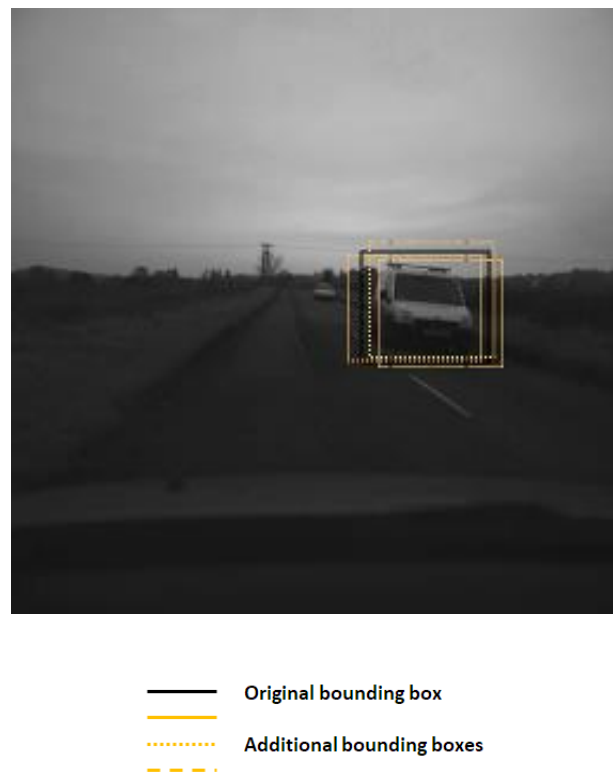
The maximum number of pixels used as offset in both directions is kept at a low +5 or -5 pixel, with the actual value generated randomly. The random generation of the offset value ensures the low probability of two identical boxes generated for the same object, while at the same time, the low maximum offset value ensures that:

- i. the object remains within the newly generated box (offset less than 1% of the width/height of the image – vehicle remains in focus) and
- ii. the newly generated box contains the vehicle along with a slightly different view of the surrounding environment (new pixel values for the surrounding environment, bringing new information to the model).

Arguably, generating additional vehicle instances using this augmentation method resembles annotation of the same image by different people or even annotating the same object in a different frame of the same second in the video file.

The original dataset of 5,000 images was increased in size to reach 20,000 images for training purposes. With each image containing two vehicle instances or more on average, the total number of vehicles contained is around 50,000. If the detector was intended to classify multiple objects (multi-class) as opposed to vehicle only, the dataset would have to increase in size to reflect the need to train on other objects too.

An example of the original and additional bounding boxes can be seen in Figure 4-8 below:



**Figure 4-8: Example of data augmentation**

## 4.5 Testing dataset

The dataset used to evaluate the real-world performance of the two detection methods consists of 1,000 images containing multiple vehicle instances (1,853 in total). It is a dataset independent from the two sets used to train the SVM classifier and CNN-based detector but has the same level of variability (vehicle types, operating environment, and weather conditions).

Similarly to the training dataset, the resolution of the testing images is 512x512 pixels. To ensure the dataset is varied enough, the KITTI dataset benchmark for difficulty is used (Geiger et al., 2012). According to the difficulty levels defined in this benchmark, the challenge for the detector to correctly identify vehicles in the testing dataset ranges across all three categories (easy, medium and hard) with many small bounding boxes and many occluded vehicles. The three difficulty categories are:

- Easy: Minimum bounding box height: 40 pixels, occlusion level: Fully visible, Maximum truncation: 15%
- Medium: Minimum bounding box height: 25 pixels, occlusion level: partly occluded, Maximum truncation: 30%
- Hard: Minimum bounding box height: 25 pixels, occlusion level: difficult to see, Maximum truncation: 50%

The minimum bounding box height defines the size of the vehicle in the image, with smaller boxes being a challenge for the detector. The level of occlusion is an additional difficulty factor, with vehicles being obscured by other objects in the medium and hard difficulty categories. Finally, the level of truncation is also considered, meaning it is possible that only parts of vehicles are visible in an image (with vehicles being at the edges of the image, entering or exiting the scene).

## 4.6 Limitations of the dataset

All three datasets were developed to include as many environmental conditions (sunny, overcast, rain, dusk) as possible, the goal being to produce a robust vehicle detector that is invariant to environmental or operational conditions. While the datasets used manage to include a good amount of variety, there are inherent limitations to the datasets that need to be considered when examining the performance of the vehicle detectors:

- The dataset used for training the CNN detector is constrained in size, as it was necessary in order to examine the detection performance using a limited amount of data.
- The image quality in the dataset varies, and this was purposefully done to examine the effect of degraded and noisy images to detector performance.
- Images are grayscale and contain no colour information, thus limiting the ability to use additional useful information as visual cues to detect vehicles.
- Lighting conditions are varied, with many images being dark or overexposed, thus introducing another layer of difficulty for the developed detectors.

## 4.7 Summary

This chapter presented the types of data used in this study, the collection and pre-processing that was necessary to generate the datasets.

Three different datasets were employed; two datasets used for training the SVM classifier and CNN detector, and a testing dataset upon which vehicle detection performance will be measured. The first one (SVM classifier) is comprised of positive and negative training samples that are necessary to train the classifier and validate its results. Its size (training + small validation set) is around 8,000 images. The second dataset is made up from 20,000 images. 5,000 images were selected and annotated manually, before an augmentation process increased the total number of samples to 20,000. Both detectors will be evaluated on an independent dataset



containing 1,000 images with multiple vehicle instances in many cases. The number of vehicles in this dataset is over 1,800.

Finally, radar data are used as ground truth in range measurement using a monocular camera. This data will be used to determine whether the camera is sufficiently accurate to compute surrogate safety measures such as TTC.

## 5 Results

### 5.1 Introduction

This chapter presents the results for the two vehicle detection methods: (1) the HOG-based detector with SVM (Support Vector Machine) classification and (2) the deep learning based detector – convolutional neural network. As the aim is to produce a high-precision vehicle detector that is able to operate in real-time, the two methods need to be compared and their performance discussed.

The methods are tested on an independent dataset consisting of 1000 images with multiple vehicle instances in them. The images that make up both training and testing datasets were collected over different data collection runs using the instrumented vehicle. That way, it became possible to collect data in different operational (e.g. motorway, urban, rural) and weather (e.g. clear sky, overcast, rainy) conditions.

Initially, the performance of the HOG-based detector is assessed. Since the detection process is separated in two stages, Hypothesis Generation (HG) and Hypothesis Verification (HV), it is essential that both perform equally well. If that is the case, the detection process operates robustly, otherwise (if one or both stages are found to be under performing) the detection process cannot be considered successful.

The following section examines the performance of the CNN-based vehicle detector. In this case, a unified detection pipeline handles both generation of ROIs and their classification in an efficient manner. This efficient and high-performing process is one of the reasons why Deep Neural Networks have substituted traditional image processing methods in object detection and classification.

The results clearly show the superiority of the CNN-based vehicle detector. The proposed HOG-based method does not perform sufficiently well to be considered a robust solution. The reasons for this result will be discussed extensively in the next section, with examples provided to highlight the problems that developed.

## 5.2 HOG-based detector and SVM classification results

This section investigates the performance of a vehicle detector that generates Regions of Interest (ROIs) based on information derived from the extraction of HOG features. While established methods make use of various visual cues for ROI generation (e.g. extracting salient features from the raw image data) and utilise HOG features for classification of objects, the method presented here attempts to use HOG features to extract the necessary information for the ROI generation part of the detection process.

### Hypothesis Verification (SVM classification) results

Classification of the extracted regions takes place using a Support Vector Machine (SVM) that is trained using a dataset consisting of positive and negative training samples. Initial testing has indicated that two of the available kernel functions (linear and Fine Gaussian kernel) were the most likely to produce the best classification results.

The SVM classifiers were tested on two image resolutions as well, in order to identify differences in classification performance. The two resolutions used for the training dataset were 128x128 and 256x256 pixels.

The training dataset consists of 7,693 images in total. Out of this number, 2,135 images are positive training samples (vehicle instances) while 5,558 images are negative samples (which do not contain vehicles or parts of vehicles). The ratio between positive and negative training samples is around 1:2.6. The dataset was tested and no issues regarding imbalanced dataset were observed. An increased number of negative training samples can be used to increase the variability of the non-vehicle object class (Dalal and Triggs, 2005; Li and Guo, 2013; Teoh, 2011).

### **Validation**

A trained classifier may have good performance on the training dataset but fail to perform on a validation set. An estimate of its performance can be provided by cross validating. For  $v$ -fold validation, the data is divided in  $v$  number of sub-sets.  $v - 1$

sub-sets are used for training the classifier, while the last sub-set is used for validation internally. The process is repeated for  $v$  times, each with a different sub-set of data used for validation. It is then followed by calculating the average of all validation results which is used as a measure of the SVM classifier's performance (Teoh, 2011).

The results are further validated by a small independent dataset generated for this purpose, consisting of 300 image samples (100 positive/200 negative samples).

### **ROC (Receiver Operation Characteristic) curve**

The ROC curve is another measure that is commonly used for assessing the performance of a classifier (Godil et al., 2014). It is a plot of points showing the trade-off between the classifier's true positive (TP) and false positive (FP) rate. A classifier is considered to have better performance compared to another when its operating point lies closer to the top left corner of the graph (essentially maximising the Area Under Curve – AUC).

The four terms commonly used to describe correct and incorrect classification results are the following:

**Positive (P):** Positive case in the data (vehicle)

**Negative (N):** Negative case in the data (non-vehicle)

**True Positive (TP):** Vehicle correctly identified as present in a frame (vehicle present in ground truth)

**False Positive (FP):** Object incorrectly identified as vehicle (vehicle not present in ground truth) – False Alarm

**True Negative (TN):** Object correctly identified as non-vehicle (vehicle not present in ground truth)

**False Negative (FN):** Vehicle not identified in frame (vehicle present in ground truth) – Missed detection

Key metrics that can be used to more accurately measure classifier performance are (Godil et al., 2014):

The accuracy (ACC) metric is an actual measure of performance with regards to correctly identifying targets. It is the ratio of the sum of TP and TN detections relative to the total number of objects.

$$Accuracy (ACC) = \frac{TP+TN}{P+N} \quad (5.1)$$

Recall (or sensitivity or True Positive Rate) is the ratio of true positives to the sum of TP and false negatives (FN) in the classifiers, based on the ground truth.

$$Recall (R) = \frac{TP}{TP+FN} \quad (5.2)$$

Miss rate (or False Negative Rate) is the ratio of false negatives to the sum of FP and True Negatives (FN).

$$Miss\ rate = \frac{FN}{FN+TP} = 1 - Recall \quad (5.3)$$

False Positive Rate is the ratio of FP detections over the sum of FP and TN.

$$FPR = \frac{FP}{FP+TN} \quad (5.4)$$

The results for the different combinations for SVM classification are presented below:

### **SVM model 1 – 128x128 pixels – Linear kernel function**

This first SVM produced a vehicle classifier with an accuracy of 94.2%. The confusion matrix and ROC curve for the classifier can be seen below:

**Table 5-1: Linear SVM classifier (128x128) performance**

<b>Classifier</b>	<b>TP</b>	<b>FP</b>	<b>TN</b>	<b>FN</b>	<b>Total</b>
<b>SVM linear (128x128)</b>	1834	145	5413	301	7693

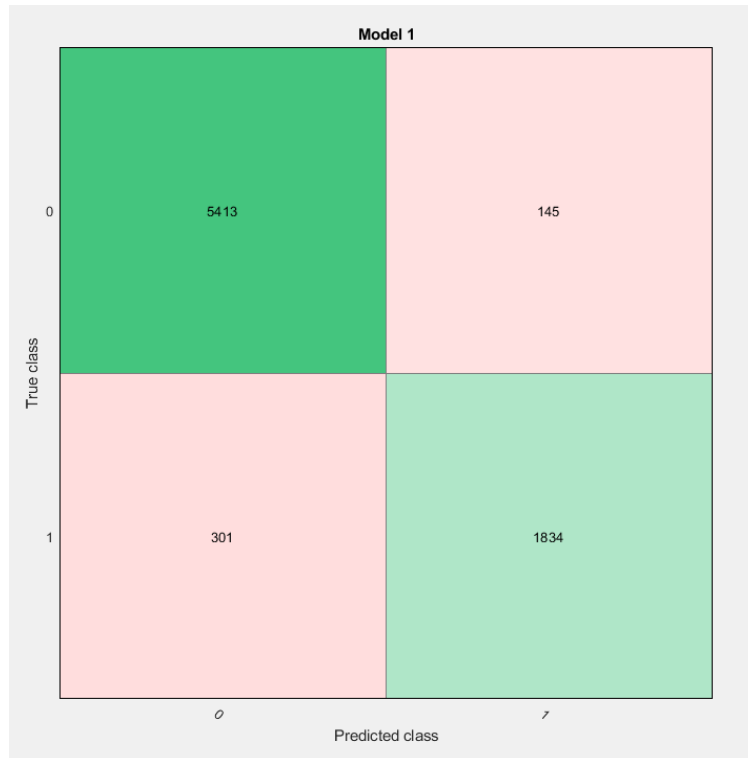


Figure 5-1: Confusion matrix - SVM linear kernel 128x128

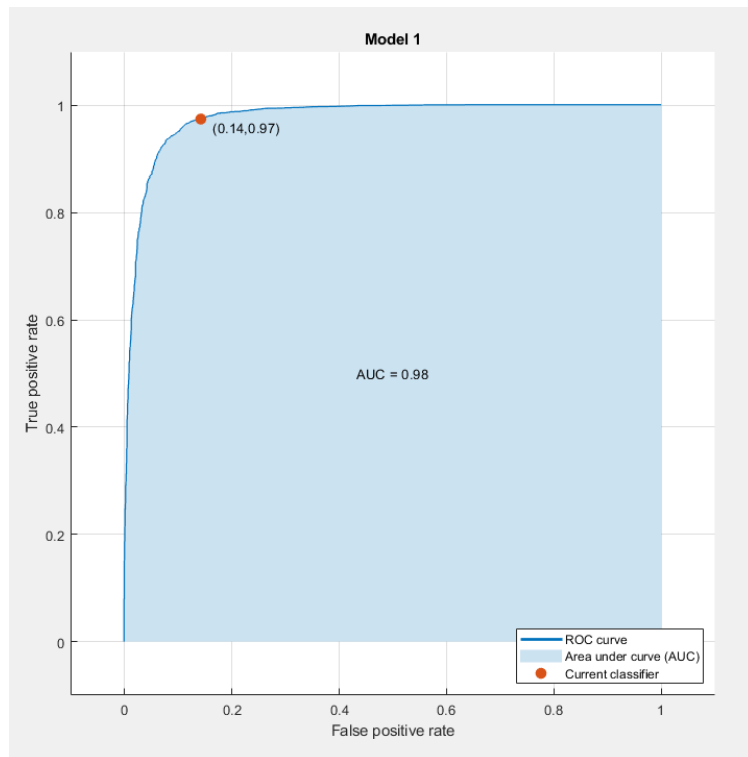
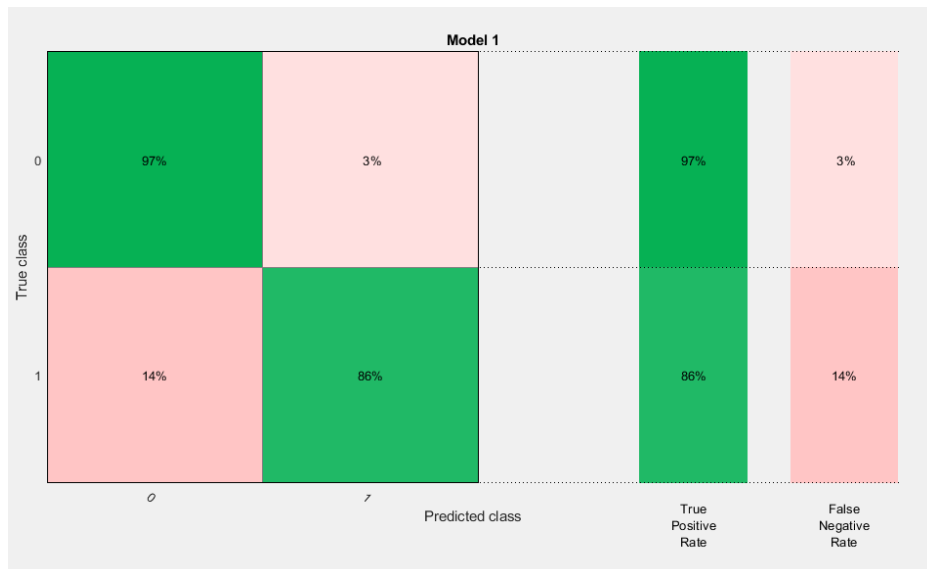


Figure 5-2: ROC curve - SVM linear kernel 128x128

The results show that the SVM classifier achieves an accuracy of 94.2% (which is high), a True Positive Rate of 0.86 (miss rate of 0.14). False positive rate is at a low 0.025 or 2.5%.



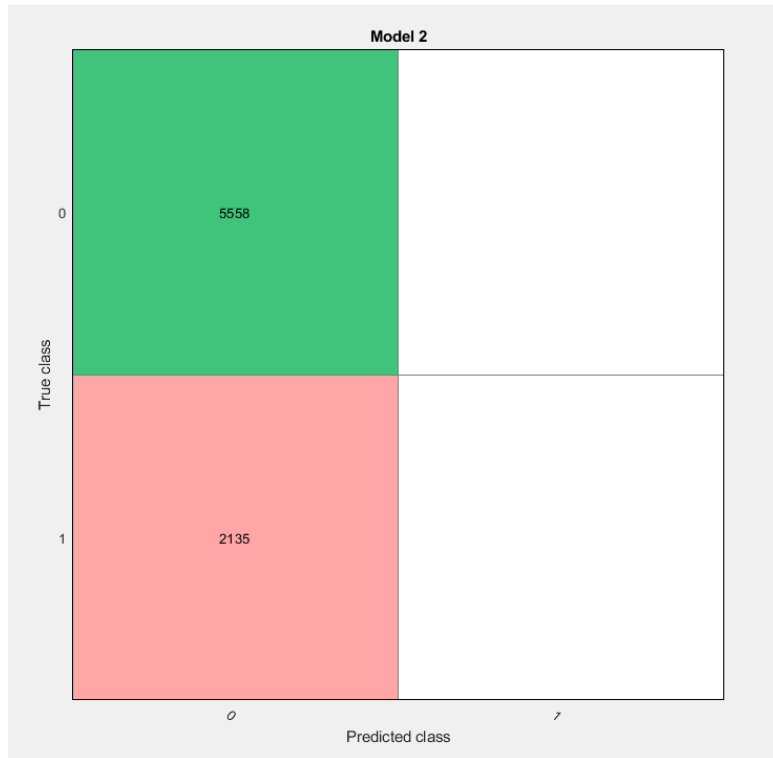
**Figure 5-3: TPR/FNR - SVM linear kernel 128x128**

### **SVM model 2 – 128x128 pixels – Fine Gaussian kernel function**

This SVM classifier using the F.Gaussian kernel produced a completely unworkable classifier compared to the one using a linear kernel. The confusion matrix and ROC curve for the particular SVM classifier can be seen below:

**Table 5-2: F.Gaussian SVM classifier (128x128) performance**

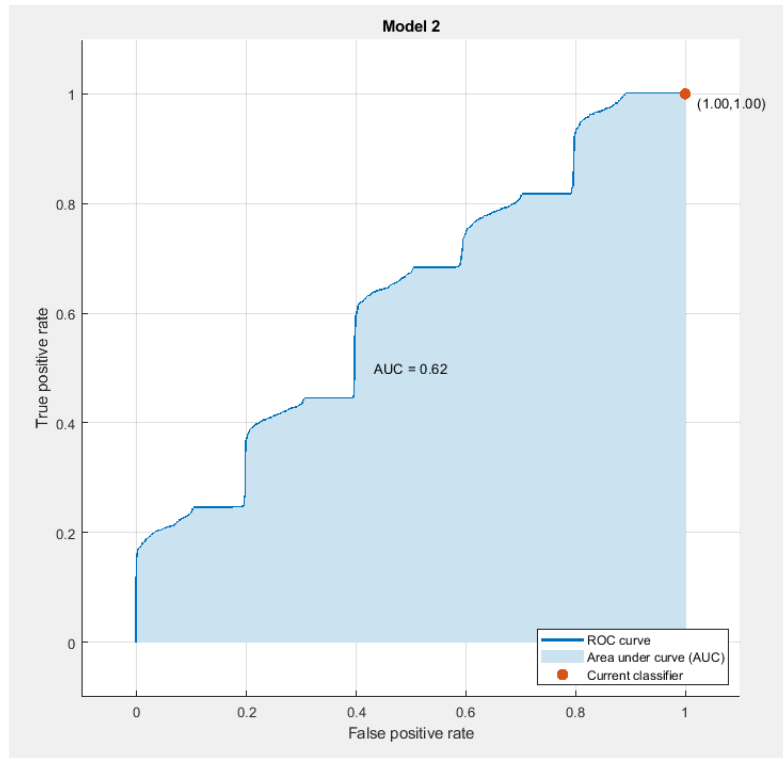
<b>Classifier</b>	<b>TP</b>	<b>FP</b>	<b>TN</b>	<b>FN</b>	<b>Total</b>
<b>SVM F. Gaussian (128x128)</b>	0	0	5558	2135	7693



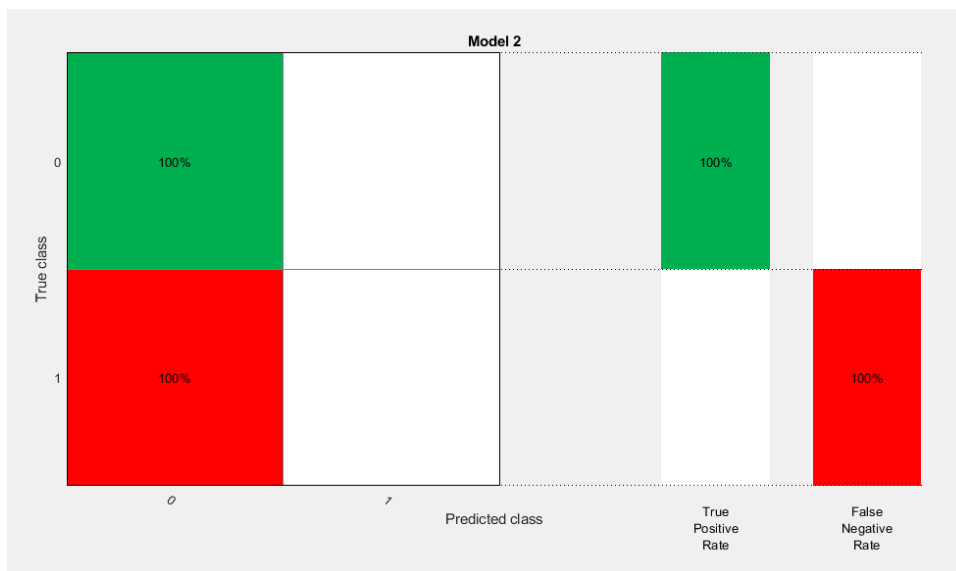
**Figure 5-4: Confusion matrix - SVM F.Gaussian kernel 128x128**

The accuracy rate for this trained classifier is 72.2%. The result is not indicative of its performance at all, as the confusion matrix shows an inability to correctly classify vehicles (no TP instances out of a total of 2,135). The classifier trained here using a F.Gaussian kernel is unsuitable for vehicle detection, producing incorrect classification results. The ROC curve is also a testament to this issue.





**Figure 5-5: ROC curve - SVM F.Gaussian kernel 128x128**



**Figure 5-6: TPR/FNR - SVM F.Gaussian kernel 128x128**

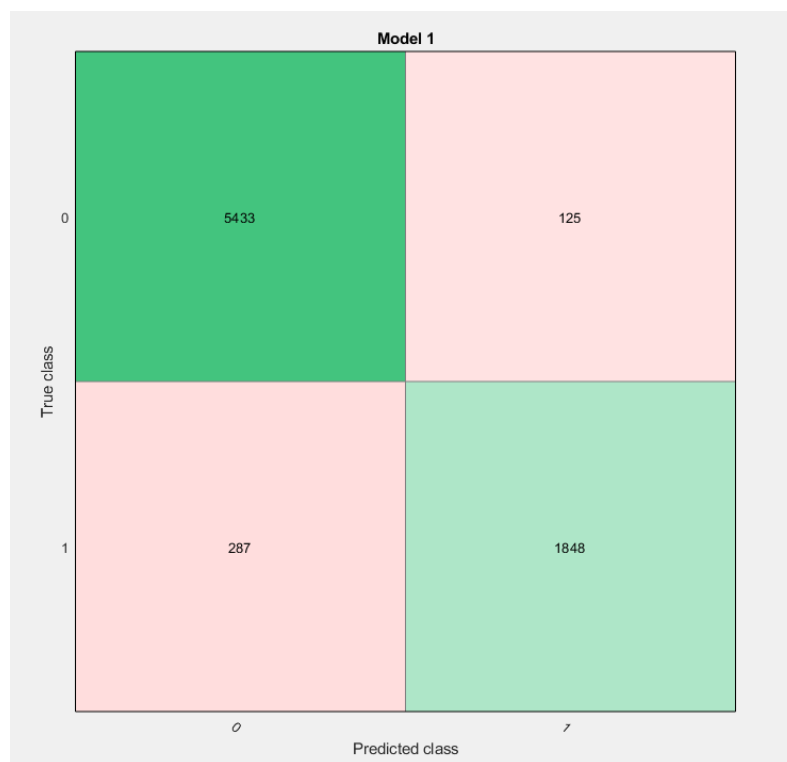
### SVM model 3 – 256x256 pixels – Linear kernel function

Finally, the last SVM classifier was trained using 256x256 pixel image patches, a rather large size given that most vehicles in an image frame should be smaller than this size. Only vehicles close to the camera should be close to this size (or higher). The aim for using this image resolution was to test whether there is any difference in classification accuracy.

The resulting confusion matrix and ROC curve can be seen below:

**Table 5-3: Linear SVM classifier (256x256) performance**

<b>Classifier</b>	<b>TP</b>	<b>FP</b>	<b>TN</b>	<b>FN</b>	<b>Total</b>
<b>SVM linear (256x256)</b>	1848	125	5433	287	7693



**Figure 5-7: Confusion matrix - SVM linear kernel 256x256**

The accuracy rate for the classifier trained on higher resolution image patches is 94.6%. True Positive rate stands at 0.87 (with a miss rate of 0.13) while the False Positive Rate is 0.022 (2.2%).

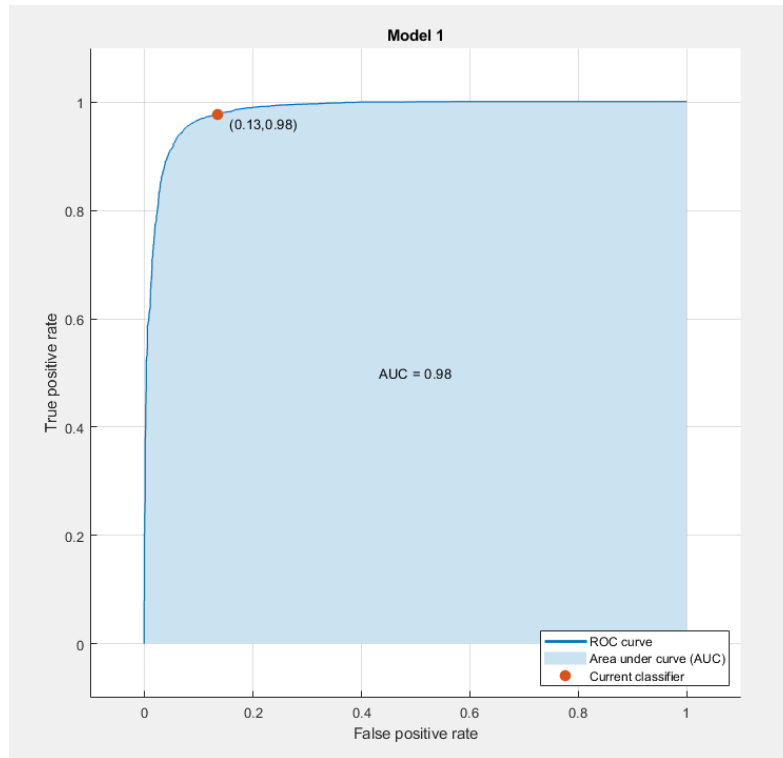


Figure 5-8: ROC curve - SVM linear kernel 256x256

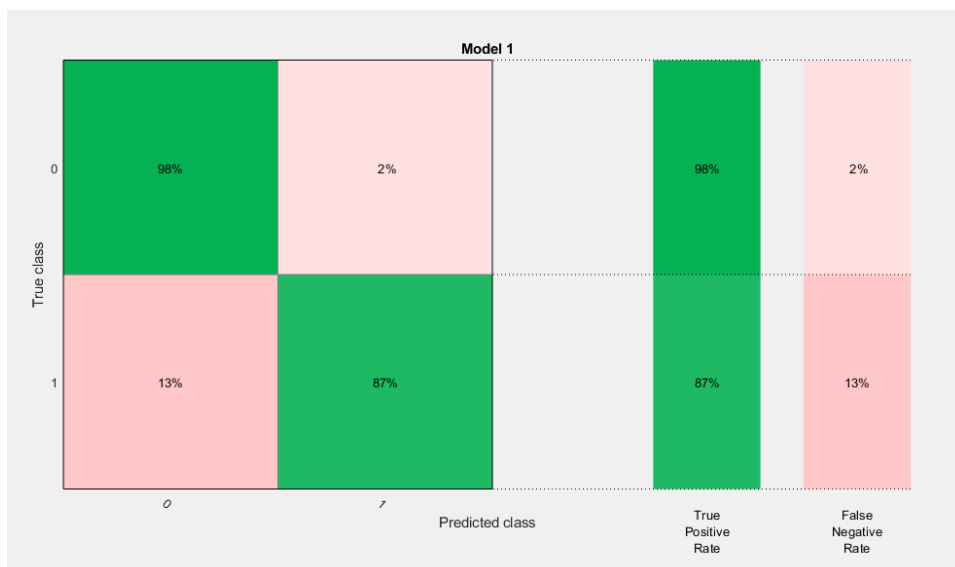


Figure 5-9: TPR/FNR - SVM linear kernel 256x256

It is clear that the SVM classifiers trained using a linear function kernel perform better compared to the F.Gaussian ones. While initial testing using a smaller dataset suggested that the F.Gaussian kernel in the SVM resulted in better classification accuracy, this is not the case when the classifier is trained using a much larger training dataset.

Between the two different resolutions, the difference in performance is small (0.4% in accuracy). For that reason, and also due to the increased processing time for the higher resolution image patches (SVM processes 110 observations at 128x128 resolution, while only 17 at 256x256 pixels), the SVM classifier trained on 128x128 pixel images is elected to be used in the vehicle detector.

Overall detector performance is highly dependent on the quality and the number of generated ROIs from the proposed HOG-based generation process. The following section presents the output of the ROI generation process.

### **Hypothesis Generation (ROI generation) results**

The ROI generation process has been in described in section 3.2 of the Methodology chapter. The method explores the possibility of using the horizontal gradients of the produced HOG feature vector to detect long horizontal edges in vehicles. Through them, a bounding box can be generated around the vehicle in the image frame and subsequently used as ROI for classification.

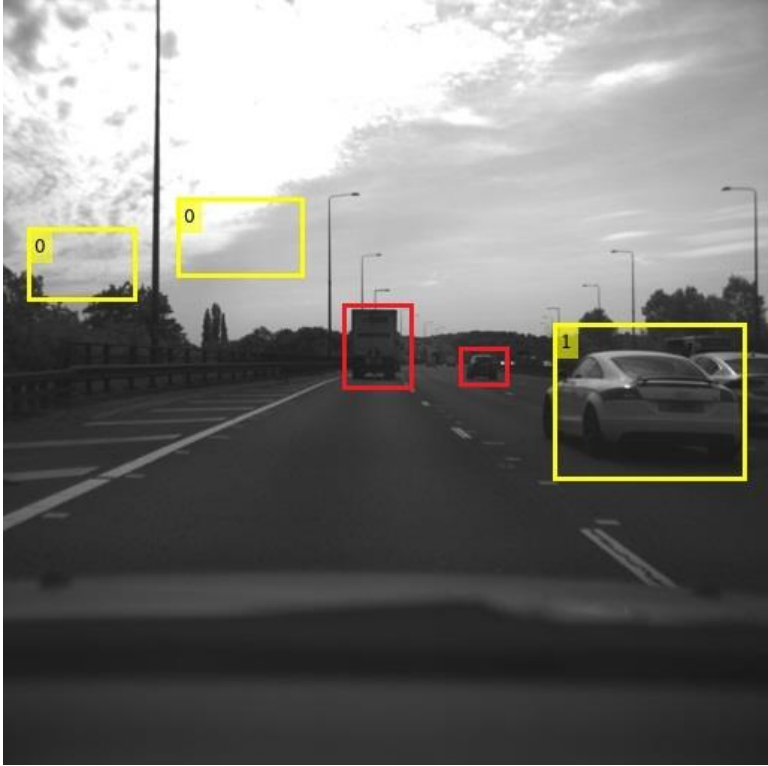
The vehicle detector (ROI generation+ SVM classifier) is tested on a dataset consisting of 1000 images. The output of the detector is a location of the vehicle and its predicted label (vehicle label: 1 and non-vehicle label: 0) from the SVM classifier. The final results indicate a high number of False Negative (FN) detections, with the detector missing the majority of vehicles in the dataset. The detector manages to correctly classify most of the generated ROIs, but the problem lies with the high miss rate. For the purpose of reviewing the results and identifying problems, TN detections are included in the images.

**Table 5-4: HOG-based detector performance**

Vehicle detector	TP	FP	TN	FN	Total
<b>HOG-based detector +SVM</b>	367	High	High	1486	1853

The Recall rate for the current detector is at a low 20%.

Examples of the ROI generation process are provided below. While there are a number of successful detections, it is obvious that the results are less than ideal, given the high miss rate. Despite the high number of generated bounding boxes, there are cases where vehicles are not correctly detected or misclassified by the SVM classifier. The reasons for this sub-optimal performance will be explained further. FN detections in the following images are indicated by red bounding boxes.

	<p>Number of ground truth vehicles: 3</p> <p>TP detections: 1</p> <p>The vehicle closest to the ego-vehicle is correctly identified.</p> <p>Other bounding boxes correctly classified as non-vehicles (0).</p>
---	--

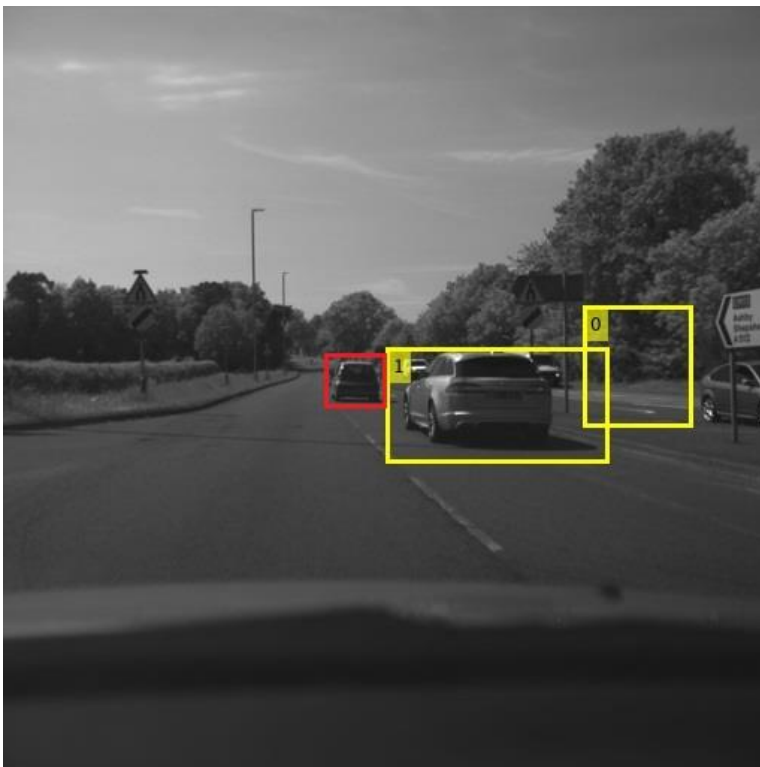


**Number of ground truth vehicles: 1**

TP detections: 1

The vehicle closest to the ego-vehicle is correctly identified.

No FP or TN detections



Number of ground truth vehicles: 2

TP detections: 1

The vehicle closest to the ego-vehicle is correctly identified.

No FP detections in the image. Other bounding boxes correctly classified as non-vehicles.



**Number of ground truth vehicles: 1**

TP detections: 1

One False Positive (FP) detection in the image.

The image quality is severely downgraded due to pixelation artefacts.

The vehicle detection system should be able to handle such cases, where there are visibility issues, faulty recordings or connection issues.



**Number of ground truth vehicles: 2**

TP detections: 2

One TN detection in the image.

The bottom half of the image is very dark, making discerning objects in it difficult.

Illumination conditions like this make it difficult for a camera system to operate.



Number of ground truth vehicles: 1

TP detections: 1

The vehicle closest to the ego-vehicle is correctly identified.

No FP or TN detections



Number of ground truth vehicles: 2

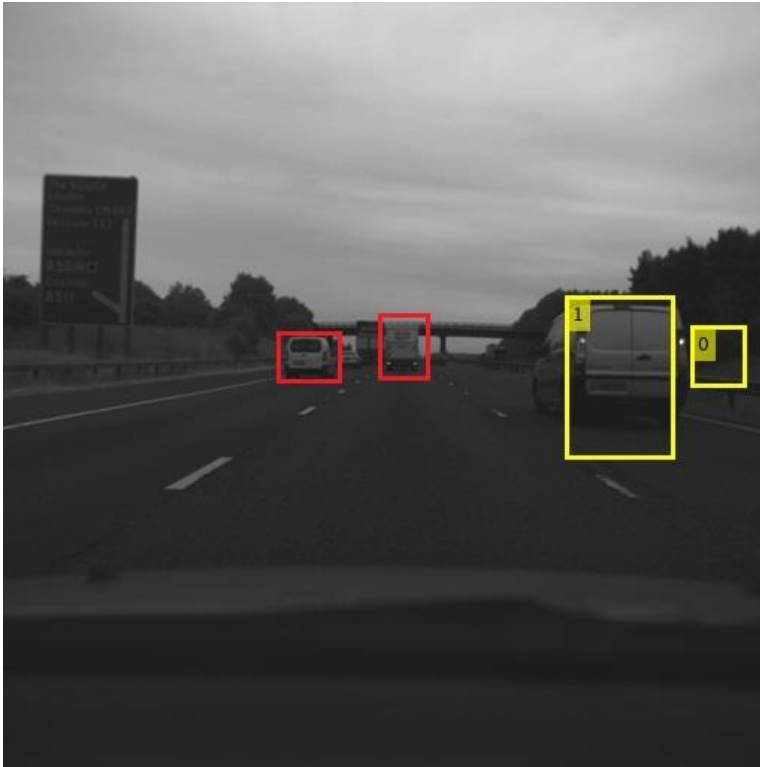
TP detections: 1

The vehicle closest to the ego-vehicle is correctly identified.

1 TN detection in the image.

The partial vehicle on the left of the image not detected (FN).



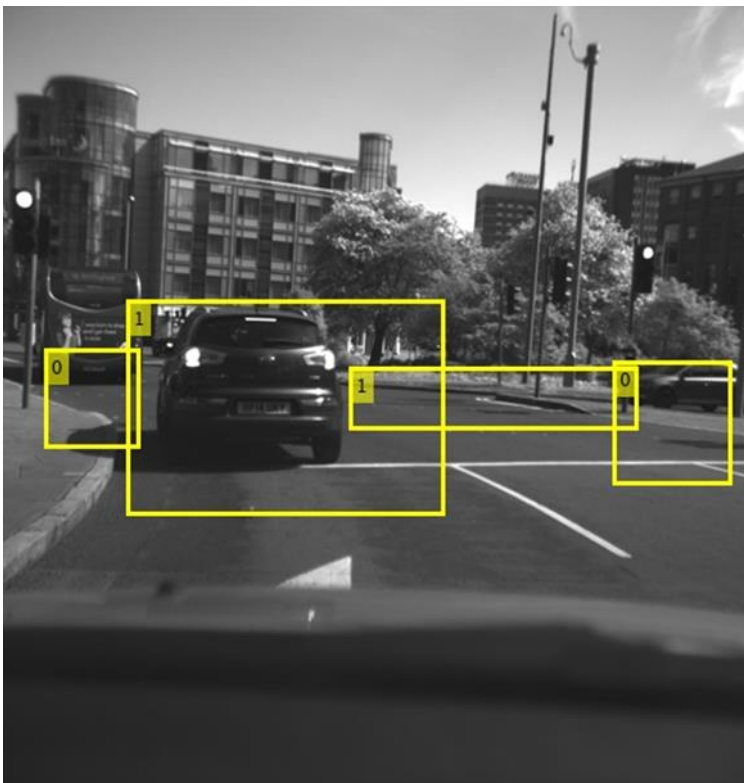


Number of ground truth vehicles: 3

TP detections: 1

The vehicle closest to the ego-vehicle is correctly identified.

1 TN detection, missed detections not critical for a safety application.

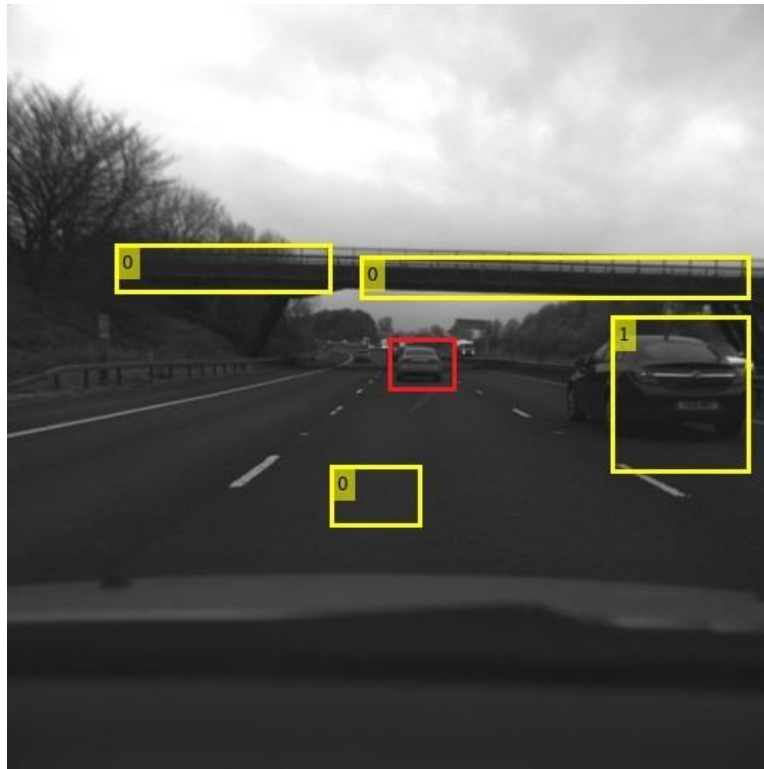


Number of ground truth vehicles: 1

TP detections: 1

The vehicle is correctly identified.

1 FP, 2TN detections. The ROI generation module generates additional candidate locations in a structured environment where there are more objects with long horizontal edges.



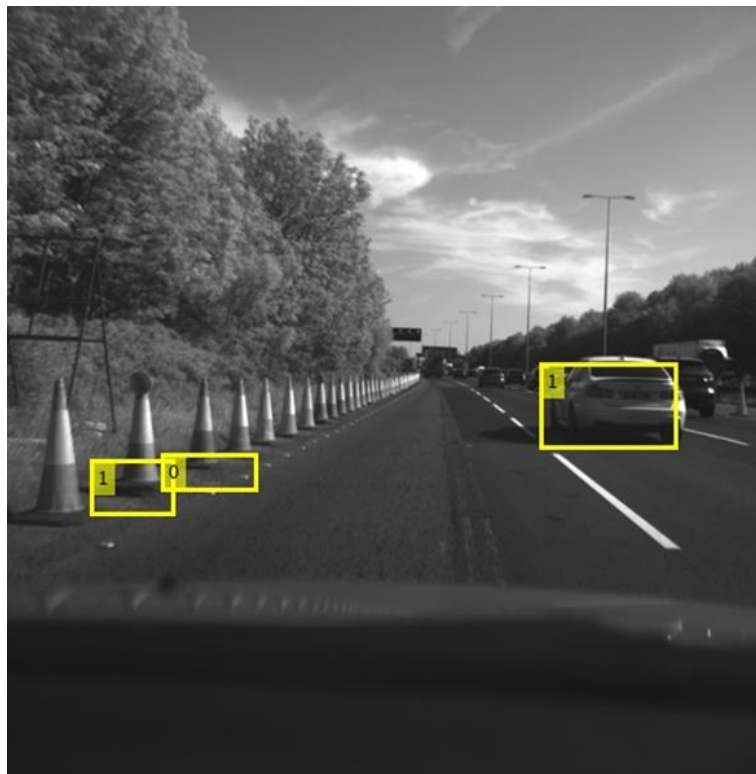
Number of ground truth vehicles: 2

TP detections: 1

The vehicle closest to the ego-vehicle is correctly identified.

3 TN detections correctly identified as non-vehicles.

The long horizontal overpass is identified as a potential vehicle location, as it is designed to detect horizontal edges.



Number of ground truth vehicles: 1

TP detections: 1

The vehicle closest to the ego-vehicle is correctly identified.

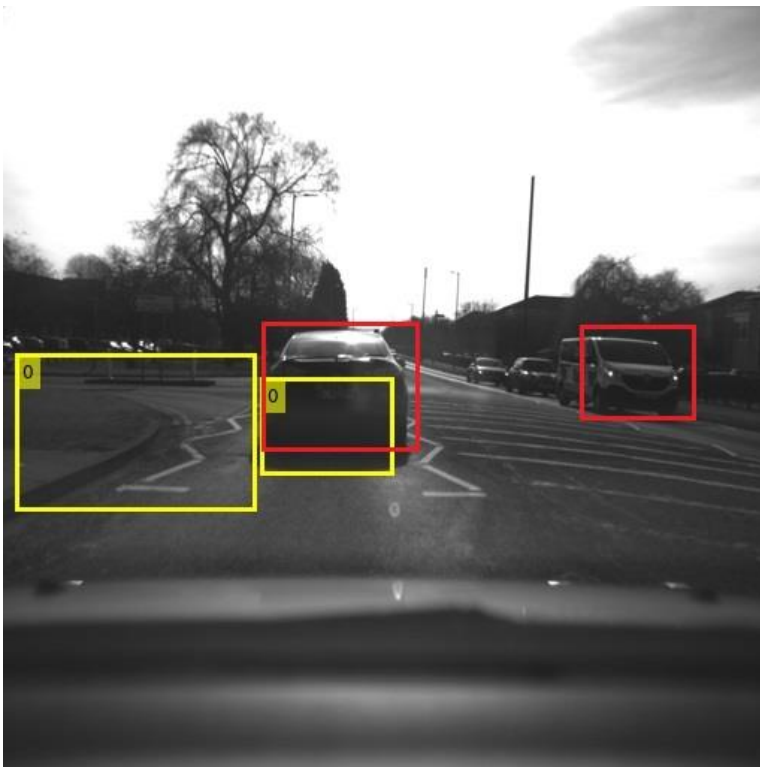
A FP and a TN detection appear in the image.



Number of ground truth vehicles: 1

TP detections: 0

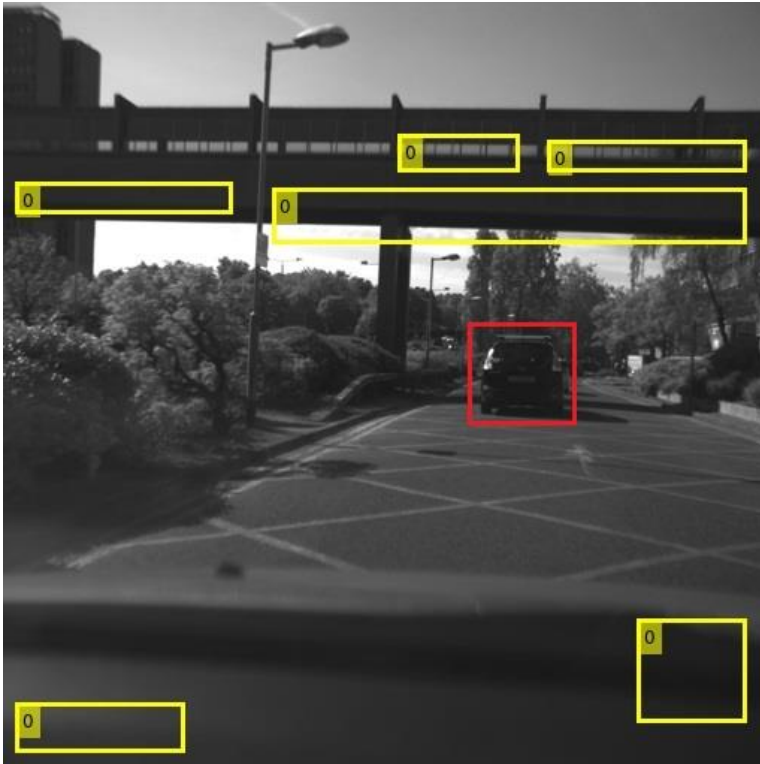
The vehicle is not identified. A ROI is generated around the vehicle but contains a significant amount of background information. The SVM classifier cannot validate the existence of a vehicle.



Number of ground truth vehicles: 2

TP detections: 0

The vehicle is not identified. In this case, the bounding box is smaller than the actual vehicle. Similarly to the previous case, the SVM classifier cannot correctly identify the vehicle.



Number of ground truth vehicles: 1

TP detections: 0

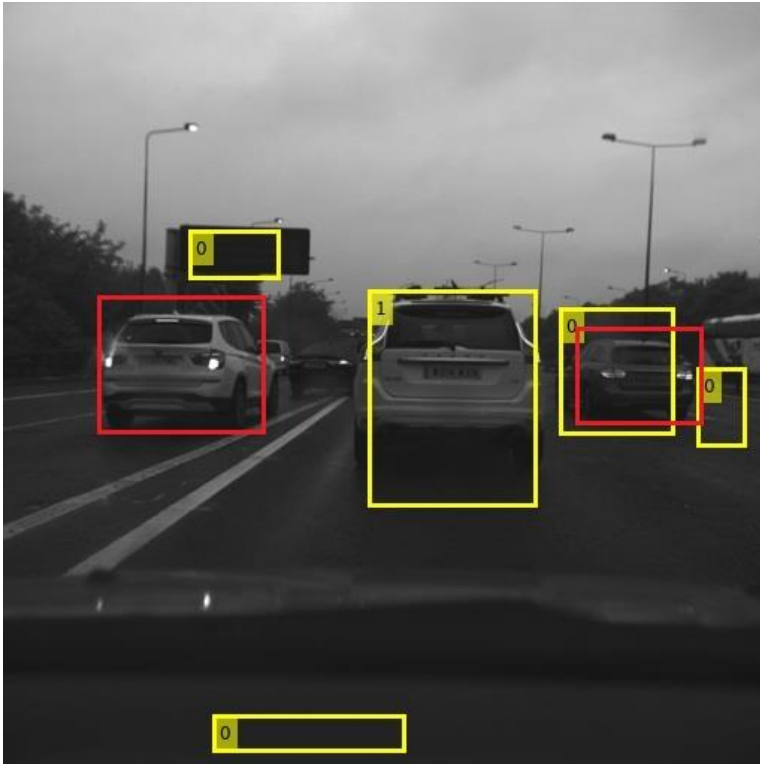
The vehicle is not identified. The ROI generation process has failed to identify a vehicle, possibly because it is unable to separate it from its surrounding environment. A large number of bounding boxes generated in this image indicate large horizontal objects. The classifier successfully identifies them as non-vehicles.



Number of ground truth vehicles: 1

TP detections: 0

The vehicle is not identified. A ROI is generated around the vehicle but contains a significant amount of background information. The SVM classifier cannot validate the existence of a vehicle.



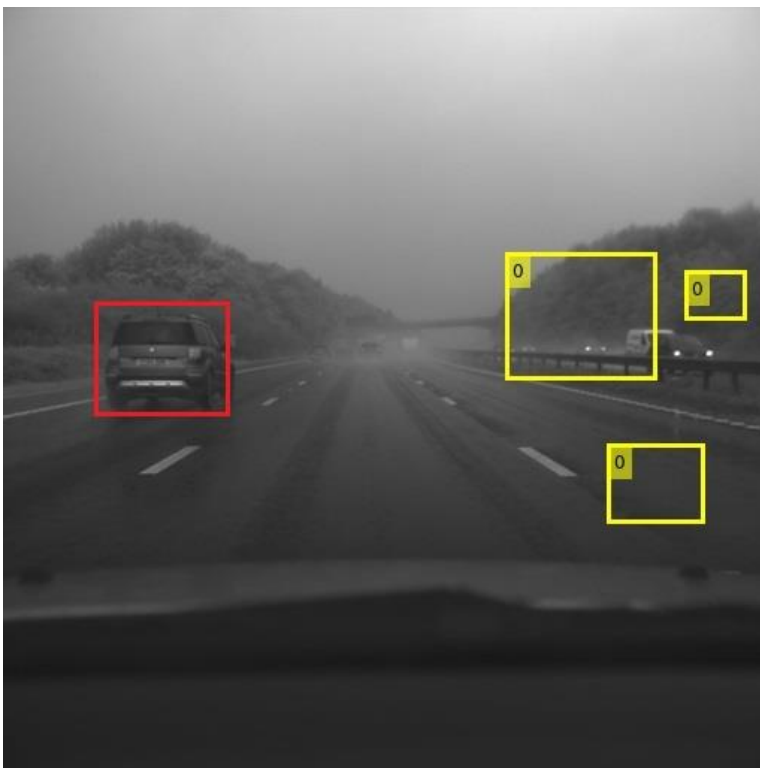
Number of ground truth vehicles: 3

TP detections: 1

The vehicle closest to the ego-vehicle is correctly identified.

The SVM classifier fails to identify a vehicle on the right, while the ROI generation module misses the vehicle on the left completely.

The information sign on the left is identified as a potential vehicle location.



Number of ground truth vehicles: 1

TP detections: 0

The detector misses completely the vehicle on the left side of image. The weather is rainy on this occasion and the low contrast of the image is possibly the reason the vehicle cannot be separated from the environment.

The detection results clearly show that this combination of ROI generation and classification is unsuitable for vehicle detection. While the initial processing stages of an image show there is potential in using HOG bin orientation as a visual cue for vehicle detection, the end result does not justify the use of this particular method as is and needs to be improved in order to become usable.

The additional use of a Canny filter as a segmentation tool to improve the clustering (see Methodology section 3.2.1 on vehicle-environment separation) of image areas into objects works but is clearly not enough to produce a consistent result. The problem lies with the fact that each image requires a different approach in order to isolate the vehicle from its surrounding environment and there is no universal set of parameters that works for every image in the dataset. The end result shows, that in some images the size of the bounding box around the vehicle is smaller than necessary for a correct classification result, while for others the opposite is true.

It is also apparent that there is a high number of generated ROIs that do not belong to a vehicle, especially where there are rectangular objects in the background (bridges, railings, buildings in urban areas). While this is not ideal, the SVM classifiers manages, in most cases, to correctly classify these objects as non-vehicle with the only drawback being the additional CPU time required to process these image areas.

The issues discussed here are intensified by the intrinsic limitations of the dataset used:

- The quality of the images is a limitation factor. While the image resolution is high enough to extract salient image information, they are all grayscale, thus immediately limiting the available methods that can be used to detect vehicles.
- Lighting conditions are in many cases poor, with many images being dark or low contrast, leading to difficulty detecting shadows or separating the vehicle from its surrounding environment and others suffering from lens flare effect. Both these issues are common problems associated with camera-based detection.

It is possible that, without the issues mentioned here, vehicle-environment separation would be better and the detection results improved. However, since the aim is to build a robust vehicle detector that is able to detect vehicles across different environment and in different conditions, and not a situation-specific detector, it can be concluded that this vehicle detector does not meet the requirements set.

### **5.3 CNN-based vehicle detector results**

In the current section, the performance and capabilities of the developed CNN-based detector models are explored. Each of the CNN models is assessed through appropriate quantitative metrics while, similarly to the previous section, sample images are provided and discussed where necessary.

In total, 6 network models with different number of layers and detection parameters are examined. The first model is used as reference and another five building on the structure of the base one with modifications that aim to determine which set of options and parameters maximise vehicle detection performance.

As described previously, the detectors were trained end to end, initially using 5,000 manually annotated images as a base training set, with the detectors re-trained in successive steps (each step adding 5,000 additional training images) until a total of 20,000 images is used for training. With every image containing two vehicles on average, the total number of vehicle instances contained is around 50,000. If the detector was intended to classify multiple objects (multi-class) as opposed to vehicle only, the dataset would have to increase in size to reflect the need to train on other objects too. The performance of the detectors is examined at each training step to determine the effect of using augmented data on the system's detection ability.

The CNN detectors are evaluated using the same 1,000 image dataset used for the HOG based detector. The images used for testing are representative of different environmental conditions and of varying difficulty for an object detector. The results for the top-performing detector are disaggregated by environment type to identify potential strengths or weaknesses of the selected detector.

The detectors' output is the locations of vehicles in an image along with the confidence values for every detection. Since CNN-based detection is a resource intensive process (run-time on CPU is usually high, with the system requiring a capable GPU to perform adequately), it was decided to reduce the search window for vehicles and thus, improve run-time in order to be able to operate in real time. Essentially, the search window is a rectangle as wide as the image frame, but with the top and bottom areas removed. The top area is the sky, while the bottom is part of the car bonnet. The size has been selected so that there no vehicle misdetections. Figure 5-10 below presents the search area in a sample image:



**Figure 5-10: CNN vehicle search window**

The performance of CNN-based detectors is commonly measured using the Average Precision (AP) metric (Huang et al., 2017; Ren et al., 2015; Wei et al., 2019). Precision (P) is the ratio of True Positive (TP) detections over all object instances in the detector (True Positives and False Positives) and AP is measured over all detection results, averaging the precision results at different recall levels. Recall, as



mentioned before, is the ratio of true positives to the sum of true positives and false negatives (FN) in the detector, based on the ground truth.

$$\text{Precision (P)} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (5.5)$$

The AP metric summarises the shape of the precision/recall curve and was first formalised in the Pascal Visual Objects Challenge (VOC) (Everingham et al., 2010). At 11 equally spaced recall values  $\text{Recall}_i = (0, 0.1, \dots, 1)$ , the value of AP is:

$$\text{AP} = \frac{1}{11} \sum_{\text{Recall}_i} \text{Precision}(\text{Recall}_i) \quad (5.6)$$

Other measures of performance are the Recall and miss rate metrics. In this study, there are no TN detections, as this is not a classification task but a two-class (vehicle-background) detector and the test dataset only contains images with vehicle instances. A detection is considered True Positive (TP) if the Intersection over Union (IoU) of detected bounding box and ground truth is  $\geq 0.5$ .

In addition to the above metrics, the Precision-Recall curve (PR curve) is also indicative of the detector's performance. The PR curve is plotted by varying the confidence value. In pattern recognition and information retrieval, precision is the fraction of retrieved instances that are relevant, while recall (or sensitivity) is the fraction of relevant instances that are retrieved. Therefore, both metrics are measures of relevance.

The six models detailed in Methodology section 3.3 are the following:

Model 1 – Reference model (11 layers)

Model 2 – 13 layer structure (64 convolution kernels)

Model 3 – Reference structure with the addition of batch normalisation layers

Model 4 - Reference structure with modified Region Proposal Network

Model 5 – 13 layers structure (M2) with increased learn rate

Model 6 – 16 layer structure with modified number of kernels

The AP results for the six CNN models are presented in the table below:

**Table 5-5: AP performance for CNN models**

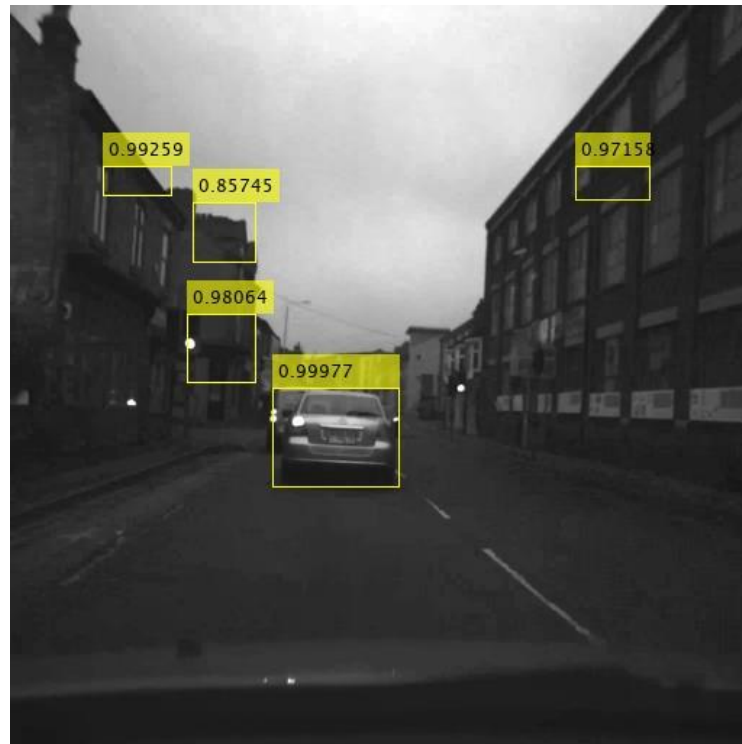
<i>No. of training images</i>	<b>AP performance (%)</b>			
	<i>5,000</i>	<i>10,000</i>	<i>15,000</i>	<i>20,000</i>
<b>Model 1 (Reference)</b>	46.53%	54.31%	<b>35.91%</b>	56.35%
<b>Model 2</b>	46.57%	67.01%	67.34%	69.24%
<b>Model 3</b>	43.42%	47.68%	49.11%	36.44%
<b>Model 4</b>	50.66%	54.16%	64.71%	61.67%
<b>Model 5</b>	0	0	0	0
<b>Model 6</b>	76.82%	83.04%	84.64%	<b>85.87%</b>

The results in Table 5-5 reflect the effect of CNN structure change on the final precision result, as well as the effect of adding training data during the training process. At first glance, the gradual increase in training data size brings improvements in detection performance, although that appears to be not true for some cases. Additionally, the change in structure and the size of the training dataset affects the model training time, with network model training taking 4-6 days (to reach 20,000 training images) depending on the complexity of the structure. In any case, the best-performing model will be selected for further testing, to identify its weaknesses and finally use it for application in real-time.

### **Model 1 (Reference)**

The reference model, containing 11 layers in total, exhibits relatively low performance (46.53 %) in the AP metric after training with 5,000 images. The result is inferior comparable to the performance of other detectors using the Faster R-CNN method, although that is expected due to the low number of training images used and the fact that other models are based on architectures pre-trained with a large number of images (Huang et al., 2017; Lee et al., 2016).

The result indicates that there is a large number of FP detections, hence the low precision score. A sample image with high-confidence FP detections can be seen below:



**Figure 5-11: FP detections in testing image**

Detection performance gradually improves as training data is added, with the exception of the 3<sup>rd</sup> step (15,000 images) where performance drops significantly. The drop in precision can be attributed to an increase in FP detections, while the number of correct detections remains the same.

The final vehicle detector based on the reference model has a 56.35% precision rate, an almost 10% increase over the base model. The detector does not perform sufficiently well for any kind of application, safety-critical or other.

### **Model 2 (13 layer structure with 64 filters for each convolutional layer)**

Model 2 benefits significantly from the addition of another convolution layer and the increased number of filters, as observed by the improved AP score. It confirms that

the increased information extracted from the image dataset benefited the detection process and improved the detector's generalisation ability. However, the trade-off was apparent, as the training time required was approximately twice that of the reference model.

While the detector performs identically in the first training step (46.57% precision), the jump is quite significant afterwards, highlighting the benefits brought by the larger dataset. The jump is slightly over 20% during this step and remains relatively constant, with the 20,000 image detector reaching 69.24% precision.

Using the same sample image as before, it is obvious that the number of FP detections is now reduced. There is a single FP detection with a low confidence score:



**Figure 5-12: Model 2 output image**

### **Model 3 (Reference structure with batch normalisation layers)**

The addition of batch normalisation layers offers nothing to the vehicle detector. On the contrary, their addition seems to degrade the detector significantly. Maximum

average precision for this type of models is 49.11% which makes them behave worse than the base model.

This level of degrading leads to the conclusion that this type of layers is completely unnecessary and its use should be avoided in the final detector used for application.

The total number of detections (TP+FP) for the 20,000 image model 3 was also high which leads to the conclusion that there is a very high number of False Positive detections in order to achieve such a low precision score.

#### **Model 4 (Reference structure with modifications to the RPN to produce a lower number of ROIs)**

This series of models were developed to explore the relationship between detection performance and run-time, if a modification in the RPN to generate half the number of Regions of Interest per image (1,000 down from 2,000) impacts the end result and the detection run-time.

The outcome of this experimentation was that there is no discernible difference in performance compared to the reference model. In fact, the precision scores appear to be slightly higher in comparison, with the end 20,000 image model scoring around 5% higher compared to the respective reference detector (61.67% and 56.35% respectively).

Detection time was also not affected significantly, with this model being around 5% faster (time unit is seconds) during detection compared to the reference detector.

The sample image below shows the detected vehicle along with a detection of a partial vehicle ahead of it:



**Figure 5-13: Model 4 output image**

### **Model 5 (Model 2 with increased weight learn rate)**

This model network was developed with the goal of determining whether a higher learn rate for the network parameters would lead to a faster/better convergence to the optimal result.

The learn rates selected for each training stage were part of a process of manually testing various settings, from which several were proven to be very high and the Neural Network failed to train. The learning rate for the first two training stages is set at  $5 \times 10^{-4}$  (0.0005) and for the fine-tuning stages at  $1 \times 10^{-5}$  (0.00001).

While using these settings the network managed to finish training, the detection process revealed that the network did not converge (or even diverged significantly) to a solution and thus, failed to perform its task.

### **Model 6 (16 layer structure with variable number of filters)**

This final model of the series proved to be the highest performing out of all the models tested in this study. The network structure is enhanced by an additional convolutional layer, a max pooling layer to down-sample the output of the convolutional layers and differing number of Conv filters for each convolutional layer).

The vehicle detector trained using the 20,000 image dataset manages to achieve an Average Precision rate of 85.87% and is by far the best performing model of the ones tested. Its result is comparable to other state of the art detectors that operate using deeper networks (containing a much higher number of convolutional layers and millions of parameters) such as Res-Net, GoogleNet/Inception and others and are pre-trained using many thousands (or even millions) of image samples (Jia Deng et al., 2009).

By comparison, this is a single-purpose detector tasked only with detecting vehicles and not multi-class detectors/classifiers and therefore, does not require the deep structure or million training samples as the aforementioned high-performance (and complexity) Neural Networks.

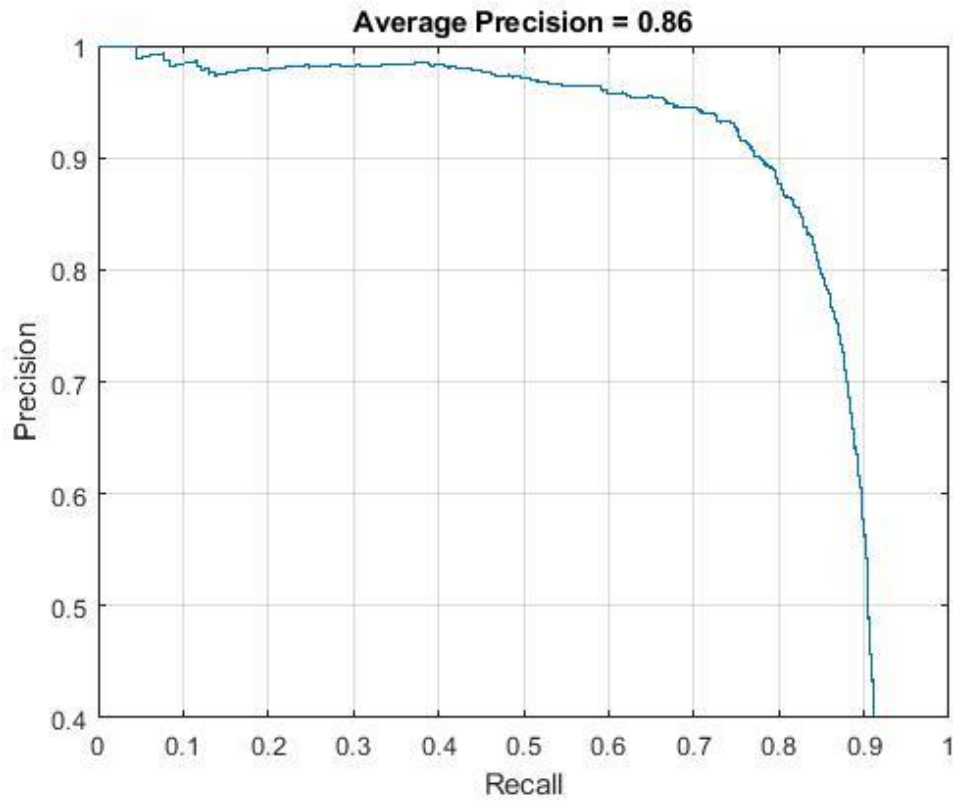
Since this is a detector trained end to end and given the size of the training dataset, the selected network depth is sufficient to avoid potential overfitting issues. That means that the detector retains its generalisation ability and performs well with new data. Overfitting in a network means that its parameters are tuned to the training data to such an extent, that instead of *learning* the complex relationships in the data structures, it *memorises* them. When overfitting occurs, the network appears to perform well but, when encountered with new, never seen before data, its actual performance is significantly lower.

The performance of the model will be examined in more detail below:

To evaluate the performance of a vehicle detector, especially when used in safety-critical application there a few key points that need to be considered:

- The most important issue in vehicle detectors used in safety-critical applications is the number of False Negative (FN) or missed detections. For a Collision Avoidance system, that is the difference between safe and unsafe conditions. Unsafe conditions can lead to the vehicle colliding with other vehicles or objects in general, posing a risk for fatality or injury or even property damage. The highest priority would be to minimise the number of missed detections in a Collision Avoidance System.
- A good object detector is able to identify only relevant objects quickly (producing a low number of FP – high precision) while at the same time find all ground truth objects (low number of FN - high Recall). Usually there is a trade-off between precision and recall, as it is necessary to increase the number of detected objects in order to find the ground truth objects. The Precision/Recall curve helps visualise this trade-off. The higher the curve (precision) as Recall increases, the better the detector performs its task.
- False Positive (FP) detections are important as they are False Alarms, indicating the presence of a vehicle when there is not actually one there. They are however, less important than missed detections (FN), as they pose a real danger to the vehicle.





**Figure 5-14: Model 6 Precision/Recall curve**

The 1,000 image testing dataset contains a total of 1,853 ground truth vehicle instances. The detector manages to detect a total of 1,837 vehicles, with 16 missed detections (FN) across the whole dataset.

The miss rate for the dataset is very low, at 0.008 while Recall at this rate is approximately 0.991 (99.1%).

Sample images of the detector output can be seen below:



**Figure 5-15: Model 6 output images**

To evaluate the robustness of the detector irrespective of traffic environment, the dataset was disaggregated by environment type (Urban, Motorway and Rural) and the precision was calculated for each type.

Even though the splitting the dataset into the 3 categories does not produce equally sized subsets, there is a good indication of performance by environment type. The number of images per type is:

- Urban: 461 images
- Motorway: 373 images
- Rural: 166 images

**Table 5-6: Precision by environment type**

Environment type	Urban	Motorway	Rural	Average Precision (total)
<b>Model 6 (20K images)</b>	86.66%	85.53%	85.56%	85.87%

The results demonstrate a robustness of the detector irrespective of operational environment. The difference in precision for each type is negligible and consistent with the overall result. A good indication of strong performance is the precision for the urban environment (86.66%) which is considered the most challenging, given its density. Given that the highest number of FP detections are expected in urban environments, the results show promise that the vehicle detector can handle difficult conditions well.

To summarise the results, the vehicle detector achieves a high Average Precision score (85.87%) which indicates a low number of FP detections. It also achieved a very low miss rate (0.8%) which indicates that almost all vehicle instances are identified in a scene. Additional testing validates its performance across all operational environments tested, with similarly high AP scores.

### **Missed detections**

The CNN detector's missed detections (FN) are presented over the next few pages. In total, only 16 vehicles were not identified across the whole dataset (1,853 vehicle instances). FN detections are indicated by red bounding boxes:



Number of ground truth vehicles: 3

TP detections: 2

The vehicle closest to the ego-vehicle is correctly identified.

The lorry on the left is included in the ground truth vehicles but is not detected. It poses no threat though due to its distance from the ego-vehicle.



Number of ground truth vehicles: 3

TP detections: 2

The vehicle closest to the ego-vehicle is correctly identified.

The vehicle on the right side of the image (opposite direction) is included in the ground truth.

It is far from the vehicle so there is no threat present.



Number of ground truth vehicles: 3

TP detections: 1

The vehicle closest to the ego-vehicle is correctly identified.

A vehicle and a lorry are not detected (both are part of the ground truth).

No threat present.

A partial vehicle on the opposite direction is detected.



Number of ground truth vehicles: 3

TP detections: 2

The vehicle closest to the ego-vehicle is correctly identified.

The detector does not manage to detect a small white van.

No threat present.



Number of ground truth vehicles: 2

TP detections: 1

The vehicle closest to the ego-vehicle is correctly identified.

The vehicle ahead (same lane) is not identified by the detector.



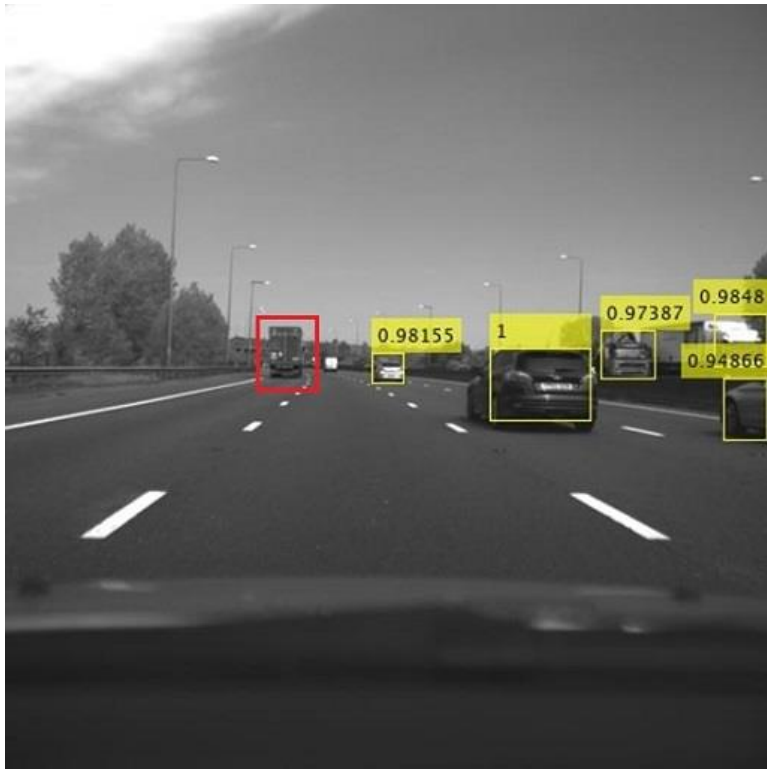
Number of ground truth vehicles: 2

TP detections: 1

The vehicle closest to the ego-vehicle is correctly identified.

The vehicle on the right hand side is not detected.

A FP detection in the image and an additional detection that should have been suppressed.



Number of ground truth vehicles: 3

TP detections: 2

The vehicle closest to the ego-vehicle is correctly identified.

The lorry on the left side of the image is not detected.

There are additional detections (vehicle partials) that do not belong in the ground truth.

No threat present.



Number of ground truth vehicles: 2

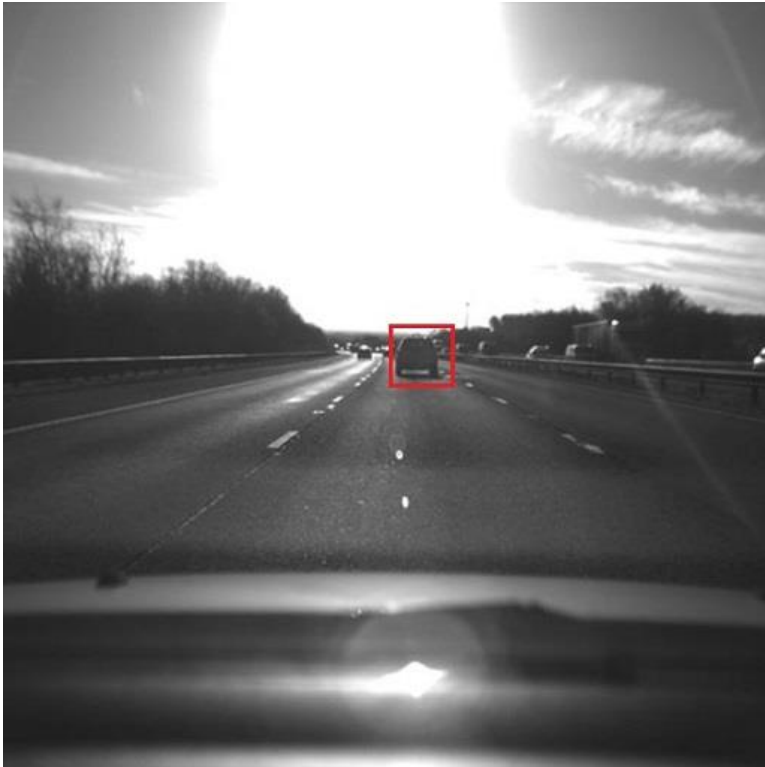
TP detections: 1

The vehicle closest to the ego-vehicle is correctly identified.

The vehicle in the same lane is not detected.

The lighting conditions are challenging (dark image).

No threat present.



Number of ground truth vehicles: 1

TP detections: 0

The vehicle in the image is not identified.

The lighting conditions are especially challenging for the camera.



Number of ground truth vehicles: 2

TP detections: 1

The vehicle on the oncoming direction is identified.

The vehicle in the same lane is not detected





Number of ground truth vehicles: 2

TP detections: 1

The vehicle closest to the ego-vehicle is correctly identified.

The lorry is not detected.

No threat present.



Number of ground truth vehicles: 2

TP detections: 1

The vehicle on the left is identified, although its irregular shape is a challenge for the detector.

The vehicle on the right is not identified due to the condition of the road.



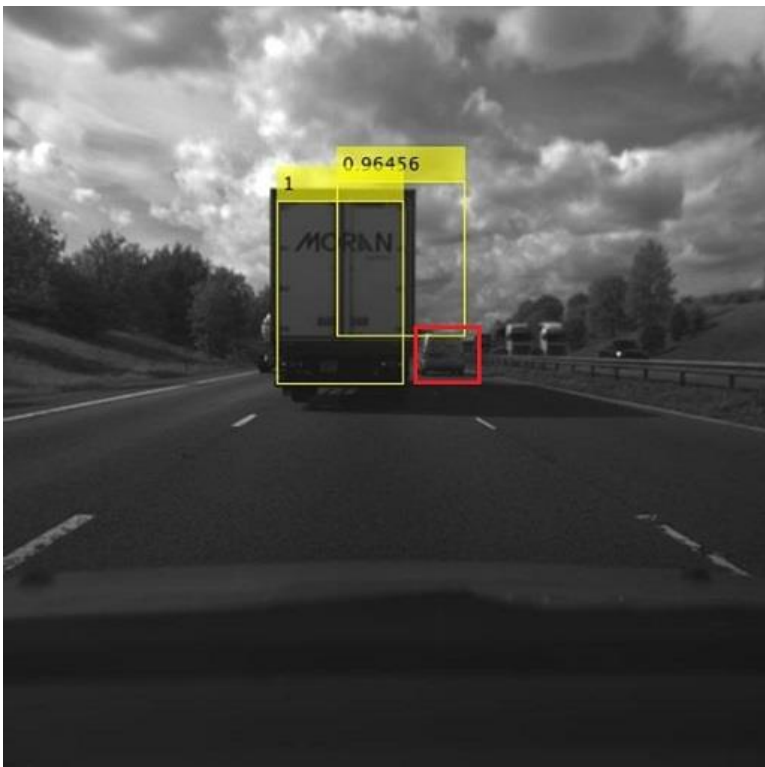
Number of ground truth vehicles: 2

TP detections: 1

The vehicle closest to the ego-vehicle is correctly identified.

The lorry ahead is not identified.

No threat present.




Number of ground truth vehicles: 2

TP detections: 1

The ahead is detected but an additional bounding box exists.

The vehicle on the right is not detected.

No threat present.

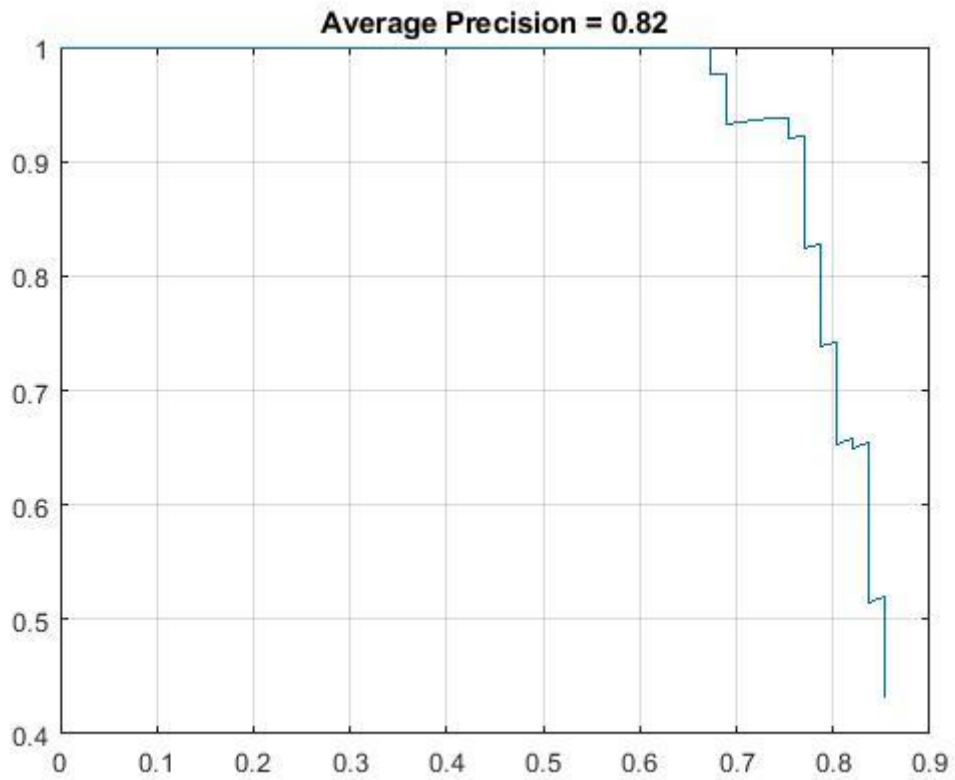
	<p>Number of ground truth vehicles: 2</p> <p>TP detections: 1</p> <p>The vehicle closest to the ego-vehicle is correctly identified.</p> <p>The lorry that is far on the left is not identified.</p> <p>No threat present.</p>
--	--

### **Detection results for an independent dataset**

To further validate the performance of the produced vehicle detector, additional independent data were used to test the detector. The average precision was calculated for two short dash cam videos with characteristics different to the data used to train and test the vehicle detector.

The two dash cam videos from the Caltech LISA vehicle dataset were used. Both videos are colour and of different resolution than the data (704x480) (Sivaraman and Trivedi, 2010).

The AP for vehicle detection calculated for the two sample videos is 0.8204. To visualise the result, the Precision/Recall curve was plotted in Figure 5-16 below:



**Figure 5-16: Precision/Recall curve for dash cam videos**

Sample images from the test utilising independently collected data are presented in Figures 5-17 and 5-18 below:



Figure 5-17: Detector output for dash cam video - Sunny



Figure 5-18: Detector output for dash cam video – Urban

## 5.4 Summary of results

This chapter presented the results of the two methods developed to detect vehicles on the road. The first method follows a traditional image processing approach for detection, where specific visual features are identified in an image to generate potential vehicle locations and then a classifier verifies the existence of a vehicle in the image. The method proposed here attempts to utilise the gradients produced by the extraction of HOG features to detect horizontal edges and through them, generate the desired Regions of Interest (ROIs). The produced ROIs are then passed on to a SVM classifier that decides whether a vehicle is present or not. Three SVM classifiers were trained using 7,639 images. Two of them were trained on a linear and F.Gaussian kernel on 128x128 pixels image patches, while the third one was trained on a linear kernel at 256x256 resolution images. The performance of the particular method, when tested on a 1,000 image test dataset, was found lacking. While the selected SVM classifier performed as expected, the ROI generation method failed to generate ROIs robustly enough to be used effectively as part of a vehicle detection system. The detection rates were low, resulting in a high number of missed detections that do not allow the developed system to be used in any kind of automotive application.

For the second method, six different CNN networks were developed, each with its own network depth and settings. All of them are based on the Faster RCNN architecture, where an RPN that generates ROIs is incorporated into the network, negating the need to generate candidate locations externally. All models were trained end to end, without the use of pre-trained networks, using own collected data. It was attempted to identify which network model performed the best in terms of detection precision and produced the lowest number of missed detections as possible, given that the available training dataset was limited. The models were trained gradually in four steps, where each step added additional training images. Starting at a low of 5,000 image samples, until a maximum of 20,000 images was reached in the final step, performance gains were observed. The 5,000 base images were collected and annotated manually and an augmentation method was used to increase the training dataset size.

The same 1,000 image dataset containing images from various environmental conditions was used for testing. One of the detectors tested produced good results, with good precision and high detection rates, only missing a low number of vehicles across the testing dataset. To validate its performance, it was tested on an independent dataset, consisting of images from dash cam videos. The detector produced good results on that dataset as well, demonstrating its capabilities as an effective vehicle detector.

The main finding of this testing process was that an efficient vehicle detector can be developed without the use of vast amounts of data or complex architectures. It is of course expected that in a Neural Network, the increase in training data will improve results and detection performance, though even with small datasets, vehicle detectors aimed at specific applications can be produced.

## **6 Application of the Detector and Discussion**

### **6.1 Introduction**

This chapter discusses the outcomes of the development process and examines the performance and application of the highest-performing vehicle detector presented in Chapter 5. The second section discusses the detectors developed using the two different approaches while sections 3 and 4 examine the performance of the detector which is the most suitable for application.

The highest-performing CNN-based detector (Model 6, AP = 85.87%, high recall rate) is examined with respect to its application for developing different safety surrogate measures. It should be noted that any potential application involving vehicle detection shall have the lowest miss rate and lowest number of generated False Positive (FP) detections. The detector developed using the traditional image processing approach (HOG+SVM combination) is excluded from this test, as it under-performed and is therefore completely unsuitable for any kind of application.

### **6.2 Discussion on vehicle detection models**

The main part of the thesis was dedicated to exploring two approaches to vehicle detection and identifying the best solution for real-time implementation. Initially, the combination of HOG features and SVM classification that has been successfully used before for pedestrian and vehicle detection was modified in an attempt to improve its efficiency and run-time performance. Instead of using an established method for generating vehicle candidate locations such as sliding windows (brute force approach that generates numerous proposals), EdgeBoxes or Selective Search (that reduce the number of generated proposals), ROI generation was based on the output of HOG feature extraction. The HOG feature vector is processed to identify regions that contain strong horizontal elements. The concept behind this was to use high-level features for ROI generation, coming from a process commonly used in the classification process.



The results from this first method indicate that the developed vehicle detector does not perform adequately to be considered for any kind of application. In many cases, the ROI generation method does not generate the appropriate candidate locations and therefore the results are poor with many missed detections. The recall rate for this detector is low, at 20% while the number of FP detections is also high. The detector's main issue is the lack of effective vehicle-environment segmentation, resulting in erroneous candidate locations. This problem is particularly apparent in environments with poor lighting conditions or complex and cluttered environments such as urban areas, where objects are either difficult to discern or exhibit strong horizontal edges respectively. The main finding from the use of this particular method to generate candidate locations is that using this kind of aggregate information (horizontal edges based extracted from HOG bins), coming from a previous processing step (HOG vector) is not sufficient to produce a working result. Too much information is lost in the process and without a doubt, an additional visual cue is required to generate vehicle candidate locations robustly.

To test the CNN-based approach, six network models based on the Faster RCNN architecture were developed. The choice of architecture is justified by the all-around good performance of Faster R-CNN, which has no inherent disadvantages (with other architectures such as YOLO or SSD struggling with small objects and producing localisation errors). Compared to other architectures, it is slower in run time, but can be modified to achieve real-time performance. The first model served as a base upon which the other five were built. Each of the models tested introduces another element that affects detection performance. That is either modifying the network size or modifying a parameter that impacts performance and detection speed. The common elements across all tested models are:

- The models were trained end to end, without making use of pre-trained network structures
- Limited amount of training data available (5,000-20,000 images for each model), which is low in comparison to existing databases. Data augmentation was used to overcome this limitation

- The RPN structure is common across all examined network models and is based on the Faster RCNN architecture. The feature extraction part of the CNN is different (different number of layers), but for all models, it is not as deep as other CNN detectors.

The results for each model present the effect each of the modified parameters has on detection performance:

- Using batch normalisation did not benefit detection performance at all, instead producing inferior results.
- Modifying the output of the RPN network produced improved results by about 5% compared to the base model at 20,000 training images.
- Modifying the learning rate for training the network without the use of an optimiser can have unexpected results. In this case, the network did not manage to converge and failed to produce a working detector.

The highest performing detector ( $AP = 85.87\%$ , low miss rate) is a simple structure that performed well on the test dataset. No inherent problems were identified, with the detector managing to perform well under all examined environments (motorway, urban, rural), in both simple and complex scenes containing multiple vehicles. The results were validated on an independent dataset, thus proving the effectiveness of this simple structure in a specific task.

The results demonstrate that, for a given task (in this case vehicle detection), a simple well-structured network can perform better even when the available data is limited. The dataset size deficiency can be overcome by augmenting the dataset using the *translation* method proposed here (see section 4.4 on Data collection and pre-processing). The research here does not overlook the benefit of using deeper structures or transfer learning for complex tasks such as multi-class identification. Instead it provides an alternative solution to vehicle detection, especially when there are limitations in the quantity and quality of data.

### **6.3 Discussion on the processing time**

The highest performing CNN-based detector (Model 6) was tested on an Intel i7 desktop computer, equipped with 16GB of RAM and an Nvidia GTX1080 GPU. The detector was developed and tested on Matlab 2018a without any software optimisations. The use of a fast GPU accelerator is required for training and testing a Deep Neural Network (DNN) and currently, even top of the line CPUs are unable to perform as good as even moderately powered GPUs. CNNs' training and inference are processes made up of numerous simple mathematical operations that take advantage of the hundreds of simple processing cores in a GPU. By comparison, CPUs contain few very complex cores, designed to handle complex tasks, making them unsuitable for CNN training and inference.

The CNN detector was tested on a series of short video streams. Processing the video files includes converting the video into frames, running the vehicle detector, saving the output frames and creating an output video file. The reported processing time therefore, does not include the detection process solely, but every associated image processing operation. The resolution of the video files is 512x512 pixels (same as the training data) but, similarly to the testing process, the search window is limited in size (height of the search area is 300 pixels) without any impact on the detection accuracy. The effective search area does not include the bottom and top parts of the image (vehicle and part of the sky) where no vehicles are located. The size of the search window can be seen in Figure 6-1 below:



**Figure 6-1: CNN detector search space**

The average processing time for each image frame in the videos is 90ms. The system frame rate is therefore around 11.1 FPS.

**Table 6-1: Run-time performance 1**

	<b>Average time per frame</b>	<b>System FPS</b>
<b>Model 6 (16 layer CNN)</b>	90ms	11.1 FPS

There are many possible ways to improve run-time performance. By adjusting the search space just slightly by 50 pixels in height, the improvement in speed is around 7%, bringing the system's speed at 12FPS. The new search space can be seen in Figure 6-2 below. The road is still completely covered by the area where the detector operates:



**Figure 6-2: Reduced search space**

**Table 6-2: Run-time performance 2**

	<b>Average time per frame</b>	<b>System FPS</b>
<b>Model 6 (16 layer CNN)</b>	84ms	<b>12 FPS</b>

It is obvious from the results that the developed detector is not the fastest CNN-based detector, especially compared to detectors developed using the YOLO (Redmon et al., 2016) or SSD (Liu et al., 2016) models. However, the result is a detector that performs well with good accuracy and low miss rate at a good speed for implementation. In addition, there is room for improvement in speed, given that the code is unoptimised and written in a high-level language such as Matlab as opposed to using a dedicated framework for CNNs such as Caffe (Jia et al., 2014).

Ways to improve run-time performance:

- Transfer detector to a dedicated framework for Neural Networks, such as TensorFlow, Caffe or Keras. These frameworks contain tools and software libraries useful for the development of Neural Networks, offering the opportunity to develop advanced models in an efficient way.
- Implement adaptable search space process to speed up detection. By identifying the operational conditions, an algorithm could dynamically modify the search window to reduce the computations required for fast detection. A fixed size search window was implemented in the present thesis.
- Optimise processes not directly related to the CNN detection (image pre-processing). Unoptimised software code can impact the run-time operation of the vehicle detector. Streamlining the pre-processing stage can yield improvements in run-time operation.
- Transfer the developed vehicle detector to a platform optimised for Intelligent Vehicle applications (FPGA platforms or embedded systems). FPGA platforms are flexible hardware configurations that can be optimised to run CNNs for vision based applications. Exploring these hardware options is not in scope of the present work.

## **6.4 Potential application of the detector**

This section explores the possibility of using the developed vehicle detector to estimate the longitudinal distance from other vehicles for use in a safety application such as Adaptive Cruise Control (ACC) or Autonomous Emergency Braking (AEB). Accurate estimation of range offers the possibility to calculate safety surrogate measures such as TTC (Time to Collision) or distance to collision which is a key variable in most vehicle-based active safety systems.

To estimate the distance, a simple perspective geometry calculation takes place. It does not offer the accuracy of more advanced methods or systems, but it serves as a good indication of the errors associated with range measurement using a single monocular camera. Such camera systems do not possess the capability for direct

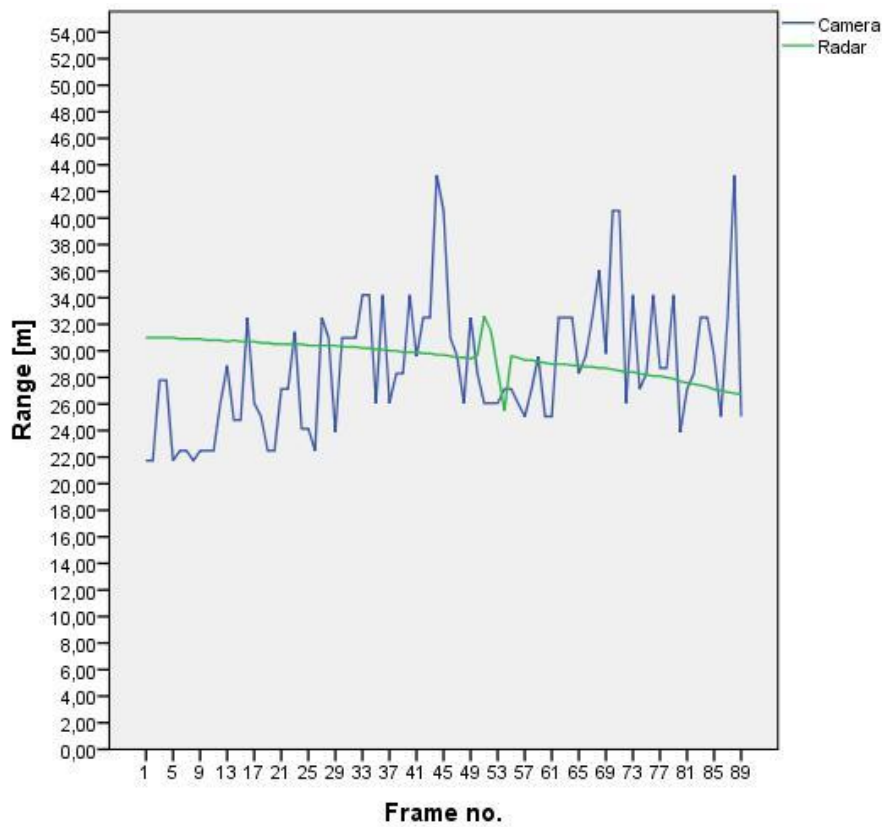
range estimation that other systems such as radar or stereo cameras have and therefore, advanced methods are required to compensate for this deficiency.

The system is tested in short video clips (5-10 seconds each, up to 150 frames) where the ego-vehicle travels on the M1 motorway. The distance between the ego vehicle and the target vehicle varies and changes through time. During the same data collection trips, radar data were also collected. This radar data is used here as ground truth, to compare the accuracy of the camera and radar systems. The camera and radar data were synchronised (15FPS and 15Hz respectively) to provide an accurate comparison.

**Table 6-3: Range estimation for camera and radar**

	<b>Video 1</b>	<b>Video 2</b>	<b>Video 3</b>	<b>Video 4</b>
<b>Radar range (avg)</b>	29.53m	29.99 m	27.45 m	27.92 m
<b>Camera range (avg)</b>	28.79 m	27.20 m	28.13m	55.59 m
<b>Camera range (min)</b>	21.74m	19.19 m	14.85 m	40.55 m
<b>Camera range (max)</b>	43.23m	46.28 m	41.28 m	71.5 m
<b>Max deviation</b>	16.43m	17.48 m	13.68 m	44.6 m

**Video clip 1** – Both target and ego vehicles travel in the middle lane of the motorway.

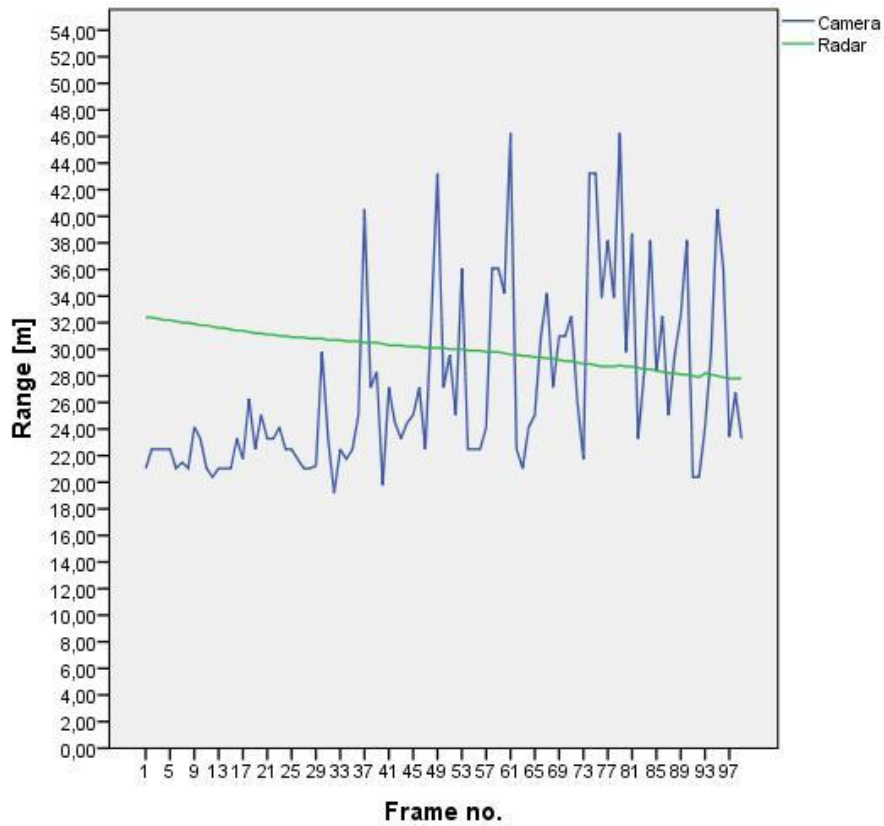


**Figure 6-3: Range estimation - Video 1**

The distance between ego and target vehicle decreases at a slow rate (the longitudinal distance measured by the radar is used as ground truth). The range estimated by the camera is not consistent but deviates significantly from the radar readings. The maximum difference between camera and radar is 16.43m.



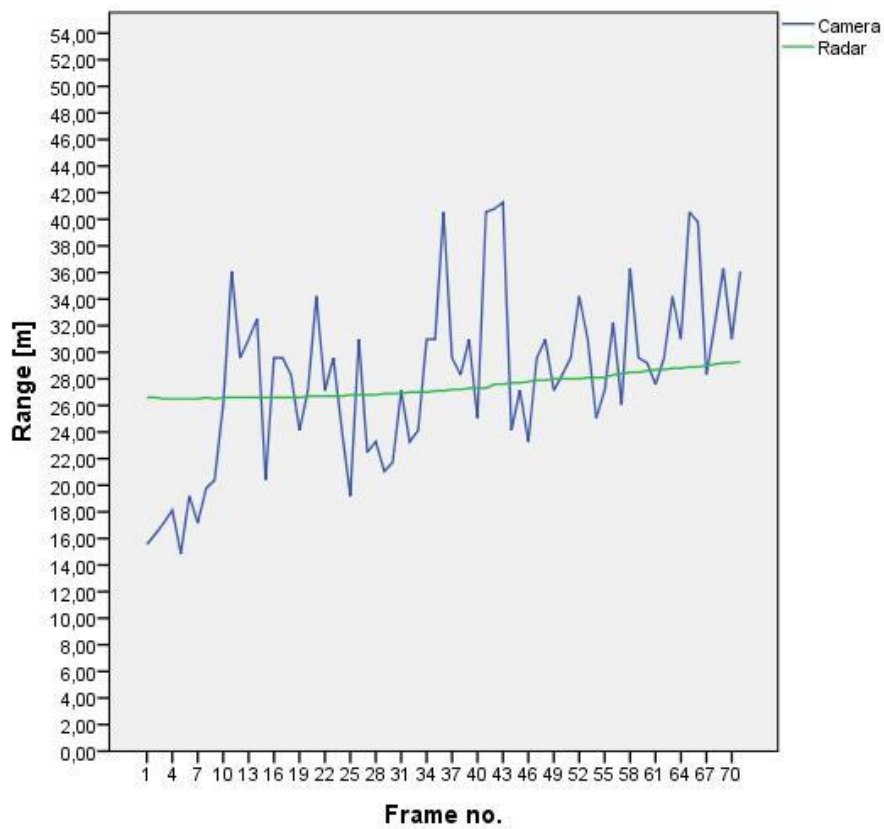
**Video 2** – Target and ego vehicles in the middle lane of the motorway. Target vehicle decelerates and changes lane.



**Figure 6-4: Range estimation - Video 2**

The distance between ego and target vehicle decreases at a slow rate. The range estimated by the camera is not consistent but deviates significantly from the radar readings and tends to overestimate the longitudinal distance. The maximum difference between camera and radar is 17.48m.

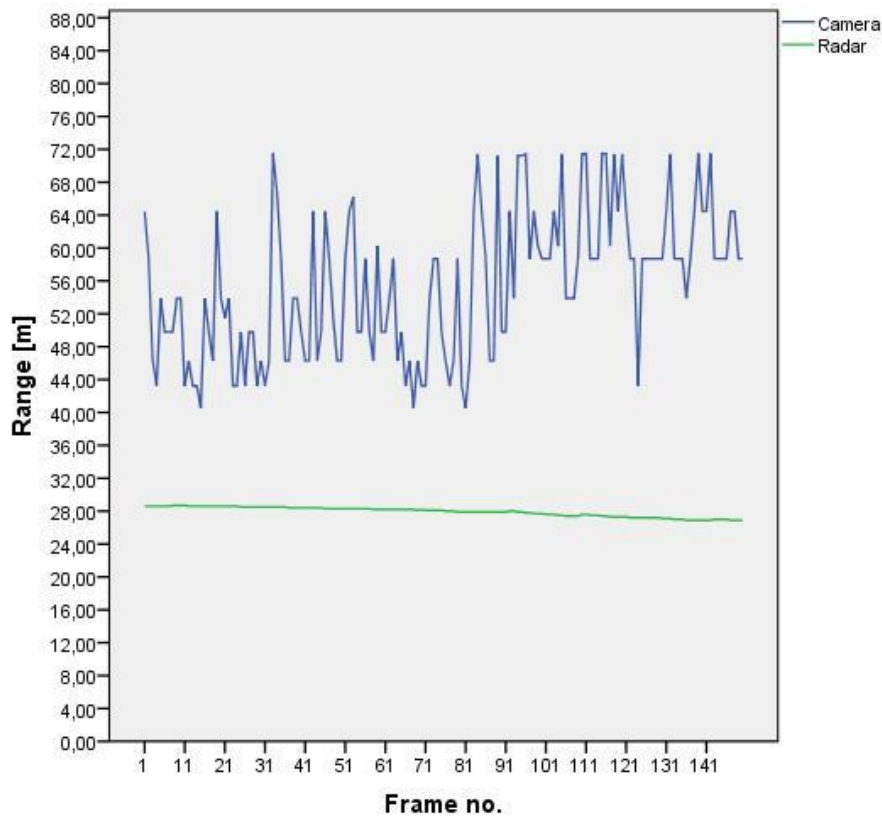
**Video 3** – Ego vehicle in the middle lane of the motorway. Target vehicle on the right lane increasing speed.



**Figure 6-5: Range estimation -Video 3**

The distance between ego and target vehicle increases at a slow rate. The distance between the two vehicles is overestimated by the camera. The maximum difference between camera and radar is 13.68m.

**Video 4** - Ego vehicle follows a target vehicle in the middle lane of the motorway.



**Figure 6-6: Range estimation - Video 4**

The distance between ego and target vehicle remains relatively constant. The distance between the two vehicles is greatly overestimated by the camera. The maximum difference between camera and radar is 44.6m.

It is clear from the graphs that the camera system does not produce consistent range measurements. While it is expected that a camera would be less accurate compared to the radar system, there are a lot of inconsistencies in the camera's produced measurements. The reason for this is the various limitations of measuring longitudinal distance with a monocular camera using only simple homography:

- The camera system returns the centre location of the bottom of the detector's bounding box. It is only able to accurately compute distances along the surface of the road. If the vehicle detector successfully detects the vehicle

ahead but the produced bounding box is slightly higher, the vehicle is assumed to be further away than it actually is. This is the reason for the spikes in the graphs. So while the detection is successful, the range measurement is not. Computation of an arbitrary location in 3D space requires a stereo camera system or another type of sensor (such as radar).

- The conversion between the camera coordinate system and the real world coordinates assumes a flat road. Roads that are not flat introduce errors in the computation, with the error increasing the higher the distance.
- The range measurement system would produce more accurate results by using a more sophisticated method, either accounting for the error in measurements and including the distance rate change in the computation or by using tracking (for example a Kalman filter) to estimate the actual distance instead of measuring range from a single snapshot.

The difference in range estimation between radar (used as ground truth) and camera is simply too large. The system cannot be used for TTC calculation as is; and cannot be used for anything else other than as a sanity check (to actually see that distances that make some sense can be calculated). The method is simple and the limitations too many for this to be used in a safety application.

## **6.5 Discussion summary**

This section discussed the results presented in Chapter 5 and tested the suitability of the highest performing detector for real-time application. Regarding the detection performance of the examined detector, the metrics used to evaluate CNN-based detectors indicate a very good model (high AP, low miss rate). The employment of a simple architecture, trained end to end with a limited amount of data manages to produce results that are comparable with state of the art detectors for this specific task (vehicle detection). The discussion of run-time performance indicated that the produced vehicle detector, while not as fast as other state of the art object detectors (YOLO, SSD) still manages to perform well despite a lack of significant optimisations. The gap in speed between the proposed detector and other models is due to the structure of the Faster R-CNN architecture and the way detections are

generated. Any discussion about performance should acknowledge the limitations of each architecture and that there is a trade-off between accuracy and speed. The potential for improvements is there though, to achieve an improved detection speed using the proposed model. Finally, the simple distance calculation highlighted the inherent issues with estimating distance using a monocular camera. The lack of any error correction results in significantly inferior results compared to the more accurate camera system.

## 7 Conclusions

### 7.1 Introduction

Vehicle collisions are one of the most significant problems in transport, as they are one of the leading causes of deaths and injuries around the world. Collisions are attributed on environmental, vehicle and human factors, with human errors (either recognition, decision or performance errors) being the dominant causation factor of accidents.

To mitigate the effect of human errors in accidents, the automotive industry is moving towards removing the human element from driving. Research from both industry and academic institutions is heavily invested in bringing every necessary component (hardware, software, methods) together to reach that stage where a human driver is not required to handle any driving-related task. Either by revolution (going from 0 to full autonomy via high-tech solutions) or evolution (slowly improving and automating driving functions until full autonomy is achieved), soon human driving behaviour will no longer be a liability and a threat to safety.

A safe trip with an Autonomous Vehicle is ensured by the presence of an effective CAS, which ensures all potential threats ahead of the vehicle are correctly identified and every danger avoided. Usually, AVs use multiple sensors to scan the surrounding environment, collect information and detect targets; and that means that the hardware cost and system complexity is high. Data from multiple sensors (active and passive) need to be collected, processed and fused together to confidently produce an accurate detection along with its classification as pedestrian, vehicle or other object.

This research focuses on the task of vehicle detection and attempts to produce an accurate vehicle detector based on the data coming from a single low-cost monocular camera. Current literature regarding the subject of object detection using vision systems favours two approaches. The first one follows traditional image processing principles where specific visual cues identifying potential targets are sought for in an image and then a classifier trained with relevant data is used to verify each object's class. The second approach uses a specific type of Neural Network modified to

process image data, the CNN, to unify the detection pipeline (ROI generation and classification) in a single process. While this approach was introduced some years ago (LeCun et al., 1989), the high computational requirements and need for large amounts of data meant that only the last few years has it been made a viable option for object detection, with a rise in computing performance and deep learning making it possible.

The data used for this research were collected using Loughborough University's instrumented vehicle. Both training and testing data were manually annotated with ground truth labels while radar data were used as ground truth for the camera's range measurements. The relevant literature on object detection does not usually follow this approach. Particularly for CNN-based detection, readily available datasets are preferred over processing raw data; in addition to using transfer learning (pre-trained CNN networks) as opposed to training a new network from scratch. The reason for this is the effort required to collect and manually annotate large amounts of data and also possibly, a concern that the amount of data collected will not be sufficient to efficiently train a network end to end.

This PhD can serve as a guide to developing object detectors using end to end training. It demonstrates that, for specific applications, complex structures or the reliance on out of the box solutions are not a requirement. An alternative and efficient solution to vehicle detection, especially when there are limitations in the data, is proposed here.

Additionally, the developed detector can be used as a base for more complex tasks such as TTC calculation for a CAS. The simple calculation performed here gives only a rough estimate of range, but it can be improved by accounting for range error and indicates that a complete CAS is possible using a camera sensor.

## 7.2 Achieving the aim and objectives

To achieve the aim of this study, which is the development of a reliable vehicle detection system based on a monocular camera, the following five objectives were necessary to be accomplished. This section presents these objectives and discusses how they were met in the thesis.

**Objective 1:** Investigate existing vision-based approaches to vehicle detection.

An extensive and critical literature review was conducted in Chapter 2 focusing on vehicle detection using a monocular camera. More specifically, the review explores both the traditional image processing approach to detection as well as the CNN-based approach with the stages and findings of each approach being discussed.

**Objective 2:** Identify the methods most likely to produce the desired performance, given the limitations of existing methods.

Based on the literature review, the methods most likely to produce the highest detection performance were identified. Both approaches to detection are explored in this thesis by developing different detectors, one based on the mature method of using HOG features and SVM classification in an attempt to optimise it; the second based on high-performance CNN architecture. Instead of following the traditional approach though, where pre-trained networks are used, the detector is trained end to end with own collected data.

**Objective 3:** Collect, synthesize and process the data required to develop a vehicle detector

The necessary data for training and testing the developed detectors were collected using Loughborough University's instrumented vehicle. The data used in this research are camera data representative of traffic in various operational environments (urban roads, motorway, rural roads) as well as radar data used as ground truth for range measurements. Processing the data included manual annotations and ground truth labels for image data as well as using data augmentation to increase the size of



the dataset. The process of collecting and processing the data is described in Chapter 4 of this thesis.

**Objective 4:** Develop a robust and high-performing vehicle detector based on a low-cost monocular camera

The operation of the HOG-based detector is described, and a new ROI generation method is proposed for use with a traditional SVM classifier. For the CNN-based approach, six models are developed in total, each modified in structure or parameters in order to determine the highest performing vehicle detector. The development process of the camera-based detectors is detailed in Chapter 3 (Methodology).

**Objective 5:** Assess and validate the performance of the developed vehicle detector

The performance of all developed detectors was assessed using appropriate metrics established in the literature. The goal was to determine which detector performed more effectively in terms of detection rate and number of missed or false detections; this detector was further tested to assess its suitability for real time applications (Chapters 5 and 6).

### **7.3 Contribution to knowledge**

The results of this work have produced outcomes that can be used as insight when developing CA systems. The main contributions to knowledge of this research are:

#### 1. Network complexity – performance relationship

This research has examined the role of network depth and end to end training in feature extraction for CNN based detection. Relevant research on the topic of object detection relies on deep network structures for feature extraction. The outcomes of this analysis indicate that a structure with relatively few layers and filters can perform well when faced with a specific task. More specifically, the results indicate that enough descriptive information is generated by the network to correctly classify vehicles in images. This does not contradict the common practice of using pre trained

networks for feature extraction, but merely indicates that another option is available for performing a given task.

Looking forward, this finding shows the potential of using multiple CNNs for different tasks, each assigned to perform a specific function. To provide an example, in a vehicle system, different CNNs could be assigned each of the following tasks: object detection, navigation and range estimation, monitoring and tracking. This is an approach gaining traction in hardware systems designed to handle multiple CNNs concurrently, where the performance of each CNN in its task does not affect another handling a different task.

## 2. Significance of dataset size on detection performance

One of the most significant factors affecting CNN performance is the size of the training dataset. The classification/detection performance of deep NNs increases as the number of observations increases. In comparison, classifiers such as SVMs do not benefit from an increase in data after a point. For this research, a dataset limited in size was used to train the CNN network. 5,000 images were used initially; data augmentation and iterative training allowed each CNN model to reach 20,000 images. Compared to databases such as ImageNet, containing around 374,000 images for the vehicle class category and even smaller databases such as Stanford University's Cars dataset containing over 16,000 training images (8,144 used for training), the size of the initial dataset in this study is considered small.

Still, the final detector model developed manages to perform well (AP = 85.87%, low miss rate) and complete its task successfully. The increase in detection performance occurs during the jump for 5,000 to 10,000 images. After this point, performance gains are smaller. This indicates a limit to the effectiveness of data augmentation. It should be expected that after the 20,000 image step, the gains will be minimal without the use of new data.

The detection performance of the last model indicates a good balance between dataset size and network depth and is something to consider in the design of CNN

architectures in the future. An imbalance between dataset size and network can lead to problems such as overfitting when the network is deep and the dataset small.

## 7.4 Study limitations

This research presented in this thesis does not come without its limitations. The most important of them are outlined below:

- **Dataset size limitations:** The data used in this study come from several data collection runs using an instrumented vehicle. Collecting data this way provides practically unlimited raw data (camera and radar) that can be used. However, the need to process them and provide the required ground truth information (positive and negative training samples for SVM training, ground truth coordinates and labels for CNN training) can be an extremely daunting task, one that practically has no end as the influx of data is constant. While SVM classifier performance is not affected by the amount of data after a point, the same does not apply for CNN training. NN performance benefits with the addition of training data, which is why NNs are used to process big data. Automatic labelling is possible in some cases, though in this study it proved to be ineffective. The original training dataset size was 5,000 images with data augmentation used to artificially increase the size to 20,000 images. This limit was selected due to concerns that after this point, the same images would be fed to the network.
- **Data quality:** Image data was grayscale and in many cases in poor lighting conditions. This introduces some constraints as to which methods can be used to process the images. It also means that a robust detector trained under these limitations should perform better under more favourable conditions.
- **Performance of transfer learning approaches:** The performance of the developed CNN based detector was not compared to a transfer learning method. Using the same dataset, it would be possible to fine-tune a deeper pre-trained network for feature extraction. However, the focus of this

research was end to end training and the produced vehicle detector produced good results nonetheless.

- **Detector transferability:** The performance of the detector was evaluated on an independent dataset collected from driving trips. The detector performance has not been evaluated using the benchmarking challenges that are available, an example of which is the Pascal VOC challenge (Everingham et al., 2010). Benchmarking the detector in the vehicle class category of such a challenge would enhance the results and give a better picture of the detector's potential. It has to be stressed however, that the detector has been tested with independent dash cam videos with success, which is a good indicator of its performance in a CA system. By comparison, the images contained in such benchmarking challenges are not directly relevant to vehicle detection in real world scenarios.
- **Optimal CNN architecture:** The search for an optimal CNN architecture is a non-trivial task. Experimenting with layer depth and width, layer sequence and parameter optimisation is essentially a never ending process. This study identified a layer structure and a set of parameters that, given the limitations in dataset size, offers a good balance in detection performance and run-time speed. It does recognise however, that there is potential to improve performance by further adjusting network parameters.
- **Range estimation:** Range estimation is based on simple perspective geometry from the calibrated monocular camera. This introduces significant deviation from the radar measurements used as ground truth. For this study, range estimation is mainly used as sanity check, to identify in which cases the estimation is particularly problematic and where the measured distance is closer to the real value. Introducing a more complex calculation or a tracking function would increase the accuracy of the measurement.

- **Generated bounding box size:** The CNN detector generates bounding boxes of variable size along with class probabilities. While this is not an issue for vehicle detection, it introduces additional error during the range estimation process as the position of the box affects the measurement. It is possible that the introduction of fixed size bounding boxes will increase the accuracy of the range estimation.
- **Software limitations:** The detector models were all trained and tested on a high-level language such as Matlab. Irrespective of hardware improvements, optimising the code and using a dedicated framework should improve run-time performance.

## 7.5 Extensions and suggestions for future research

The work presented in this thesis can contribute to the further improvement of object detection methods. While the final detector developed here is task specific to vehicle detection, the approach to its development can be extended to generalised object detection. Building robust detectors using end to end training with the use of data from a low-cost sensor system is possible, even with limited amounts of data. Considering the limitations presented in the previous section, there are some improvements that can be implemented to improve overall performance and functionality.

The current detector is dedicated to vehicle detection and the proposed structure suits the particular application well. It would be useful to add functionality by using this simple network structure to detect other object classes such as pedestrians and cyclists. This would further validate the approach of using simple CNNs and not rely completely on transfer learning.

Improving detection performance would require additional training data. Since manually processing data to train a classifier or CNN is a demanding task, the option to use simulated data could be explored. The advance in graphics engines during the last few years indicates that simulated data can now be used for this purpose, with

the potential to solve the issue of generating the large amounts of data required for deep learning.

Range estimation can be improved by accounting for distance error in the calculation. Calculating the distance error rate or implementing a tracking filter to limit extreme errors can significantly improve the measurements and provide a robust alternative to using additional sensors for this task.

Run-time performance can be improved further by implementing various methods. Optimising the search area dynamically instead of using a static window would yield significant speed benefits. Identifying the bottlenecks and improving the structure of the RPN (proposal network) is also a priority, as this stage is one of the most processor intensive of the detection process. One of the most significant upgrades would be to transfer the developed CNN detector onto a hardware platform optimised for CNN inference. Such lower power platforms are becoming available and their combination with low-cost sensors such as cameras would drive mass adoption of CA systems.

Finally, optimising and assessing a CNN detector's performance is a constant process and should continue beyond this study. Through this, it will become possible to develop a high-integrity system, for use in any kind of safety critical ADAS application.

## References

- Achanta, R., Shaji, A., Smith, K., Lucchi, A. (2010) SLIC superpixels. Lausanne.
- Alencar, F.A.R., Rosero, L.A., Filho, C.M., Osorio, F.S., Wolf, D.F. (2015) Fast Metric Tracking by Detection System: Radar Blob and Camera Fusion. In 2015 12th Latin American Robotics Symposium and 2015 3rd Brazilian Symposium on Robotics (LARS-SBR). pp. 120–125.
- Baek, J.W., Kwon, K.-K., Lee, S.-I. (2014) Mono-camera based Vehicle Detection Using Effective Candidate Generation. In IEEE International Conference on Software Engineering. Hyderabad, India, pp. 1–2.
- Ballesteros, G., Salgado, L. (2014) HISTOGRAMS OF ORIENTED GRADIENTS FOR FAST ON-BOARD VEHICLE VERIFICATION. In International Conference on Image Processing (ICIP). Paris, pp. 1638–1642.
- Belongie, S., Malik, J. (2000) Matching with shape contexts. In in Proceedings - IEEE Workshop on Content-Based Access of Image and Video Libraries, CBAIVL 2000. Hilton Head Island, South Carolina, pp. 20–26.
- Bensrhair, A., Bertozzi, M., Broggi, A., Michc, P., Mousset, S., Toulminet, G. (2001) A Cooperative Approach to Vision-based Vehicle Detection. In 2001 IEEE Intelligent Transportation Systems Conference Proceedings - Oakland (CA), USA - August 25-29, 2001 A. pp. 207–212.
- Berlin Team, Gunnarsson, K., Simon, M., Wiesel, F., Ruff, F., Wolter, L., Zilly, F., Ganjineh, T., Sarkohi, A., Ulbrich, F., Latotzky, D., Jankovic, B., Hohl, G., Wisspeintner, T., May, S., Pervölz, K., Nowak, W., Maurelli, F., Dröschel, D., Iais, F.G., Augustin, S. (2007) Spirit of Berlin : An Autonomous Car for the DARPA Urban Challenge Hardware and Software Architecture. , 1–25.
- Bertozzi, M., Bombini, L., Cerri, P., Medici, P., Antonello, P.C., Miglietta, M. (2008) Obstacle detection and classification fusing radar and vision. In in Intelligent Vehicles Symposium (IV). Eindhoven, pp. 608–613.

- Betke, M., Haritaoglu, E., Davis, L.S. (1997) Highway scene analysis in hard real-time. In IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC. Boston, pp. 812–817.
- Bishop, C.M. (2007) Pattern Recognition and Machine Learning. M. Jordan, J. Kleinberg, & B. Scholkopf, eds. Cambridge: Springer.
- Broggi, A., Cerri, P., Debattisti, S., Laghi, M.C., Medici, P., Panciroli, M., Prioletti, A. (2014) PROUD – Public ROad Urban Driverless test : architecture and results. In IEEE Intelligent Vehicles Symposium (IV). Dearborn, Michigan, pp. 648–654.
- Burlet, J., Dalla Fontana, M. (2012) Robust and efficient multi-object detection and tracking for vehicle perception systems using radar and camera sensor fusion. IET and ITS Conference on Road Transport Information and Control (RTIC 2012), 24–24.
- Canny, J. (1986) A computational approach to edge detection. IEEE transactions on pattern analysis and machine intelligence. 8(6), 679–698.
- Chan, Y.-M., Huang, S.-S., Fu, L.-C., Hsiao, P.-Y., Lo, M.-F. (2012) Vehicle detection and tracking under various lighting conditions using a particle filter. IET Intelligent Transport Systems. 6(1), 1.
- Chavez-garcia, R.O., Aycard, O. (2015) Multiple Sensor Fusion and Classification for Moving Object Detection and Tracking. IEEE Transactions on Intelligent Transportation Systems, 1–10.
- Chen, Yc, Su, T., Lai, S. (2013) Efficient vehicle detection with adaptive scan based on perspective geometry. In IEEE International Conference on Image Processing. Melbourne, pp. 3321–3325.
- Chen, Yen-lin, Chen, Yuan-hsin, Chen, C., Wu, B. (2006) Nighttime Vehicle Detection for Driver Assistance and Autonomous Vehicles. In 2006 IEEE Proceedings of the 18th International Conference on Pattern Recognition. Hong Kong, pp. 18–21.



Chen, Zhiqian, Chen, K., Chen, J. (2013) Vehicle and Pedestrian Detection Using Support Vector Machine and Histogram of Oriented Gradients Features. 2013 International Conference on Computer Sciences and Applications, 365–368.

Cheon, M., Lee, W., Yoon, C., Park, M. (2012) Vision-Based Vehicle Detection System With Consideration of the Detecting Location. IEEE Transactions on Intelligent Transportation Systems. 13(3), 1243–1252.

Christiansen, R.H., Hsu, J., Gonzalez, M., Wood, S.L. (2018) Monocular vehicle distance sensor using HOG and Kalman tracking. Conference Record of 51st Asilomar Conference on Signals, Systems and Computers, ACSSC 2017. 2017-Octob, 178–182.

Cortes, C., Vapnik, V. (1995) Support Vector Networks. Machine Learning. 20(3), 273–297.

Dai, B., Fu, Y., Wu, T. (2007) A vehicle detection method via symmetry in multi-scale Windows. ICIEA 2007: 2007 Second IEEE Conference on Industrial Electronics and Applications, 1827–1831.

Dai, J., Li, Y., He, K., Sun, J. (2016) R-FCN: Object Detection via Region-based Fully Convolutional Networks. In NIPS'16 Proceedings of the 30th International Conference on Neural Information Processing Systems. Barcelona, pp. 379–387.

Dalal, N., Triggs, B. (2005) Histograms of Oriented Gradients for Human Detection. CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1, 886–893.

Deng, Y., Liang, H., Wang, Z., Huang, J. (2014) An integrated forward collision warning system based on monocular vision. 2014 IEEE International Conference on Robotics and Biomimetics, IEEE ROBIO 2014, 1219–1223.

Dhanachandra, N., Manglem, K., Chanu, Y.J. (2015) Image Segmentation Using K-means Clustering Algorithm and Subtractive Clustering Algorithm. Procedia Computer Science. 54, 764–771.

Di, Z., He, D. (2016) Forward Collision Warning system based on vehicle detection and tracking. In Proceedings - 2016 International Conference on Optoelectronics and Image Processing, ICOIP 2016. Warsaw, pp. 10–14.

Eskandarian, A. (2012) Handbook of intelligent vehicles. Second Edi. A. Eskandarian, ed. London, UK: Springer.

Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A. (2010) The PASCAL Visual Object Classes (VOC) Challenge. International Journal of Computer Vision. 88(2), 303–338.

Fan, Q., Brown, L., Smith, J. (2016) A closer look at Faster R-CNN for vehicle detection. IEEE Intelligent Vehicles Symposium, Proceedings. 2016-Augus(Iv), 124–129.

Filipowicz, A., Liu, J., Kornhauser, A. (2017) Learning to Recognize Distance to Stop Signs Using the VirtualWorld of Grand Theft Auto 5. Transportation Research Board, 96th Annual Meeting. (January 2017), 1–16.

Fisher, R., Perkins, S., Walker, A., Wolfart, E. (2003a) Feature Detectors - Sobel Edge Detector. [online]. Available from: <http://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm> [Accessed August 14, 2017].

Fisher, R., Perkins, S., Walker, A., Wolfart, E. (2003b) Pixel Connectivity. [online]. Available from: <http://homepages.inf.ed.ac.uk/rbf/HIPR2/connect.htm> [Accessed May 16, 2019].

Fodor, I.K. (2002) A survey of dimension reduction techniques. Livermore, CA.

Freeman, W.T., Roth, M. (1995) Orientation histograms for hand gesture recognition. In International Workshop on Automatic Face and Gesture Recognition. Zurich, pp. 296–301.

Freund, Y., Schapire, R.E. (1999) A Short Introduction to Boosting. Journal of Japanese Society for Artificial Intelligence. 14(5), 771–780.

Gao, Y., Guo, S., Huang, K., Chen, J., Gong, Q., Zou, Y., Bai, T., Overett, G. (2017) Scale optimization for full-image-CNN vehicle detection. In IEEE Intelligent Vehicles Symposium, Proceedings. Redondo Beach, pp. 785–791.

Garcia, F., Cerri, P., Broggi, A., de la Escalera, A., Armingol, J.M. (2012) Data fusion for overtaking vehicle detection based on radar and optical flow. 2012 IEEE Intelligent Vehicles Symposium, 494–499.

García, F., Martín, D., De La Escalera, A., Armingol, J.M. (2017) Sensor Fusion Methodology for Vehicle Detection. IEEE Intelligent Transportation Systems Magazine. 9(1), 123–133.

Geiger, A., Lenz, P., Urtasun, R. (2012) Are we ready for autonomous driving? the KITTI vision benchmark suite. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Providence, pp. 3354–3361.

Girshick, R. (2015) Fast R-CNN. In Proceedings of the IEEE International Conference on Computer Vision. pp. 1440–1448.

Girshick, R., Donahue, J., Darrell, T., Malik, J. (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. pp. 580–587.

Godil, A., Bostelman, R., Shackelford, W., Hong, T., Shneier, M. (2014) Performance Metrics for Evaluating Object and Human Detection and Tracking Systems.

Gu, X.F., Chen, Z.W., Ma, T.S., Li, F., Yan, L. (2017) Real-Time vehicle detection and tracking using deep neural networks. 2016 13th International Computer Conference on Wavelet Active Media Technology and Information Processing, ICCWAMTIP 2017, 167–170.

Haddon, W. (1980) Advances in the epidemiology of injuries as a basis for public policy. Public health reports (Washington, D.C. : 1974). 95(5), 411–21.

- Han, J., Heo, O., Park, M., Kee, S., Sunwoo, M. (2016) Vehicle Distance Estimation using a mono-camera for FCW/AEB systems. *International Journal of Automotive Technology*. 17(3), 483–491.
- Handmann, U., Kalinke, T., Tzomakas, C., Werner, M., Seelen, W.. (1998) An image processing system for driver assistance. In *IEEE International Conference on Intelligent vehicles*. Stuttgart, pp. 1–6.
- Haselhoff, A., Kummert, A., Schneider, G. (2007) Radar-vision fusion for vehicle detection by means of improved haar-like feature and adaboost approach. In *European Signal Processing Conference*. pp. 2070–2074.
- He, C.H., Lam, K.M. (2018) Fast Vehicle Detection with Lateral Convolutional Neural Network. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. Calgary: IEEE, pp. 2341–2345.
- He, K., Zhang, X., Ren, S., Sun, J. (2016) Deep residual learning for image recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. pp. 770–778.
- He, L., Ren, X., Gao, Q., Zhao, X., Yao, B., Chao, Y. (2017) The connected-component labeling problem: A review of state-of-the-art algorithms. *Pattern Recognition*. 70, 25–43.
- Heisele, B., Ritter, W. (1995) Obstacle detection based on color blob flow. In *Proceedings of the Intelligent Vehicles '95 Symposium*. Detroit, pp. 282–286.
- Hornik, K., Stinchcombe, M., White, H. (1989) Multilayer feedforward networks are universal approximators. *Neural Networks*. 2(5), 359–366.
- Hsu, S. (2018) Vehicle Detection using Simplified Fast R-CNN. In *International Workshop on Advanced Image Technology (IWAIT)*. Chiang Mai, pp. 3–5.
- Hu, Y., He, Q., Zhuang, X., Wang, H., Li, B., Wen, Z., Leng, B., Guan, G., Chen, D. (2013) Algorithm for vision-based vehicle detection and classification. *2013 IEEE International Conference on Robotics and Biomimetics, ROBIO 2013*. (December),

568–572.

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., Murphy, K. (2017) Speed/accuracy trade-offs for modern convolutional object detectors. In Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017. Honolulu, pp. 3296–3305.

Huang, L., Zhe, T., Wu, J., Wu, Q., Pei, C., Chen, D. (2019) Robust inter-vehicle distance estimation method based on monocular vision. *IEEE Access*. 7, 1–1.

International Telecommunication Union (2011) Recommendation ITU-R BT.601-7. Geneva.

Ioffe, S., Szegedy, C. (2015) Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In ICML'15 Proceedings of the 32nd International Conference on International Conference on Machine Learning. Lille, pp. 448–456.

Jazayeri, A., Cai, H., Zheng, J.Y., Tuceryan, M. (2011) Vehicle Detection and Tracking in Car Video Based on Motion Model. *IEEE Transactions on Intelligent Transportation Systems*. 12(2), 1–13.

Jia Deng, Wei Dong, Socher, R., Li-Jia Li, Kai Li, Li Fei-Fei (2009) ImageNet: A large-scale hierarchical image database. 2009 IEEE Conference on Computer Vision and Pattern Recognition, 248–255.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T. (2014) Caffe: Convolutional Architecture for Fast Feature Embedding. In Proceedings of the 22nd ACM international conference on Multimedia. Orlando, pp. 675–678.

Jinhui, L., Meng, Z. (2010) A New Vehicle Detection Algorithm for Real-time Image Processing System. *International Conference on Computer Application and System Modeling. (Iccasm)*, V10-1-V10-4.

Joglekar, A., Joshi, D., Khemani, R., Nair, S., Sahare, S. (2011) Depth Estimation

Using Monocular Camera. *International Journal of Computer Science and Information Technologies*. 2(4), 1758–1763.

Jolliffe, I.T. (2002) *Principal Component Analysis*. Second. New York: Springer.

Kalman, R.E. (1960) A New Approach to Linear Filtering and Prediction Problems 1. *Journal of Basic Engineering*. 82(Series D), 35–45.

Kanjee, R., Carroll, J. (2015) A Three-Step Vehicle Detection Framework for Range Estimation Using a Single Camera. In *IEEE Symposium Series on Computational Intelligence*. pp. 442–448.

Karpathy, A., Li, F.F. (2019a) CS231n Convolutional Neural Networks for Visual Recognition. [online]. Available from: <http://cs231n.github.io/neural-networks-2/>.

Karpathy, A., Li, F.F. (2019b) CS231n Convolutional Neural Networks for Visual Recognition. [online]. Available from: <http://cs231n.github.io/convolutional-networks/> [Accessed April 24, 2019].

Khammari, A., Nashashibi, F., Abramson, Y., Laugeau, C. (2005) Vehicle detection combining gradient analysis and AdaBoost classification. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*. 2005, 1084–1089.

Kim, Huieun, Lee, Y., Yim, B., Park, E., Kim, Hakil (2016) On-road object detection using Deep Neural Network. In *2016 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*. IEEE, pp. 1–4.

Kim, J., Baek, J., Kim, D.Y., Kim, E. (2013) On-Road Vehicle Detection based on Effective Hypothesis Generation. In *IEEE RO-MAN: The 22nd IEEE International Symposium on Robot and Human Interactive Communication*. Gyeongju, pp. 252–257.

Kim, J., Baek, J., Kim, E. (2015) A Novel On-Road Vehicle Detection Method Using  $\pi$ HOG. *Ieee Transactions on Intelligent Transportation Systems*. 16(6), 3414–3429.

Kim, J., Baek, J., Kim, E. (2014) On-road precise vehicle detection system using

ROI estimation. 17th International IEEE Conference on Intelligent Transportation Systems (ITSC), 2251–2252.

Kmiotek, P., Ruichek, Y. (2008) Multisensor fusion based tracking of coalescing objects in urban environment for an autonomous vehicle navigation. 2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, 52–57.

Krizhevsky, A., Hinton, G.E. (2012) ImageNet Classification with Deep Convolutional Neural Networks. In NIPS'12 Proceedings of the 25th International Conference on Neural Information Processing Systems. Lake Tahoe, pp. 1097–1105.

Kuehnle, A. (1991) Symmetry-based recognition of vehicle rears. Pattern Recognition Letters. 12(April), 249–258.

Kuo, Y.C., Pai, N.S., Li, Y.F. (2011) Vision-based vehicle detection for a driver assistance system. Computers and Mathematics with Applications. 61(8), 2096–2100.

Laopracha, N., Thongkrau, T., Sunat, K., Songrum, P., Chamchong, R. (2014) Improving vehicle detection by adapting parameters of HOG and kernel functions of SVM. 2014 International Computer Science and Engineering Conference, ICSEC 2014, 372–377.

LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D., Laboratories, T.B. (1989) Handwritten digit recognition with a back-propagation network. Advances In Neural Information Processing Systems 2 (NIPS).

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P. (1998) Gradient-Based Learning Applied to Document Recognition. In Proceedings of the IEEE. pp. 1–46.

LeCun, Y., Kavukcuoglu, K., Farabet, C. (2010) Convolutional networks and applications in vision. In ISCAS 2010 - 2010 IEEE International Symposium on Circuits and Systems: Nano-Bio Circuit Fabrics and Systems. pp. 253–256.

Lee, C., Kim, H.J., Oh, K.W. (2016) Comparison of faster R-CNN models for object detection. In International Conference on Control, Automation and Systems. pp.

107–110.

Lee, Seung Hyun, Bang, M., Jung, K.H., Yi, K. (2015) An efficient selection of HOG feature for SVM classification of vehicle. In Proceedings of the International Symposium on Consumer Electronics, ISCE. pp. 1–2.

Lee, Tae Young, Oh, J.S., Kim, J.H. (2015) On-road vehicle detection based on appearance features for autonomous vehicles. ICCAS 2015 - 2015 15th International Conference on Control, Automation and Systems, Proceedings. (Iccas), 1720–1723.

Li, X., Guo, X. (2013) A HOG Feature and SVM Based Method for Forward Vehicle Detection with Single Camera. 5th International Conference on Intelligent Human-Machine Systems and Cybernetics. 2(1), 263–266.

Lim, Q., He, Y., Tan, U.X. (2019) Real-Time Forward Collision Warning System Using Nested Kalman Filter for Monocular Camera. 2018 IEEE International Conference on Robotics and Biomimetics, ROBIO 2018, 868–873.

Lin, M., Xu, X. (2006) Multiple Vehicle Visual Tracking from a Moving Vehicle. Sixth International Conference on Intelligent Systems Design and Applications. 2, 373–378.

Lin, T., Zitnick, C.L., Doll, P. (2014) Microsoft COCO: Common Objects in Context. In European Conference on Computer Vision - ECCV 2014. Zurich, pp. 740–755.

Litman, T. (2019) Autonomous vehicle implementation predictions : Implications for Transport Planning. Victoria.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C. (2016) SSD: Single shot multibox detector. In Proceedings of the European Conference on Computer Vision (ECCV). pp. 21–37.

Lowe, D.G. (1999) Object recognition from local scale-invariant features. Proceedings of the Seventh IEEE International Conference on Computer Vision. 2(8), 1150–1157.



Ma, Y., Chen, X., Chen, G. (2011) Pedestrian detection and tracking using HOG and oriented-LBP features. In IFIP International Conference on Network and Parallel Computing (NPC 2011). pp. 176–184.

Mahalanobis, P.C. (1936) On the Generalized Distance in Statistics. In Proceedings of the National Institute of Sciences of India. p. pp.49-55.

Martinez, M., Sitawarin, C., Finch, K., Meincke, L., Yablonski, A., Kornhauser, A. (2018) Beyond Grand Theft Auto V for Training, Testing and Enhancing Deep Learning in Self Driving Cars. Washington.

Matas, J., Sochman, J. (2009) AdaBoost presentation.

McCormick, C. (2003) HOG Person Detector tutorial. [online]. Available from: <http://mccormickml.com/2013/05/09/hog-person-detector-tutorial/> [Accessed January 19, 2017].

Mikołajczyk, A., Grochowski, M. (2018) Data augmentation for improving deep learning in image classification problem. In 2018 International Interdisciplinary PhD Workshop (IIPhDW). Swinoujście, Poland: IEEE, pp. 117–122.

Mukhtar, A., Xia, L., Tang, T.B. (2015) Vehicle Detection Techniques for Collision Avoidance Systems: A Review. IEEE Transactions on Intelligent Transportation Systems. 16(5), 1–21.

Müller, K.R., Mika, S., Rätsch, G., Tsuda, K., Schölkopf, B. (2001) An introduction to kernel-based learning algorithms. IEEE Transactions on Neural Networks. 12(2), 181–201.

Nagi, J., Ducatelle, F., Di Caro, G.A., Cireşan, D., Meier, U., Giusti, A., Nagi, F., Schmidhuber, J., Gambardella, L.M. (2011) Max-pooling convolutional neural networks for vision-based hand gesture recognition. 2011 IEEE International Conference on Signal and Image Processing Applications, ICSIPA 2011, 342–347.

Nair, V., Hinton, G.E. (2010) Rectified Linear Units Improve Restricted Boltzmann Machines. In Proc. 27th International Conference on Machine Learning. Haifa.

Nath, R.K., Deb, S.K. (2010) On Road Vehicle / Object Detection And Tracking Using Template. *Indian Journal of Computer Science and Engineering*. 1(2), 98–107.

NHTSA (2015) Traffic Safety Facts - Published by NHTSA's National Center for Statistics and Analysis. Washington DC.

Papageorgiou, C. (2000) A Trainable System for Object Detection in Images and Video Sequences. *International Journal of Computer Vision*. 38(1685), 15–33.

Park, K.-Y., Hwang, S.-Y. (2014) Robust Range Estimation with a Monocular Camera for Vision-Based Forward Collision Warning System. *The Scientific World Journal*. 2014, 1–9.

Parodi, P., Piccioli, G. (1995) A feature-based recognition scheme for traffic scenes. *Intelligent Vehicles Symposium (IV), 1995 IEEE*, 229–234.

Petit, S. (2019) World Vehicle Population Rose 4.6% in 2016. [online]. Available from: <https://subscribers.wardsintelligence.com/analysis/world-vehicle-population-rose-46-2016> [Accessed May 7, 2019].

Plemakova, V. (2018) Vehicle Detection Based on Convolutional Neural Networks. University of Tartu.

Prabhakar, G., Kailath, B., Natarajan, S., Kumar, R. (2017) Obstacle detection and classification using deep learning for tracking in high-speed autonomous driving. *2017 IEEE Region 10 Symposium (TENSymp)*, 1–6.

Premebida, C., Monteiro, G., Nunes, U., Peixoto, P. (2007) A Lidar and vision-based approach for pedestrian and vehicle detection and tracking. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 1044–1049.

Qi, X., Wang, T., Liu, J. (2017) Comparison of Support Vector Machine and Softmax Classifiers in Computer Vision. *Proceedings - 2017 2nd International Conference on Mechanical, Control and Computer Engineering, ICMCCE 2017*, 151–155.

Rauskolb, F.W., Berger, K., Lipski, C., Magnor, M., Cornelsen, K., Effertz, J., Form, T., Graefe, F., Ohl, S., Schumacher, W., Wille, J.M., Hecker, P., Nothdurft, T., Doering, M., Homeier, K., Morgenroth, J., Wolf, L., Basarke, C., Berger, C., Gülke, T., Klose, F., Rumpe, B. (2009) Caroline: An autonomously driving vehicle for urban environments. Springer Tracts in Advanced Robotics. 56, 441–508.

Redmon, J., Divvala, S., Girshick, R., Farhadi, A. (2016) You Only Look Once: Unified, Real-Time Object Detection. In CVPR '16: Proceedings of the 2016 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'16). pp. 1–10.

Redmon, J., Farhadi, A. (2017) YOLO9000: Better, faster, stronger. Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, 6517–6525.

Ren, S., He, K., Girshick, R., Sun, J. (2015) Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence. 39(6), 1137–1149.

Rodriguez F., S.A., Fremont, V., Bonnifait, P., Cherfaoui, V. (2010) Visual confirmation of mobile objects tracked by a multi-layer lidar. 13th International IEEE Conference on Intelligent Transportation Systems, 849–854.

Rybski, P.E., Huber, D., Morris, D.D., Hoffman, R. (2010) Visual classification of coarse vehicle orientation using histogram of oriented gradients features. IEEE Intelligent Vehicles Symposium, Proceedings, 921–928.

SAE International (2016) Automated Driving -SAE International Standard J3016.

Salari, E., Ouyang, D. (2013) Camera-based Forward Collision and lane departure warning systems using SVM. Midwest Symposium on Circuits and Systems, 1278–1281.

Schamm, T., von Carlowitz, C., Zollner, J.M. (2010) On-road vehicle detection during dusk and at night. In 2010 IEEE Intelligent Vehicles Symposium. San Diego, pp. 418–423.

- Schmidhuber, J. (2015) Deep Learning in neural networks: An overview. *Neural Networks*. 61, 85–117.
- Schnieder, M. (2017) Development of an improved time-to-collision algorithm. P16CVC002\_2 European Short Research Project.
- Simonyan, K., Zisserman, A. (2014) Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*. San Diego, pp. 1–14.
- Sivaraman, S., Trivedi, M.M. (2010) A general active-learning framework for on-road vehicle recognition and tracking. *IEEE Transactions on Intelligent Transportation Systems*. 11(2), 267–276.
- Sivaraman, S., Trivedi, M.M. (2013) Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis. *IEEE Transactions on Intelligent Transportation Systems*. 14(4), 1773–1795.
- Smith, S.M. (1995) Real-Time Motion Segmentation and Shape Tracking. *Ieee Transactions on Pattern Analysis and Machine Intelligence*. 17(8), 814–820.
- Smoluk, G. (2015) Google net. In *CVPR '15: Proceedings of the 2016 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'15)*. pp. 1–9.
- Sobel, I. (1990) An isotropic 3 by 3 image gradient operator. *Machine Vision for three-dimensional Sciences*. 1(1), 23–34.
- Stackoverflow (2016) Gabor feature extraction. [online]. Available from: <https://stackoverflow.com/questions/20608458/gabor-feature-extraction> [Accessed March 6, 2019].
- Stein, G.P., Mano, O., Shashua, A. (2003) Vision-based ACC with a single camera: Bounds on range and range rate accuracy. In *IEEE Intelligent Vehicles Symposium, Proceedings*. pp. 120–125.

- Sun, D., Watada, J. (2015) Detecting pedestrians and vehicles in traffic scene based on boosted HOG features and SVM. In WISP 2015 - IEEE International Symposium on Intelligent Signal Processing, Proceedings. pp. 1–4.
- Sun, Z., Bebis, G., Miller, R. (2006) Monocular precrash vehicle detection: Features and classifiers. *IEEE Transactions on Image Processing*. 15(7), 2019–2034.
- Sun, Z., Bebis, G., Miller, R. (2002) On-road vehicle detection using Gabor filters and support vector machines. 2002 14th International Conference on Digital Signal Processing Proceedings. DSP 2002. 2, 1019–1022.
- Sun, Z., Miller, R., Bebis, G., DiMeo, D. (2002) A real-time precrash vehicle detection system. In Proceedings of IEEE Workshop on Applications of Computer Vision. pp. 171–176.
- Teoh, S.S. (2011) Development of a Robust Monocular-Based Vehicle Detection and Tracking System - PhD Thesis. University of Western Australia.
- Truong, Q.B., Lee, B.R. (2009) Vehicle Detection Algorithm Using Hypothesis Generation and Verification. In Emerging Intelligent Computing Technology and Applications - ICIC 2009. pp. 534–543.
- Uijlings, J.R.R., Van De Sande, K.E.A., Gevers, T., Smeulders, A.W.M. (2013) Selective Search for Object Recognition. *International Journal of Computer Vision*. 104(2), 154–171.
- Urmson, C., Anhalt, J., Bagnell, D., Baker, C., Bittner, R., Clark, M.N., Dolan, J., Duggins, D., Galatali, T., Geyer, C., Gittleman, M., Harbaugh, S., Hebert, M., Howard, T.M., Kolski, S., Kelly, A., Likhachev, M., McNaughton, M., Miller, N., Peterson, K., Pilnick, B., Rajkumar, R., Rybski, P., Salesky, B., Seo, Y.-W., Singh, S., Snider, J., Stentz, A., Whittaker, W. “Red”, Wolkowicki, Z., Ziglar, J. (2008) Autonomous Driving in Urban Environments: Boss and the Urban Challenge. *Journal of Field Robotics*. 25(8), 425–466.
- Vapnik, V.N. (1998) *Statistical Learning Theory*. New York: John Wiley & Sons.

Viola, P., Jones, M.J. (2001) Robust Real-time Object Detection. In Second International Workshop of Statistics and Computational Theories of Vision - Modeling, Learning, Computing and Sampling. pp. 1–25.

Wang, J., Perez, L. (2017) The Effectiveness of Data Augmentation in Image Classification using Deep Learning. Unpublished.

Wei, J., He, J., Zhou, Y., Chen, K., Tang, Z., Xiong, Z. (2019) Enhanced Object Detection With Deep Convolutional Neural Networks for Advanced Driving Assistance. IEEE Transactions on Intelligent Transportation Systems. PP, 1–12.

Wen, L., Du, D., Cai, Z., Lei, Z., Chang, M., Qi, H., Lim, J., Yang, M., Lyu, S. (2015) UA-DETRAC: A New Benchmark and Protocol for Multi-Object Detection and Tracking. eprint arXiv:1511.04136, 1–18.

Wille, J.M., Saust, F., Maurer, M. (2010) Stadtpilot: Driving autonomously on Braunschweig's inner ring road. 2010 IEEE Intelligent Vehicles Symposium, 506–511.

World Health Organisation (2018) Global Status Report on Road Safety 2018. Geneva: World Health Organisation.

Wu, C., Duan, L., Miao, J., Fang, F., Wang, X. (2009) Detection of front-view vehicle with occlusions using AdaBoost. In Proceedings - 2009 International Conference on Information Engineering and Computer Science, ICIECS 2009. pp. 2–5.

Yanpeng, C., Renfrew, A., Cook, P. (2008) Vehicle Motion Analysis Based on a Monocular Vision System. RTIC 2008 and ITS United Kingdom Members' Conference, 1–6.

Yao, Y., Tian, B., Wang, F.Y. (2017) Coupled Multivehicle Detection and Classification with Prior Objectness Measure. IEEE Transactions on Vehicular Technology. 66(3), 1975–1984.

Yong, H., Liangqun, L. (2018) A Novel Multi-source Vehicle Detection Algorithm

based on Deep Learning - IEEE Conference Publication. In 2018 14th IEEE International Conference on Signal Processing (ICSP). IEEE, pp. 979–982.

Yu, H., Yuan, Y., Guo, Y., Zhao, Y. (2016) Vision-based lane marking detection and moving vehicle detection. In Proceedings - 2016 8th International Conference on Intelligent Human-Machine Systems and Cybernetics, IHMSC 2016. pp. 574–577.

Zeiler, M.D., Fergus, R. (2013) Visualizing and understanding convolutional networks. In European Conference on Computer Vision. pp. 818–833.

Zhang, Y., Wang, J., Yang, X. (2017) Real-time vehicle detection and tracking in video based on faster R-CNN. Journal of Physics: Conference Series. 887, 2–7.

Zhao, Z.Q., Zheng, P., Xu, S.T., Wu, X. (2019) Object Detection With Deep Learning: A Review. IEEE Transactions on Neural Networks and Learning Systems. 14(8), 1–21.

Ziegler, J., Bender, P., Schreiber, M., Lategahn, H., Strauss, T., Stiller, C., Dang, T., Franke, U., Appenrodt, N., Keller, C.G., Kaus, E., Herrtwich, R.G., Rabe, C., Pfeiffer, D., Lindner, F., Stein, F., Erbs, F., Enzweiler, M., Knoppel, C., Hipp, J., Haueis, M., Trepte, M., Brenk, C., Tamke, A., Ghanaat, M., Braun, M., Joos, A., Fritz, H., Mock, H., Hein, M., Zeeb, E. (2014) Making bertha drive-an autonomous journey on a historic route. IEEE Intelligent Transportation Systems Magazine. 6(2), 8–20.

Zitnick, C.L., Dollár, P. (2014) Edge boxes: Locating object proposals from edges. In European Conference on Computer Vision - ECCV 2014. pp. 391–405.