Development of a Robust Monocular-Based Vehicle Detection and Tracking System

Soo Siang Teoh

This thesis is presented to the

School of Electrical, Electronic and Computer Engineering

for the degree of

Doctor of Philosophy of

The University of Western Australia

December 2011

Abstract

This dissertation investigates the techniques for monocular-based vehicle detection. A novel system that can robustly detect and track the movement of vehicles in the video frames is proposed. The system consists of three major modules: a symmetry based object detector for vehicle cueing, a two-class support vector machine (SVM) classifier for vehicle verification and a Kalman filter based vehicle tracker.

For the cueing stage, a technique for rapid detection of all possible vehicles in the image is proposed. The technique exploits the fact that most vehicles' front and rear views are highly symmetrical in the horizontal axis. First, it extracts the symmetric regions and the high symmetry points in the image using a multi-sized symmetry search window. The high symmetry points are then clustered and the mean locations of each cluster are used to hypothesize the locations of potential vehicles. From the research, it was found that a sparse symmetry search along several scan lines on a scaled-down image can significantly reduce the processing time without sacrificing the detection rate.

Vehicle verification is needed to eliminate the false detections picked up by the cueing stage. Several verification techniques based on template matching and image classification were investigated. The performance for different combinations of image features and classifiers were also evaluated. Based on the results, it was found that the Histogram of Oriented Gradient (HOG) feature trained on the SVM classifier gave the best performance with reasonable processing time.

The final stage of the system is vehicle tracking. A tracking function based on the Kalman filter and a reliability point system is proposed in this research. The function tracks the movement and the changes in size of the detected vehicles in consecutive

video frames.

The proposed system is formed by the integration of the above three modules. The system provides a novel solution to the monocular-based vehicle detection. Experimental results have shown that the system can effectively detect multiple vehicles on the highway and complex urban roads under varying weather conditions.

Acknowledgements

First of all I would like to thank my supervisor, Prof. Thomas Bräunl for all his support, advice and guidance throughout the course of my PhD research.

I must also thank A/Prof. Mike Alder for his advice and discussion on the topic of pattern classification, and his help to proof-read some of my written work.

Thanks to A/Prof. Du Huynh who has introduced me to the interesting world of computer vision. Thanks also go to Linda and Roozbeh for their help to proof-read the draft of this thesis.

I thank Daimler Research for providing some of the video sequences used in this thesis.

Many thanks to the University of Science Malaysia and the Malaysian Ministry of Higher Education for their financial support which made my studies at UWA possible.

To the members of the UWA Robotics lab and the Driver Assistance Group–Azman, Jonathan, Matthew, Xin and Markus, I thank them for all their support and contributions.

I also thank many lab-mates in room 3.11 over the years. The great times and good company in the lab have made my studies at UWA more enjoyable and memorable.

Last but not least, I express my special thanks to my family, for their continual love, patience and encouragement. Without them, I would not have been able to complete this thesis.

Contents

Abstract i

Acknowledgements iii

Contents iv

List of Figures viii

List of Tables xiii

Table of Acronyms xv

1 Introduction 17

1.1 Driver Assistance Systems for Collision Avoidance 17

1.1.1 Sensors for Driver Assistance Systems 18

1.2 Problem Statements and Motivations 19

1.3 Objectives 21

1.4 Thesis Organisation 22

2 Review of Vision-based Vehicle Detection Techniques 25

2.1 Introduction 25

2.2 Monocular-based Vehicle Cueing Techniques 26

2.2.1 Motion Based Vehicle Cueing 27

2.2.2 Appearance Based Vehicle Cueing 29

2.3 Vehicle Verification 35

2.3.1 Template Matching 35

2.3.2 Classifier-based Vehicle Verification 36
2.4 Summary 39

3 A Novel Symmetry Based Vehicle Cueing Technique 41
3.1 Types of Symmetry 42
3.2 2D Bilateral Symmetry Detection 44
3.2.1 Direct Approach 44
3.2.2 Phase-based Symmetry Detection 45
3.2.3 Voting-based Symmetry Detection 46
3.2.4 Global Versus Local Symmetry 46
3.2.5 Pixel Characteristics for Symmetry Calculation 47
3.3 Vehicle Cueing Using Symmetry 49
3.3.1 Previous Work 49
3.4 A Variable Window Size and Scan-line Based Symmetry Search Technique 52
3.4.1 Variable-size Symmetry Search Window 52
3.4.2 Scan-line Based Symmetry Search 53
3.4.3 The Proposed Symmetry Cueing Technique 54
3.4.4 Colour to Gray Scale Conversion and Edge Detection 55
3.4.5 Determination of Feasible Regions for Vehicle Search 55
3.4.6 Contour-based Symmetry Detection 57
3.4.7 Detection of Peaks in the Symmetry Plots 57
3.4.8 Clustering the Symmetry Points 59
3.4.9 Vehicle Bounding Box Estimation 60
3.5 Experiments and Results 63
3.5.1 Performance of the Proposed Vehicle Cueing System 64
3.5.2 Performance Comparison of Scan-line Based Symmetry Search and Full Symmetry Search 66
3.5.3 Performance Comparison for Using Variable Size and Fixed Size Symmetry Search Windows 66
3.5.4 The Effect of Image Resolution on the Performance of the Vehicle Cueing System 67

3.5.5 Performance Comparison of the Proposed Symmetry Based and the Motion Based Cueing Techniques 68
3.6 Summary 70

4 A Classifier-based Vehicle Verification Technique 73
4.1 Introduction 73
4.2 Vehicle Verification Using a Classifier 74
4.2.1 Support Vector Machine 76
4.2.2 Multilayer Perceptron Feed-forward Artificial Neural Network . 78
4.2.3 Mahalanobis Distance 80
4.3 Image Features Extraction 81
4.3.1 Histogram of Oriented Gradient Feature 82
4.3.2 Gabor Filter 84
4.4 Feature Dimension Reduction 89
4.4.1 Feature Selection Using Principal Component Analysis 90
4.4.2 Feature Selection Using F-Score 92
4.5 Performance Evaluation 92
4.5.1 Performance Matrices 93
4.5.2 Vehicles Data Set 94
4.5.3 Classifier Training 96
4.5.4 Performance Evaluation for the HOG Feature 98
4.5.5 Performance Evaluation for the Gabor Feature 105
4.5.6 Performance Evaluation for Using Reduced Feature Sets 108
4.6 Summary 113

5 Vehicle Tracking Using Kalman Filter 115
5.1 The Vehicle Tracking Module 115
5.2 Application of Kalman Filter for Vehicle Tracking 117
5.2.1 The Discrete Time Kalman Filter 118
5.2.2 Implementation of the Kalman Filter for Vehicle Tracking . . . 120
5.3 The Reliability Point System 122
5.4 Experiments and Results 124

5.4.1 Tracking of a Preceding Vehicle 124
5.4.2 Tracking of an Overtaking Vehicle 126
5.5 Summary 128

6 System Integration, Experiments and Results 131
6.1 The Proposed Vehicle Detection System 131
6.1.1 Software Modules 134
6.1.2 Hardware Components 135
6.2 Experiments and Results 136
6.2.1 Experiment Setup 136
6.2.2 Results and Discussions 137
6.3 Summary 145

7 Conclusion and Future Work 147
7.1 Conclusion 147
7.2 Future Work 149

References 153

A Format of Feature Vector Files for the Experiments in Chapter 4 169

List of Figures

1.1 Figure shows different active and passive safety systems that operate at different time intervals, before, during and after a collision. 18

1.2 Thesis organisation 23

2.1 The two-stage approach for vision-based vehicle detection 26

2.2 ASSET-2 sparse optical flow based vehicle detection 27

2.3 Search windows (A1 and A2) are set up on plane P to group the corresponding clusters for each vehicle. (Figure taken from [20]). 28

2.4 Vehicle cueing based on shadow. (a) An example road scene. (b) Histogram of road pavement. (c) Image is thresholded to extract the shadow and edges of the vehicle. (Figures taken from [21]). 29

2.5 Vehicle cueing based on edge information. (Figure taken from [37]). 30

2.6 An example of symmetry based vehicle detection: (a) An example road scene. (b) The plot of intensity based symmetry. (Figures taken from [25]). 31

2.7 Four different masks are used to detect the four corners of a rectangle. (Figures taken from [24]). 32

2.8 A double-circle mask for corner detection (a). In (b), (c) & (d), the mask is used to detect different shapes of corners. (Figures taken from [40]). 32

2.9 Vehicle cueing based on colour: (a) Original image (b) Detected tail lights from colour segmentation. (Figures taken from [41]). 33

2.10 Vehicle segmentation based on local image entropy. (a) An example road scene. (b) Image local entropy. (c) Segmentation result. (Figures taken from [42]). 34

2.11 Vehicle detection based on tail lights: (a) An example of night time road environment. (b) Bright objects extraction. (c) Result of spatial clustering of the bright components. (Figures taken from [33]). 34

3.1 Types of 2D Symmetries: (a) Reflectional Symmetry (b) Rotational Symmetry (c) Radial Symmetry (d) Circular Symmetry 43
3.2 Example of (a) True Mirror Symmetry, (b) Partial Mirror Symmetry and (c) Skewed Symmetry 44
3.3 An example of vehicle detection using the direct approach. The similarity between the original image and the symmetry transformed image (in this case, the horizontal reflectional symmetry) is checked to determined the degree of symmetry 45
3.4 Voting-based symmetry detection. Each pair of pixels in the symmetry search window casts a vote to determine the symmetry of the centre pixel . 46
3.5 Uneven illumination and reflection on the vehicle's rear windscreen affect the symmetrical appearance of the vehicle 48
3.6 The areas of road surface and sky in the image are highly symmetric since they have uniform pixels' intensity 48
3.7 A normal road scene with vehicles at different distances: Close vehicle (1) has a bigger image size and its bottom edge is closer to the bottom of the image; the image size of a distant vehicle (5) is smaller and its bottom edge is near to the horizon; vehicles (2) and (3) are at almost the same distance, they show up in the image at almost equal size and their bottom edges are on the same image row. (Road photo taken from [100]). 52
3.8 Plot of vehicle's bottom position versus vehicle's width in the image. 53
3.9 Flow chart of the proposed symmetry cueing technique 54
3.10 Resulting image after Canny edge detection 56
3.11 Symmetry scan lines 56
3.12 Symmetry plots along the scan lines 58
3.13 Peak symmetry points on each scan line (image enlarged) 58
3.14 Result of clustering the symmetry points. Four clusters were found for this image. The bigger dots are the mean points for the clusters (image enlarged) 59

3.15 Resizing the ROI for a vehicle's bounding box detection: (a) the ROI centred at the symmetry point is enlarged if no vehicle is detected. (b) An example of ROI, the vertical line is the symmetry axis of the ROI 60
3.16 (a) Edge image of the ROI. (b) Enhancing the vehicle's edges by removing the points that are non-symmetric along the symmetry line 61
3.17 Bounding box detection: (a) the horizontal (right) and the vertical (bottom) projection maps of the edge enhanced image. (b) The estimated vehicle's bounding box 62
3.18 Implementation of the symmetry-based vehicle cueing system on the ImprovCV image processing framework 63
3.19 Area of positive detection. If the centre point of the hypothesised vehicle fall within the shaded area, it is counted as a positive detection 64
3.20 Examples of false detections: (a) road divider, (b) guardrail and (c) sign board. The big white dots are the mean points of the symmetry clusters. . . 65
3.21 Image shows one of the video frame from the result of motion based vehicle cueing. The arrows are the detected optical flow vectors and the white dots are the hypothesised vehicle's locations 70

4.1 The block diagram of a classifier based vehicle verification system 75
4.2 An example of the optimal separating hyperplane and margin for a two-dimensional feature space 76
4.3 An example of the MLP network topology with n inputs, 1 output and two hidden layers 79
4.4 The response of the symmetrical sigmoid function. β is the slope parameter 80
4.5 The 8 histogram bins for quantising the gradient orientations 82
4.6 Dividing the 32x32 subimage into 16 cells and 9 overlapping blocks for the HOG feature extraction (for clarity, only 4 blocks are drawn in the picture) . 83
4.7 Summary of the steps for HOG feature extraction 84
4.8 Example of Gabor response for: (a) a positive image and (b) a negative image. Four Gabor kernels with orientations 0°, 45°, 90° and 180° were used in this example. 86
4.9 Summary of Gabor feature extraction used in this experiment 88

4.10 An example of the ROC curves 94
4.11 Example of positive training images 95
4.12 Example of negative training images 95
4.13 Image preprocessing and feature extraction 96
4.14 Dividing the orientation angles into 4, 8, 16 and 32 histogram bins (shown in (a), (b), (c) and (d) respectively) 99
4.15 ROC curves for HOG feature with different number of histogram bins . . . 100
4.16 Figure shows the grouping of orientations with opposite direction into the same histogram bin. Two examples are shown here: (a) Grouping of 8 orientations into 4 bins and (b) Grouping of 16 orientations into 8 bins 101
4.17 ROC curves for the HOG feature with and without grouping of the opposite orientation bins 102
4.18 ROC curves for the HOG feature with different methods of gradient calculation 104
4.19 ROC curves for HOG feature using different classifiers 105
4.20 ROC curves for Gabor feature with different numbers of orientations and scales 106
4.21 ROC curves for Gabor feature on different classifiers 108
4.22 Percentage of variance explained by each Principal Component 109
4.23 F-score values for the HOG features 110
4.24 Bar chart showing the cross validation detection rates for the reduced feature sets selected by PCA and F-score 111
4.25 ROC curves for the reduced feature sets selected using PCA 112
4.26 ROC curves for the reduced feature sets selected using F-score 112

5.1 Flow diagram of the vehicle tracking process 116
5.2 The two-phase Kalman filter cycle: The time update predicts ahead in time based on the current state. The measurement update adjusts the prediction with the data from the latest measurement 118
5.3 Figure shows the parameters used in the Kalman filter. They are the coordinates of the vehicle's centre point (x, y) and the area of the vehicle's image (A) 120
5.4 Preceding vehicle tracking: The measured and the Kalman's estimated values for the vehicle's x-coordinate 125

5.5 Preceding vehicle tracking: The measured and the Kalman's estimated values for the vehicle's y-coordinate 125
5.6 Preceding vehicle tracking: The measured and the Kalman's estimated values for the vehicle's image size 126
5.7 Tracking result for an overtaking vehicle: The measured and the Kalman's estimated values for the vehicle's x-coordinate 127
5.8 Tracking result for an overtaking vehicle: The measured and the Kalman's estimated values for the vehicle's y-coordinate 127
5.9 Tracking result for an overtaking vehicle: The measured and the Kalman's estimated values for the vehicle's image size 128

6.1 The flow diagram of the proposed vehicle detection system 132
6.2 Regularly skipping the cueing and verification stages to reduce the overall processing time of the vehicle detection system 133
6.3 The main software modules for the vehicle detection system 134
6.4 Figure shows the setup for testing the vehicle detection system in a BMW X5 vehicle 135
6.5 The ImprovCV image processing framework. Figures show two examples of the test results with some intermediate outputs displayed in the smaller windows 137
6.6 Examples of complex road scenes 139
6.7 Some detection results for the highway scenes 140
6.8 Some detection results for the urban road scenes 141
6.9 More detection results for the urban road scenes 142
6.10 Examples of false positives (wrong detections) indicated by the arrows. . . 143
6.11 Examples of errors in the bounding box detection 144
6.12 Examples of false negatives (missed detections) indicated by the arrows. . . 145

List of Tables

1.1 Comparison of using active and passive sensors for vehicle detection 20

3.1 Breakdown of the processing time for each component in the vehicle cueing system 65
3.2 The performance of the proposed symmetry-based vehicle cueing system . 65
3.3 Performance comparison for using scan-line based symmetry search and whole image symmetry search 66
3.4 Performance comparison for using variable size and fixed size symmetry search windows 67
3.5 The effect of input video resolution on the performance of the vehicle cueing system 68
3.6 Performance comparison of the symmetry-based and the optical flow-based cueing systems 69

4.1 Performance of HOG feature with different number of histogram bins . . . 99
4.2 Performance of HOG feature with and without grouping of the opposite orientation bins 101
4.3 Performance of HOG feature with different methods of gradient calculation 103
4.4 Performance of HOG feature on different classifiers 105
4.5 Performance of Gabor feature with different numbers of orientations and scales 106
4.6 Performance of Gabor feature on different classifiers 107
4.7 Cross validation detection rates for different features subsets 111

6.1 Processing time of the vehicle detection system 138
6.2 The test results for the vehicle detection system 138

Table of Acronyms

Abbreviation	Description
2D	2-dimensional
3D	3-dimensional
ADAS	Advanced Driver Assistance System
ANN	Artificial Neural Network
CPU	Central Processing Unit
FA	Factor Analysis
FNR	False Negative Rate
FP	False Positive
FPR	False Positive Rate
GLCM	Gray Level Co-occurrence Matrix
HOG	Histogram of Oriented Gradient
HSL	Hue Saturation Lightness
HSV	Hue Saturation Value
ICA	Independent Component Analysis
IMP	Inverse Perspective Mapping
LADAR	Laser Detection and Ranging
LIDAR	Light Detection and Ranging
NCC	Normalised Correlation Coefficient
MLP	Multi Layer Perceptron
PC	Principal Component
PCA	Principal Component Analysis

Abbreviation	Description
RBF	Radial Basis Function
ROC	Receiver Operating Curve
ROI	Region Of Interest
SAD	Sum of Absolute Difference
SIFT	Scale Invariant Feature Transform
SSD	Sum of Squared Difference
SURF	Speeded up Robust Features
SVM	Support Vector Machine
TP	True Positive
TPR	True Positive Rate

Chapter 1

Introduction

This introductory chapter starts with an overview of the vehicle safety systems for collision avoidance, then the motivations and objectives of this research is given. Finally, an overview of the structure of this thesis is explained.

1.1 Driver Assistance Systems for Collision Avoidance

Approximately 1.3 million people die each year on the world's roads, and between 20 and 50 million sustain non-fatal injuries. If current trends continue, road crashes are predicted to increase by 65% and become the fifth leading cause of death by 2030 [1]. In economic terms, the direct costs due to road traffic injuries have been estimated at US$ 518 billion and cost governments between 1–3% of their gross national product (GNP) [2]. The high fatality rate and economic costs have prompted the United Nation to launch a global program—"Decade of Action for Road Safety 2011-2020" in May 2011. The goal of the program is to stabilise and reduce the forecast level of road traffic accidents [3]. One of the initiatives in the program is to promote more widespread use of crash avoidance technologies for vehicles.

One of the main technologies for crash avoidance are driver assistance systems. The purpose of the driver assistance systems is to perceive the surrounding using different sensors, identify critical situations and provide the information to the driver. In extreme

cases, the system may even take automatic evasive action to prevent a collision. Examples of driver assistance systems for crash avoidance are such as the forward collision warning, brake assistance and lane departure warning systems. These are part of the active safety systems since they take proactive steps to prevent an accident before it happens. On the other hand, passive safety systems such as seatbelts, airbags and crumple zones reduce the severity of injuries during a collision. Figure 1.1 shows the timeline of a collision and different active and passive safety systems that operate at different time intervals, before, during and after a collision.

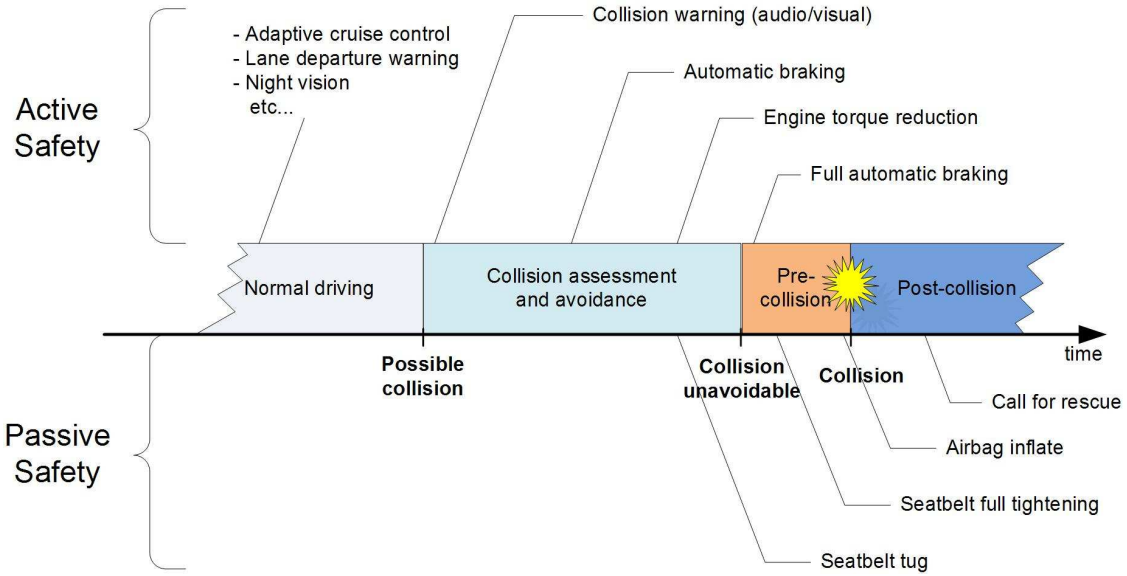


Figure 1.1: Figure shows different active and passive safety systems that operate at different time intervals, before, during and after a collision.

Passive safety systems have been widely employed for many years and it has almost reached its full potential for reducing the number of casualties [4]. The focus in automotive safety systems is now on the driver assistance systems for preventing the accident itself.

1.1.1 Sensors for Driver Assistance Systems

Different sensors can be used to collect the information about the road conditions for a driver assistance system. These sensors can be categorised into two main groups: active and passive sensors. Active sensors such as radar transmit radio waves into the atmosphere. The signals are reflected by other objects and captured by a detector. The

distance of the object can be calculated by measuring the time travelled by the signals. Other active sensors use the same concept but operate at different regions of the electromagnetic spectrum, for example LIDAR (Light Detection and Ranging) uses infrared signals and LADAR (Laser Detection and Ranging) uses laser waves. The main advantage of the active sensors is their capability to measure distance directly without requiring high computational resources. They are also more robust to environment variation such as illumination changes caused by shadow, fog, rain or different times of day.

Optical sensors such as normal cameras are the most common passive sensors used in a vision-based driver assistance system. Such sensors have attracted a lot of attention in the past decade because of the availability of low cost and high resolution cameras as well as the increasing processing speed of computers. Another key advantage of using an optical sensor is its ability to give a richer description about the vehicle's surroundings compared to the active sensor. Some applications such as lane detection and object identification have to rely on the captured visual images to extract the required information. The passive sensors are also free from interference problems commonly faced by the active sensors.

However, detection based on the optical sensor is highly dependent on the quality of the captured images, which can be easily affected by the lighting and environment conditions. Therefore, vision based vehicle detection systems require more complicated image processing algorithms and higher computational resources to extract the required information from the captured images. Table 1.1 lists the pros and cons of using the active and passive sensors for driver assistance application.

1.2 Problem Statements and Motivations

One of the main research areas for driver assistance systems is the development of algorithms for vehicle detection. The algorithms analyse the surrounding information captured by the onboard sensors. Although the technology of millimetre-wave radar for vehicle detection has been well established, it is yet to achieve mass acceptance due to its higher cost. On the other hand, vision sensors offer a significant cost saving to the

Table 1.1: Comparison of using active and passive sensors for vehicle detection

Type of sensor	Advantages	Disadvantages
Active sensor (Radar, Lidar, Laser)	1. Able to measure distance directly with less computing resources 2. Longer detection range compared to optical sensor 3. More robust compared to optical sensor in foggy or rainy day, during night time or on roads with complex shadows.	1. Same type of sensors will interfere with each other when operating close to each other 2. Higher cost 3. Lower spatial resolution
Passive sensor (camera)	1. Lower cost, easier to install and maintain 2. Higher resolution and wider view angle 3. Extensive information can be extracted from the visual images.	1. The quality of the captured images depends on the lighting and weather conditions. 2. Requires more computing resources to process the captured images

system, and yet they can capture more visual details of a vehicle. However, detecting vehicles from video images is a challenging task. One major drawback in particular the monocular-based system is its inability to measure depth information directly. It has to rely on the visual cues to detect the vehicles. Below are the main challenges of vision-based vehicles detection for driver assistance applications:

1. **The image background may be cluttered with other stationary objects**. This is especially obvious for the urban road scenes. The background objects such as buildings, sign boards, shadows and vegetation may obscure the vehicles in the image.

2. **Images are captured from a moving platform**. The captured images may contain vehicles and other objects at different distances, in a background that constantly changing, when the test vehicle is moving. This makes the background subtraction technique commonly used to detect objects in a video surveillance system unsuitable for this usage.

3. **Variable outdoor illumination conditions**. The illumination of the road environment may change during different times of day and under different weather conditions. This may affect the quality of the captured images.

4. **The need for real-time performance**. For driver assistance applications, the system must run in real-time in order to detect any potential dangerous situation and provide timely warning for the driver to take evasive action. Therefore, the detection algorithms have to be fast.

5. **The need for a robust system**. The detection system has to be robust, since a critical miss may lead to an accident. On the other hand, too many false alarms may divert the driver's attention.

Due to the above challenges, vision-based vehicle detection requires the application of sophisticated computer vision techniques to segment the vehicles in the images. Although many techniques have been proposed in the literature since the 1990s (a review is given in Chapter 2), it still remains as an active research area. The current availability of higher resolution cameras and faster processors has opened up the new possibility of using more computationally extensive techniques such as machine learning for real-time vehicle detection.

The motivation of this research is to extend the current knowledge of applying computer vision and machine learning techniques for monocular-based vehicle detection. The findings will contribute to the development of a more affordable camera based crash avoidance system that can be widely employed in different ranges of vehicles.

1.3 Objectives

The main objective of this research is to develop a robust monocular-based vehicle detection system. The system uses a single camera installed behind the windscreen to capture the road images in front of the test vehicle. Vehicles are detected by analysing the captured images using computer vision and machine learning techniques. This research will

focus on the detection of preceding vehicles during daytime in various weather conditions.

The main outcome of the research is a robust technique for monocular-based vehicle detection. The technique combines different algorithms developed in three stages— vehicle cueing, vehicle verification and vehicle tracking. It can be used as the core component for other higher level driver assistance applications such as collision warning and brake assistance systems.

1.4 Thesis Organisation

Figure 1.2 shows the thesis organisation and the dependencies among the chapters. Following this introductory chapter, a review of the literature on the existing vision-based vehicle detection techniques is given in **Chapter 2**. The review will focus on the different stages of a vision-based vehicle detection system. Each of these stages is directly related to the work presented in Chapter 3, 4 and 5.

Chapter 3 presents the symmetry-based vehicle cueing technique proposed in this research. **Chapter 4** begins with the evaluation of different vehicle verification techniques. It continues with the discussion of the proposed classifier based vehicle verifier. **Chapter 5** explains the vehicle tracking function and the integration of Kalman filter and a reliability point system to smooth the tracking.

Chapter 3 to 5 form the three main modules of the proposed vehicle detection system. The integration of these modules and the experiments to evaluate the complete system are presented in **Chapter 6**. Finally in **Chapter 7**, the conclusion and the recommendation for future work are given.

Part of the work presented in this thesis has been published in the following journal paper [5]:
"Soo Teoh and Thomas Bräunl. Symmetry-based monocular vehicle detection system, *Machine Vision and Applications*, 2011, DOI:10.1007/s00138-011-0355-7"

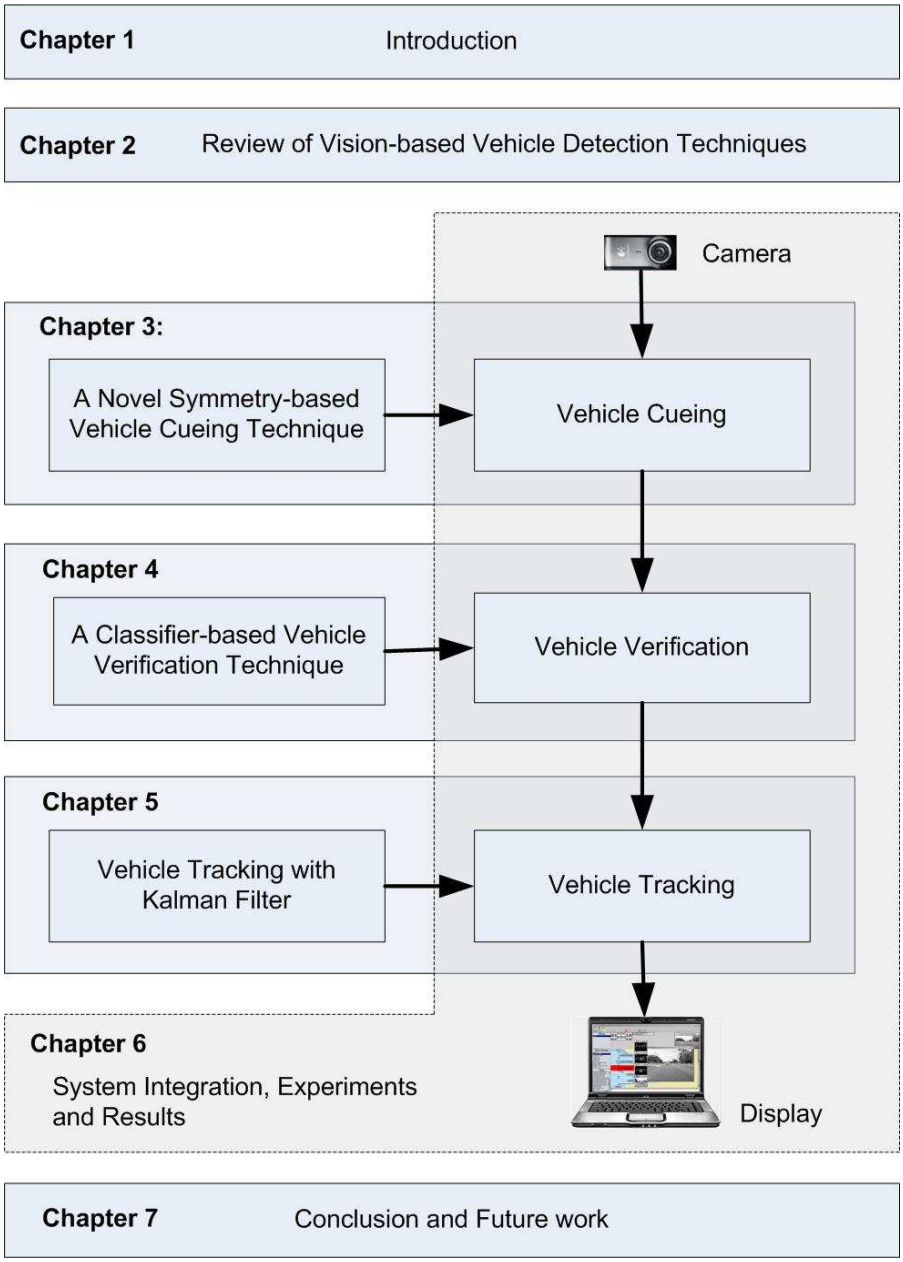


Figure 1.2: Thesis organisation

Chapter 2

Review of Vision-based Vehicle Detection Techniques

This chapter reviews some of the current literature related to vision-based vehicle detection. The techniques are divided into different categories based on their detection approach and processing stage. Some representative systems will also be described.

2.1 Introduction

The most common approach of vision-based vehicle detection consists of the following two stages [6]: vehicle cueing and vehicle verification (Figure 2.1). The purpose of the cueing stage is to search through the whole image to find all possible vehicle candidates. Then the verification stage validates the identified candidates as either vehicle or non-vehicle. Once a vehicle is verified, it be passed to a tracking function which monitor the movement of the vehicle in consecutive video frames. When moving from one stage to the following stage, the amount of information to be processed is reduced. This will allow more sophisticated and time consuming algorithms to be carried out on a smaller region of the image.

Although it is possible to skip the cueing stage and just using a verifier to scan for vehicles in the whole image, this approach is not commonly used since it requires a high

computational load. Most verification algorithms are computationally intensive and applying them on every region of the image will be very time consuming.

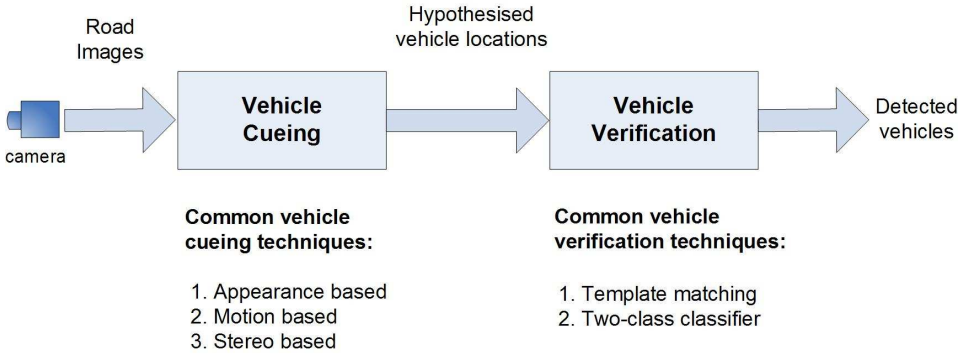


Figure 2.1: The two-stage approach for vision-based vehicle detection

The following sections review some current techniques for vehicle cueing and verification. The review will focus on the monocular based vehicle detection techniques. Some representative systems for each technique will also be described.

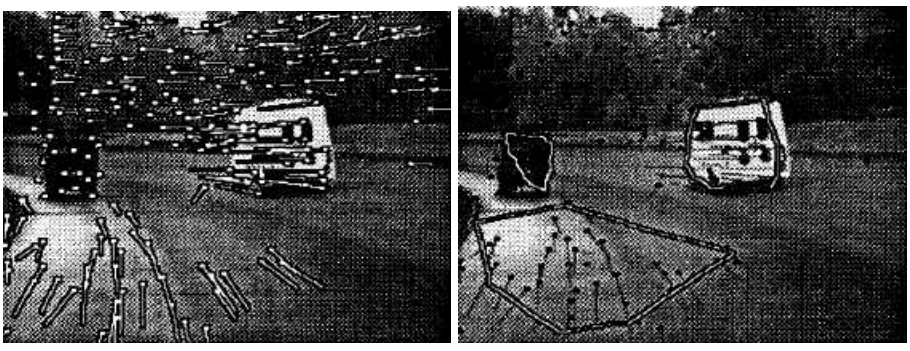
2.2 Monocular-based Vehicle Cueing Techniques

The purpose of vehicle cueing is to identify the possible vehicle locations in the captured image and mark them as regions-of-interests (ROIs). For a system with radar and vision fusion, the determination of ROIs is done by analysing the distance and the relative speed information collected by the radar. Different vision and radar fusion techniques were proposed in [7–9]. For systems with stereo cameras such as in [10–12], the disparity map or the inverse perspective mapping computed from the stereo images was used to find the possible vehicle locations. For a monocular-based system, the determination of the vehicle locations has to be done by analysing the vehicle's motion or appearance. The motion based technique requires the analysis of several image frames to detect moving objects based on their optical flows. On the other hand, the appearance based vehicle detection technique analyses a single image to find visual cues to segment the vehicles. The following subsections explain these two techniques and review some of the representative literature.

2.2.1 Motion Based Vehicle Cueing

Motion based approach exploits the temporal information to detect vehicles. The optical flow fields from a moving vehicle can be calculated by matching pixels or feature points between two image frames. Dense optical flow such as the method suggested by Horn and Schunck [13] matches every pixel in the image based on their intensity. This technique requires huge computational effort and therefore is not so suitable for real-time application. On the other hand, sparse optical flow tracks a set of specific features from a vehicle such as corners [14] or color blobs [15]. After the optical flow fields are calculated, moving objects can be segmented from the image by clustering the fields based on their positions, magnitudes and angles.

A motion based vehicle detection system called *Scene Segmenter Establishing Tracking* (ASSET-2) was proposed by Smith in [14]. The system uses features based (sparse) optical flow for motion estimation. First, the corner's features in the image was extracted using either the *Smallest Univalue Segment Assimilating Nucleus* (SUSAN) [16] or *Harris* [17] corner detectors. Then the features are tracked over several frames to create the optical flow fields. A 'flow segmenter' is used to cluster the fields based on their flow's variations. Finally the bounding box and centroid of the resulting clusters are calculated and used as the hypothesised vehicles. The system requires high computational load and therefore they used special hardware acceleration to attain real-time performance.



(a) (b)

Figure 2.2: ASSET-2 sparse optical flow based vehicle detection. (a) The flow vectors. (b) Result of clustering the flow vectors and their bounding boxes. (Figures taken from [14]).

A similar vehicle detection system proposed by Yanpeng *et al*. [18] uses the SUSAN features to estimate the sparse optical flow fields. They proposed a technique to improve

the flows calculation using a 3-D Pulse Coupled Neural Network (PCNN) [19]. In their experiments, they showed that the accuracy of the motion based detection depends on the relative speed between the host and the preceding vehicles. Vehicles with small relative speed (< 10 km/h) achieved low detection rate (69.1%). For this reason, they also used appearance based technique (shadow underneath the vehicle and edges) in the detection.

A technique for grouping the clusters of optical flow into individual moving objects was proposed by Willersinn *et al*. [20] (Figure 2.3). First, the detected clusters are projected into a plane (P) which is parallel to the road surface. Starting at point W, which has been identified as an outer boundary point of a vehicle, a search area is set up based on the expected minimum and maximum vehicle's width. If the search is successful, a coordinate is calculated using the coordinates of all compatible flow clusters found in that area. This coordinate is used to estimate the width and the centre point of the vehicle.

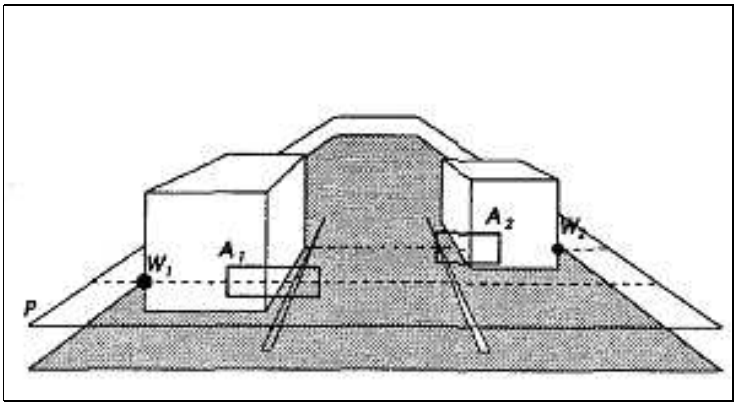


Figure 2.3: Search windows (A1 and A2) are set up on plane P to group the corresponding clusters for each vehicle. (Figure taken from [20]).

A motion based method is effective for detecting moving objects, however it is computationally intensive and requires analysis of several frames before an object can be detected. It is also sensitive to camera movement and may fail to detect objects with slow relative motion. This is a major disadvantage for driver assistance applications since the onboard camera may vibrate when the vehicle is moving and the relative motion between the test vehicle and a distant vehicle is usually small.

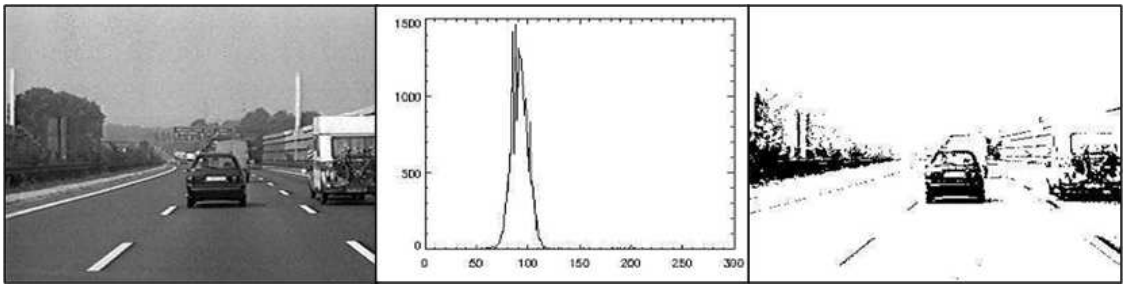
2.2.2 Appearance Based Vehicle Cueing

The appearance-based cueing technique detects vehicles based on some specific appearances of a vehicle's rear view. Examples of the appearances are the shadow underneath the vehicle [21, 22], vertical and horizontal edges [23], corners [24], symmetry [25–30], texture [31], colour [32] and the vehicle's lights [33, 34]. They are reviewed in the following subsections.

Shadow Underneath The Vehicle

The shadow underneath the vehicle which is usually darker than the surrounding road surface can provide a cue for vehicle location. In [21], Christos *et al*. evaluated the histogram of the paved road to find a threshold for segmenting the shaded areas on the road. The locations of shaded areas together with edge information are used to hypothesise the location of vehicles (see Figure 2.4).

Detection based on the shadow is simple and fast. However, it may fail when the colour of the road pavement is uneven or when the road surface is cluttered with shadows from nearby buildings, overhead bridges or trees. In the morning or evening, a long shadow is cast at one side of the vehicle. This will produce a hypothesised location which is not at the centre of a vehicle.



(a) (b) (c)

Figure 2.4: Vehicle cueing based on shadow. (a) An example road scene. (b) Histogram of road pavement. (c) Image is thresholded to extract the shadow and edges of the vehicle. (Figures taken from [21]).

Horizontal and Vertical Edges

Most vehicles' rear view show strong vertical and horizontal edges. These characteristics can be used to hypothesise the presence of a vehicle. A group of horizontal and vertical edges that form a rectangular shape with an aspect ratio between 0.4 and 1.6 are good candidates for potential vehicles. Different techniques of edge detection can be used. For example Canny [35], Sobel or morphological [36] edge detections.

Jin *et al*. [37] used the shadow underneath the vehicle as an initial cue for a possible vehicle. Then they located the position of the vehicle based on the projection maps of its horizontal and vertical edges (Figure 2.5). Zehang *et al*. [38] proposed a multiscale approach to detect a vehicle's edges using three different image resolutions. Betke [23] suggested a coarse to fine search technique to detect distant vehicles. The coarse search looks for groups of prominent edges in the image. When such groups are found, a finer search is performed in its surrounding region to locate rectangular shaped objects.

One major difficulty for detecting vehicles based on the horizontal and vertical edges is due to the interference from outlier edges generated by the background objects such as buildings, lamp posts or road dividers. It is also difficult to select an optimum threshold for the edge detection in order to capture most of the vehicle's edges with minimum edges from the background.

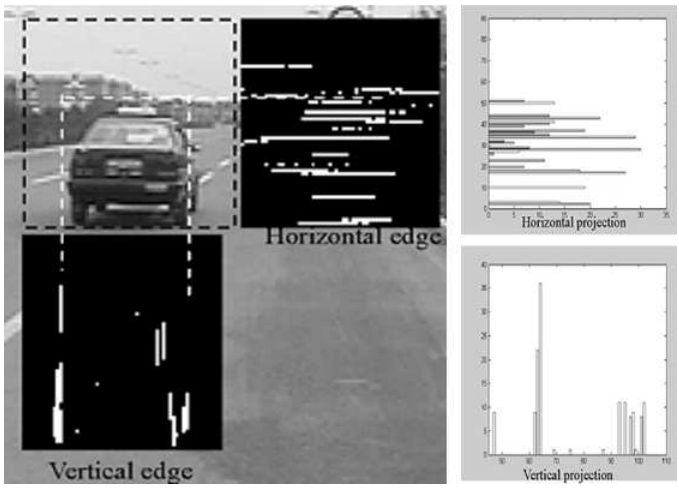
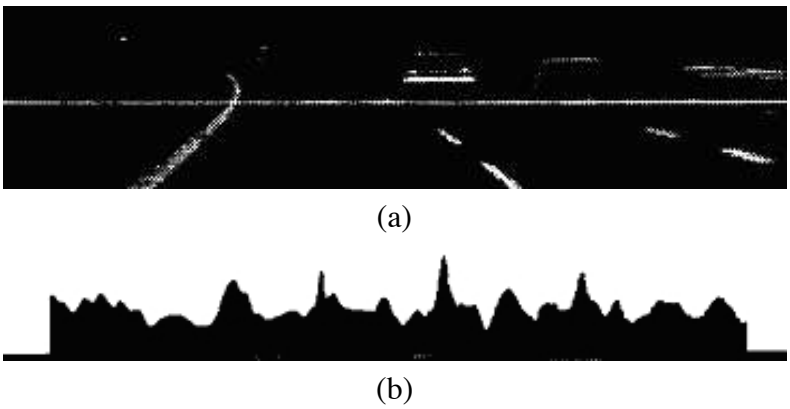


Figure 2.5: Vehicle cueing based on edge information. (Figure taken from [37]).

Symmetry

Symmetry is one of the prominent visual characteristics of a vehicle. Most vehicles' rear or front views are symmetrical over a vertical centreline. Therefore, it is possible to hypothesise the locations of vehicles in the image by detecting the regions with high horizontal symmetry. Some of the literature that uses symmetry for vehicle detection are from Bensrhair *et al*., Bin *et al*., Wei *et al*., Zielke *et al*., Kuehnle *et al*. and Du *et al*. [26–28, 25, 29, 30].

In general, the proposed symmetry detection technique uses a symmetry operator to calculate the symmetry value of an image region. Different pixel characteristics can be used in the calculation. They include gray scale value, binary contour, horizontal edges, colour and feature points.



(a)

(b)

Figure 2.6: An example of symmetry based vehicle detection: (a) An example road scene. (b) The plot of intensity based symmetry. (Figures taken from [25]).

Zielke *et al*. [25] proposed a method to detect the centreline of the leading vehicle based on the image intensity symmetry. The vehicle's bounding box is estimated by performing edge detection and finding pairs of edges that are mutually symmetric with respect to a detected symmetry axis. Kuehnle *et al*. [29] proposed a system that uses three different symmetry criteria for locating vehicles: contour symmetry, gray level symmetry and horizontal line symmetry. The histograms generated from these criteria are used to find the vehicle's centreline.

Symmetry is a useful cue for detecting vehicles. However it requires comparatively

higher computational load. Some literature such as [37, 39] proposed to use this approach for vehicle verification where only a small region of the image needs to be processed by the symmetry operator. A more comprehensive review on the symmetry based vehicle detection is given in Chapter 3.

Corners

The general shape of a vehicle is rectangular with four corners. This characteristic can be exploited to hypothesise the presence of a vehicle. In [24], Bertozzi *et al*. used four different image templates (Figure 2.7) to detect all the possible vehicle's corners in the image. A possible vehicle is detected if there are four matching corners with enough edge pixels at the positions corresponding to the vehicle's sides. In [40], the corner detection process is sped up by using a common double-circle mask to detect all types of corners (Figure 2.8). The detected corners are then clustered based on their types and locations. Finally, the features of the corners in each cluster are extracted and used as the input to an SVM classifier to determine whether it belongs to a vehicle.

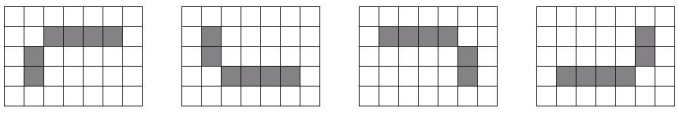


Figure 2.7: Four different masks are used to detect the four corners of a rectangle. (Figures taken from [24]).

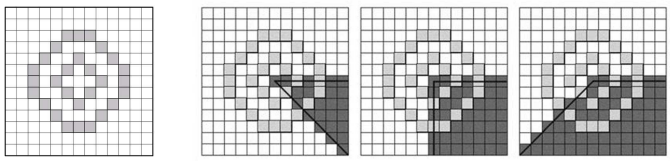


Figure 2.8: A double-circle mask for corner detection (a). In (b), (c) & (d), the mask is used to detect different shapes of corners. (Figures taken from [40]).

Colour

Colour can be a useful cue for segmenting the vehicle from the background. Most vehicles have a homogenous colour different from the road surface and the background

objects. This fact is exploited in [32] to segment the vehicles from the images. The authors proposed a color transform model to find the important 'vehicle colour' for locating possible vehicle candidates.

The pair of red brake lights and yellow signalling lights can also be a cue for detecting the vehicles. In [41], Ming *et al*. used a colour segmentation technique for detecting the vehicle's tail lights. Vehicles are hypothesised from pairs of horizontal light blobs (Figure 2.9).

(a) (b)

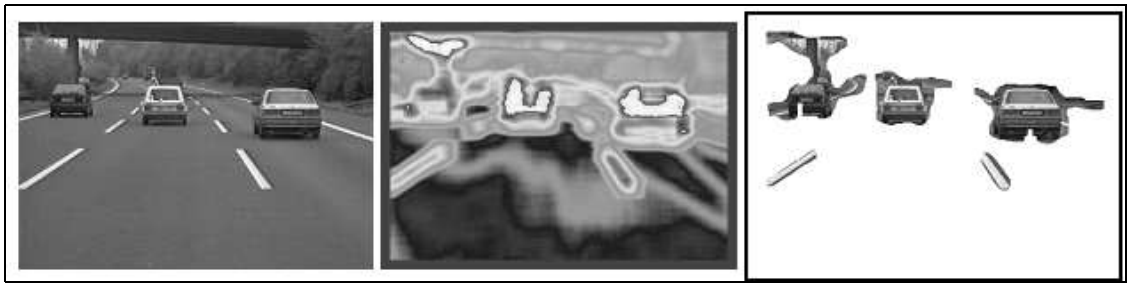
Figure 2.9: Vehicle cueing based on colour: (a) Original image (b) Detected tail lights from colour segmentation. (Figures taken from [41]).

However, colour based object detection is very sensitive to illumination change and the reflectance properties of the object. For an outdoor environment, these properties may change under different weather conditions or during different times of the day. This will increase the difficulty in vehicle detection based on colour.

Texture

The texture of a vehicle is different from its surrounding road surface or vegetation. By using statistical analysis on the image's texture, for example entropy [42] or co-occurrence matrix [43], the locations of vehicles in the image can be segmented (Figure 2.10).

However, this technique could generate a lot of false detections especially in an urban environment. This is because the texture for some of the man made structures such as buildings, sign boards or overhead bridges may resemble a vehicle.

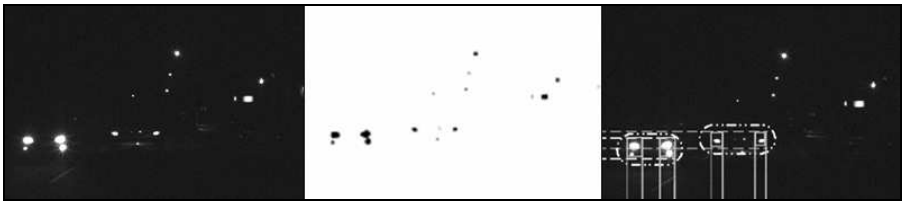


(a) (b) (c)

Figure 2.10: Vehicle segmentation based on local image entropy. (a) An example road scene. (b) Image local entropy. (c) Segmentation result. (Figures taken from [42]).

Tail Lights

During night time, the tail lights are the main cue for detecting vehicles when other features are vague. Yen *et al*. [33] proposed a system for detecting vehicles at night time. Vehicles are located by detecting the bright objects that belong to the headlights or taillights of the vehicles. Bright objects are extracted using spatial clustering and segmentation. A heuristic rule-based technique is used to analyse the light pattern and the results are used to hypothesise the location of vehicles.



(a) (b) (c)

Figure 2.11: Vehicle detection based on tail lights: (a) An example of night time road environment. (b) Bright objects extraction. (c) Result of spatial clustering of the bright components. (Figures taken from [33]).

Combining Multiple Cues

Some literature uses multiple cues to detect vehicle [44, 23, 22, 45]. For instance in [22], Leeuwen *et al*. employed a method that merges shadow, entropy and symmetry features for vehicle cueing. Their procedure starts with the identification of all image regions that potentially belong to the shadow underneath the vehicles. Then all the rows with low entropy in the detected region are removed and its horizontal intensity symmetry is checked to determine whether it belongs to a vehicle.

Vehicle detection using multiple cues is more robust since one cue may compensate the weakness of the other. However, it requires higher computational resources for calculating the additional cue which might be redundant.

2.3 Vehicle Verification

The output of the cueing stage is a set of ROIs in the image that possibly contain vehicles. They will be validated in the verification stage where the false detections are removed. Usually the detected ROIs are cropped from the image, scaled to a uniform size and have their contrast normalised before the verification process. The verification techniques can be categorised into two groups: (1) correlation based approaches using template matching and (2) learning based approaches using an object classifier.

2.3.1 Template Matching

The correlation based method uses a predefined template and calculates the correlation between the ROI and the template. The resulting correlation value is used as a similarity measure for vehicle verification. Since there are many possible vehicle's modals with different appearances on the road, a 'loose' and general template that includes the common features of a vehicle is usually used. These features include the rear window and number plate [46], a rectangular box with specific aspect ratio [47] and the "$\cup$" shape pattern from two vertical edges and a bottom horizontal line [48]. A vehicle could appear in different sizes depending on its distance from the camera. Therefore, the correlation test is usually performed at several scales in the ROI. The intensity of the image is also normalised before the correlation test in order to get a consistent result.

The biggest drawback for the fixed template method is the difficulty in getting a proper vehicle template that can fit all variants of vehicles. Therefore, instead of using a fixed generic template, some literature proposed using dynamic templates. In this approach, once a vehicle is detected using the generic template, the template for the subsequent detection is created online by cropping the image of the detected vehicle [49].

The advantage of dynamic tracking is that it is able to accurately track different modals of

vehicles once the vehicle is correctly detected. However, if there is a false detection and a dynamic template is generated based on the wrong result, all subsequent tracking will also be wrong. Mingxiu *et al*. [50] proposed a template update mechanism to address this problem. The reliability of template matching is measured based on the edges, area and aspect ratio of the target. The online update of the template is only done when this reliability measure is above a certain threshold.

Hu *et al*. [51] have also used a dynamic template method for vehicle verification. However, the quality of matching and the template update is monitored by estimating their 'vitality' values. The 'vitality' of a tracked vehicle increases when there is a sequence of successful template matching, while it decreases after a sequence of bad matches. When the 'vitality' value falls to zero, the vehicle is assumed to be no longer valid and it is removed from the tracking list.

2.3.2 Classifier-based Vehicle Verification

This approach uses a two-class image classifier to distinguish between vehicle and non-vehicle. The classifier learns the characteristics of the vehicle's appearance from the training images. The training is normally based on a supervised learning approach where a large set of labelled positive (vehicle) and negative (non-vehicle) images are used. The most common classification schemes for vehicle verification include Artificial Neural Networks (ANN) [52], Support Vector Machines (SVM) [53], Mahalanobis distance [54] and AdaBoost [55].

To facilitate the classification, the training images are first preprocessed to extract some representative features. The selection of features is very important in order to achieve good classification results. A good set of features should be able to capture most of the variability of the vehicle's appearances [56]. Different features for vehicle classification have been proposed in the literature; The most common being Histogram of Oriented Gradient (HOG) [57, 58], Gabor [59, 60], Principal Component Analysis (PCA) [61, 62] and Haar Wavelets [63, 64].

HOG feature captures the local histogram of an image's gradient orientations in a dense grid. It was first proposed by Dalal [57] for human classification. Mao *et al*. [60]

used the HOG feature trained on a linear SVM classifier to detect preceding vehicles. They showed that the system is able to detect vehicles in different traffic scenarios but no quantitative result was given. Paul *et al*. [58] proposed a system for classifying the orientation of a vehicle, where a set of orientation specific HOG features is created and trained on the SVM classifiers. Their test results show that the orientation specific classifiers can achieve 88% classification accuracy.

Papageorgiou *et al*. [63] used Haar wavelets transform to extract the vehicle's features and trained them using the SVM classifier. Three oriented wavelets responses: horizontal, vertical and diagonal are computed at different scales to allow a coarse to fine representation of the wavelets responses. They achieved a 90% detection rate when experimented on their vehicle data sets. However, a high number of false detections (10 per image) is also reported.

Viola *et al*. [64] used a similar set of Haar wavelet features but they employed the AdaBoost training algorithm and constructed a cascade of increasingly more complex classifiers. The speed up of the Haar feature extraction is achieved by using an integral image technique. They tested the system for face detection and reported a 76% to 94% detection rate. However, the system also has high false detection (average 1.3 per image).

In [65], Lienhart *et al*. addressed this problem by introducing a richer set of Haar-like features and reported an average reduction in false alarm by 10%. Chunpeng *et al*. [66] applied the same framework as [65] to detect cars and buses from video images. However, the results are not so encouraging since only 71% detection rate (at 3% false detection) is achieved.

Gabor features capture the local lines and edges information at different orientations and scales. They have been commonly used for texture analysis of images [67–70]. Zehang *et al*. [71] tested the Gabor features trained on the SVM classifier for vehicle detection. The evaluation results show that the classifier can achieve 94.5% detection rate at 1.5% false detection. They also show that the classifier outperforms a PCA feature based ANN classifier. Another paper by the same authors investigated a technique to select the Gabor parameters (orientation and scale) based on the Genetic Algorithm [59]. They found that the most important orientations of a filter are consistent with the orientation

of the vehicle edges, which are at 0°, 45°, 90° and 135°. The best scales are also tuned to encode the implicit information present in vehicle images.

Hong *et al*. [72] used boosted Gabor features and an SVM classifier for vehicle detection. Their technique selects the Gabor filter's parameters (orientation and scale) through learning from examples. Using this technique, they reported a 96% detection rate.

In [73], Yan *et al*. combined Gabor and Lagendre moment features for vehicle detection. They evaluated the performance of the features on the SVM classifiers and reported a detection rate of 99% at 1.9% false detection. They also showed that these combinations of features outperform the Haar wavelets features.

Principal Components Analysis (PCA) can be used to reduce the dimension of the image data by projecting the data into a new sub-space (eigenspace) and extract only the representative features. Truong *et al*. [61] used the PCA to build a feature vector for vehicle, naming it 'Eigenspace of vehicle'. An SVM classifier is used for the classification. The authors reported a 95% detection rate using their test data.

Alonso *et al*. [74] used the Mahalanobis distance classifier to verify vehicles. Three different measures are used in the classification: a symmetry measure, a shadow model likelihood measure and a rectangular likelihood measure. These measures are concatenated to form the feature vector. The Mahalanobis distance between a candidate's feature vector and the vehicle's class centroid is used to decide whether it belongs to a vehicle. The authors reported a 92.6% detection rate at 3.7% false detection.

Handmann *et al*. [75] proposed a texture based vehicle classification technique. This technique calculates texture features using the Cooccurrence Matrix [76] and uses a Multilayer Perceptron (MLP) Artificial Neural Network (ANN) as the classification scheme. A candidate image is classified as either car, truck or background. However, no quantitative result was given.

In [77], Milos used a similar technique as HOG for feature extraction. But instead of calculating the histogram of gradient, the histogram of the redness measure for the tail lights is calculated. The AdaBoost learning algorithm is used to construct a cascade of weak classifiers. The author used the system to recognise the rear view of Honda Accord

cars and reported a 98.7% detection rate at 0.4 false detection.

Vehicle verification based on a classifier has become more popular in recent years. This is because they are generally more accurate compared to the template matching techniques. Although there are many different types of features and classification schemes proposed in the literature, it is very hard to make a fair comparison from their published results since they have been tested using different data sets and performance measures. There is also a lack of representative data sets and common benchmark to access the performance of different vehicle classification systems.

2.4 Summary

Most of the vehicle detection systems reported in the literature consist of the following two parts: (1) Vehicle cueing to hypothesise all the possible vehicles in the image; and (2) Vehicle verification to validate the hypothesised vehicle. The algorithms for vehicle cueing can be less accurate, but they have to be fast in order to process a large area of the image to find potential vehicles. The verification stage uses more accurate but computationally expensive algorithms. However, they only operate on the regions identified by the cueing stage.

Most of the vehicle cueing techniques utilise one or more combination of features presented in section 2.2.2 to hypothesise vehicles. Among them, symmetry is one of the most widely used features. This is apparent since most vehicles possess symmetrical characteristics. By detecting the region in the image with high horizontal symmetry, all potential vehicle's candidates can be directly identified. However, the main disadvantage for using symmetry in the cueing stage is due to its comparatively higher computational load. Performing symmetry calculation on every region in the image will be very time consuming. To overcome this problem, this research proposes a rapid and efficient scan-line based, multi-sized symmetry search technique. It will be discussed in detail in Chapter 3.

For vehicle verification, most of the recent literature has proposed using the classifier based approach due to its better accuracy. Although different types of image features

and classification schemes were proposed in the literature, there is no comparative evaluation of their performances using the same data set. In this study, two efficient image features (HOG and Gabor) and three popular classifiers (SVM, ANN and Mahalanobis distance) are studied and their performances for vehicle classification are systematically evaluated under the same experimental setups. From the results, the best classifier that meets the performance and real-time requirement for the vehicle detection application is proposed. The details of the classifier development and the evaluation results are presented in Chapter 4.

Tracking takes advantage of the temporal coherence of the consecutive video frames to narrow down the areas for re-detecting a vehicle. This will improve the detection rate as well as reducing the required time for detecting the vehicles in the subsequent image frames. In this study, a tracking function that integrates the Kalman filter and a reliability point system is proposed. This will be explained in Chapter 5.

Chapter 3

A Novel Symmetry Based Vehicle Cueing Technique

The purpose of the vehicle cueing stage is to detect all possible vehicle candidates in the image. Since this is the first stage in the vehicle detection system, it has to scan a large area of the image in order not to miss any potential vehicles. In general, the algorithm for the cueing stage must have a fast computation speed and a low rate of missed detections. When tuning the system to have a low miss rate, it will inevitably allow some false detections to be generated. However, this is tolerable since the hypothesised vehicles will be further verified in the verification stage.

This chapter introduces a novel symmetry based vehicle cueing technique. Symmetry is used since it is one of the most important visual characteristics of vehicles. This is particularly apparent for the vehicle's front and rear views, which are normally symmetric over a vertical centreline. It is therefore possible to hypothesise the location of vehicles in the image by searching the regions with high horizontal symmetry.

Object detection based on symmetry is one of the most widely used techniques in computer vision. Symmetry has been used to detect the manufacturing products in industrial applications [78]. It is also used for detecting the human body and face for surveillance applications [79–81]. Different algorithms have been proposed to detect the symmetry axis of an object in the image [80, 82–84]. However, most of the techniques require

that the objects be segmented prior to the symmetry detection. Adding a segmentation process will increase the processing time and is not practical for a real-time vehicle cueing system. Furthermore, getting the precise location of the vehicle's symmetry axis is not critical in the cueing stage. Based on these constraints, this research proposed a fast and efficient vehicle cueing technique. It does not require any prior object segmentation. Yet, it is able to detect multiple vehicles at different sizes in the image.

The chapter begins with some basic concepts about symmetry and the symmetrical object detection. Then the technique of symmetry-based vehicles detection is reviewed. Next, in section 3.4, a fast and efficient vehicle cueing technique is proposed. This is followed by the experiment and results section where the performance of the proposed system is evaluated. Finally a summary is given in the last section.

3.1 Types of Symmetry

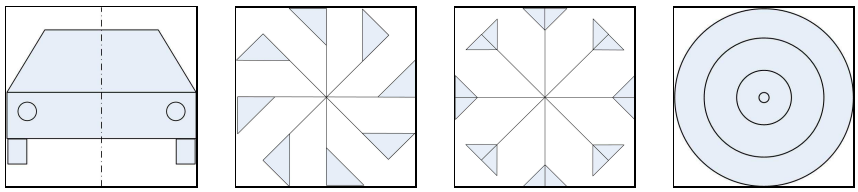
Symmetry can exist in one or multidimensional space. Analysis of two-dimensional (2D) symmetry is of particular interest in computer vision since the image data acquired from a digital camera is commonly in the form of 2D images. In general mathematical notion, an object is regarded as symmetrical if it is invariant as a whole, while allowing some parts permutation under a symmetric transformation [85]. Based on this definition, there are several possible types of symmetries that exist in the 2D Euclidean space, $\mathbb{R}^2$:

- Reflectional symmetry

  An object is reflectional or mirror symmetry if there is a line that can divide the object into two halves, with each half being the exact mirror image of the other. Figure 3.1(a) shows a horizontal reflectional symmetric object. The symmetry axis (dotted line) passes through the centroid of the object.

- Rotational symmetry

  If an object is invariant under rotation of $2\pi/n$ radians about the centroid of the object, then the object is rotational symmetry of order n. It is denoted by C_n. Figure 3.1(b) shows a C_7 rotational symmetric object, it has seven symmetry axes, all of them passing through the centroid of the object.



(a) (b) (c) (d)

Figure 3.1: Types of 2D Symmetries: (a) Reflectional Symmetry (b) Rotational Symmetry (c) Radial Symmetry (d) Circular Symmetry

- Radial symmetry

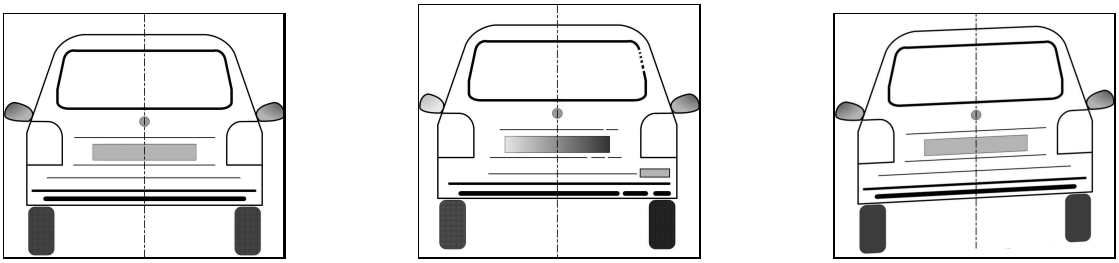
  If a rotational symmetric object is also reflectional symmetry, then the object is radial symmetry. Radial symmetry is a subset of rotational symmetry and is denoted by D_n, where n is the order of the radial symmetry. An example of a D_7 radial symmetric object is shown in Figure 3.1(c).

- Circular symmetry

  An object is circular symmetry if it is radial symmetry of order ∞ (D_∞). Figure 3.1(d) shows an example of a circular symmetric object. Any line that passes through its centroid is a symmetry axis.

An object can be partially symmetric if it does not meet the criteria for perfect symmetry. For example, a partial reflectional symmetric object is not truly symmetry, but there is a line near its centroid that divides the object into two parts, with each part very close to the mirror image of the other (Figure 3.2(b)). Most natural objects are partially symmetric. Images of symmetric objects captured on a camera usually fall into this category due to the uneven lighting conditions and pixels distortion.

When a 3D symmetric object is projected onto a 2D plane, the perspective transformation will cause the object to appear slanted away from its frontal plane. The resulting pattern is called skewed symmetry. Figure 3.2(c) shows an example of a skewed symmetric object. The centre line is now called the skewed symmetry axis. A skewed symmetric object is not reflectional symmetry over its centre line, but all lines that connect the corresponding skewed-symmetric points in the object are parallel and intersect with the skewed symmetry axis at a fixed angle.



(a) (b) (c)

Figure 3.2: Example of (a) True Mirror Symmetry, (b) Partial Mirror Symmetry and (c) Skewed Symmetry

For the context of vehicle detection using a forward looking camera, the rear or front view of a vehicle captured in the image is normally horizontal reflectional symmetry. However, due to the inconsistent outdoor illumination and the perspective distortion, it will appear as partial or skewed symmetry. A robust symmetry-based vehicle detector should be able to tolerate these conditions.

3.2 2D Bilateral Symmetry Detection

Various techniques for 2D symmetry object detection have been proposed in the literature [82, 86–91] and reviewed in [92–94]. In general, the techniques can be broadly divided into the following three categories: direct approach, phase-based and voting-based symmetry detections.

3.2.1 Direct Approach

This approach directly checks whether an object is invariant under a symmetry transformation. This is done by applying a symmetry transform on the image and then comparing the result with the original image. Different comparison methods like Normalised Correlation Coefficient (NCC) or Sum of Absolute Difference (SAD) can be used [29, 95]. An example of using this approach for vehicle detection is shown in Figure 3.3.

The main disadvantage of this approach is that it is very sensitive to occlusion and the selection of the bounding area for the symmetry calculation. For this reason, it will only work well in situations where the object has been correctly segmented. Another

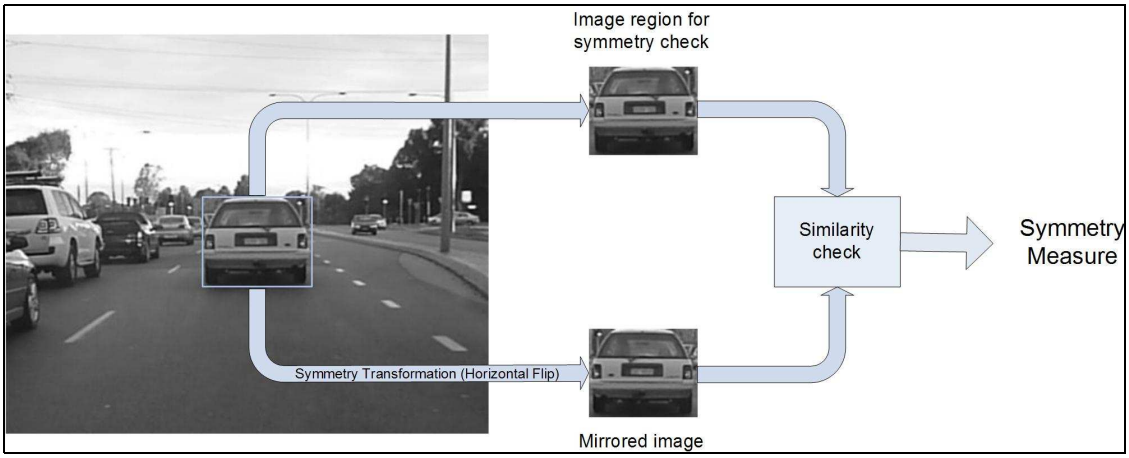


Figure 3.3: An example of vehicle detection using the direct approach. The similarity between the original image and the symmetry transformed image (in this case, the horizontal reflectional symmetry) is checked to determined the degree of symmetry

drawback of this approach is the high computational cost due to the time consuming similarity check. Therefore, this it is not suitable to be used in the cueing stage of the vehicle detection system. At the cueing stage, the vehicles have not been segmented and a fast algorithm is needed in order to process a large image area.

3.2.2 Phase-based Symmetry Detection

This approach detects symmetry based on the analysis of the image's local frequency information. The point of symmetry and asymmetry is recognised from the pattern of the local phase [96].

Two phase based symmetry measures were proposed by Kovesi [96] and Xiao *et al*. [97]. Both measures can be applied directly to the image without needing a prior object segmentation stage. They are also invariant to rotation and independent of the brightness and contrast of the image. In addition, the measures can detect all the reflectional, rotational and curve symmetries at one time.

However, the phase-based approach suffers from a high computational load. In order to analyse the frequency components in the image, it requires the convolution of complex wavelet kernels, the Fourier and Inverse Fourier Transforms and several large matrix multiplications. It is therefore not suitable for a real-time vehicle detection application.

3.2.3 Voting-based Symmetry Detection

This method uses the same principle as the Hough Transform to allow each pair of pixels to vote for their preferred symmetry axis. This is usually done over a symmetry search window. Pixels with a high vote are the good candidates for the dominant symmetry axis. An example of using this technique to detect a one dimensional horizontal reflectional symmetry is illustrated in Figure 3.4. The same technique can be applied to detect 2D reflectional symmetry by using a 2D symmetry search window. In this case, each pair of pixels in the symmetry search window votes for a horizontal line as their preferred symmetry axis.

The size of the symmetry search window can affect the votes and thus the position of the detected symmetry axis. The symmetry detection will be optimal if the size of the search window matches the size of the symmetry object to be detected in the image.

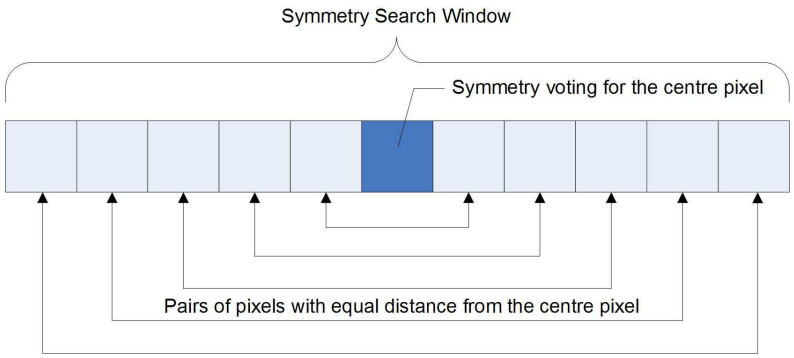


Figure 3.4: Voting-based symmetry detection. Each pair of pixels in the symmetry search window casts a vote to determine the symmetry of the centre pixel

3.2.4 Global Versus Local Symmetry

Two approaches can be considered when performing the symmetry detection on a 2D image. They are the *global* and the *local* symmetry detections. The selection of the best approach depends on whether the object to be detected is symmetry over the whole image or over a local sub-region.

The global symmetry approach treats every pixel in the image as belonging to the object to be detected. Therefore, it is only useful to calculate this symmetry if the object has

been segmented. Otherwise, the images of the background or other objects will affect the location of the detected symmetry axis.

On the other hand, the local symmetry approach calculates the symmetry of a sub-region in the image. It is able to detect small symmetric objects against the background clutter. However, the efficiency of this approach depends on how well the image regions are being selected for the symmetry detection.

3.2.5 Pixel Characteristics for Symmetry Calculation

Several pixel characteristics of the image can be used to calculate the symmetry value. These characteristics include gray scale value, binary contour, horizontal edges, colour and feature points. The following discussions will focus on the use of these pixel's characteristics for symmetry based vehicles detection.

- Gray scale value

  Gray scale symmetry is quicker at isolating the vehicle from its background. It is however prone to influences of illumination variations. An example is shown in Figure 3.5. The reflection of sunlight on one side of the rear windscreen together with the shadow from the surrounding trees caused uneven illumination to the vehicle. Although the vehicle appears to be symmetrical, there are variations in the pixels intensity for its left and right halves in the image. This will affect the accuracy of the symmetry detection. Another problem for the gray scale symmetry is that the uniform area in the image such as road surface and sky will turn out to have high symmetry values (see Figure 3.6). An additional processing step is required to remove these areas prior to the symmetry detection.

- Binary contour

  This approach uses the contour or the edge image for the symmetry calculation. It is insensitive to illumination variations and has no problem with the uniform areas (since these areas have been removed in the edge image). However, the detection can be affected by the symmetric or partially symmetric objects in the background

Figure 3.5: Uneven illumination and reflection on the vehicle's rear windscreen affect the symmetrical appearance of the vehicle

Figure 3.6: The areas of road surface and sky in the image are highly symmetric since they have uniform pixels' intensity

such as overhead bridges, sign boards or buildings. The detection can also be very sensitive to occlusion.

- Horizontal lines

  Horizontal line symmetry is a subset of contour symmetry. It may reduce the influences from the background symmetric objects, but the result depends on how well the horizontal lines of the vehicles can be extracted. Shadow casting on one side of the vehicle may shift the vehicle's bottom horizontal edge to one side resulting in a symmetry axis that is not at the centre of the vehicle.

- Colour

  Colour symmetry is useful for detecting the vehicle's pair of brake lights or the yellow signalling lights. However, similar to the gray scale symmetry, detecting symmetry from colour is very sensitive to illumination variation.

- Feature points

  This approach extracts the feature points from the image and uses them for the symmetry calculation. Examples of feature points are corners [17] and SIFT [98]. The feature points approach can efficiently detect local symmetry against background clutter. However, the accuracy depends on how well the extracted feature points can represent the shape of the vehicles. It also requires extra computational load for the feature points extraction.

3.3 Vehicle Cueing Using Symmetry

Monocular-based vehicle detection for collision avoidance uses a single forward looking camera installed behind the windscreen to capture the road images. This will capture most of the rear view of the preceding vehicles. One prominent visual characteristic of a vehicle's rear view is the horizontal reflectional symmetry. This characteristic can be useful for hypothesising the locations of the vehicles in the image. The advantages for using the horizontal reflectional symmetry are twofold. First, it provides an excellent cue for the vehicle's lateral position. Second, the detected vertical symmetry axis is invariant to the camera's nodding movement, which normally happens due to the vibration of a moving vehicle.

The captured road image may contain multiple vehicles at different locations. Without a prior vehicle segmentation stage, it is only feasible to use the local symmetry to hypothesise the positions of the vehicles. Furthermore, the regions for the local symmetry calculation need to be properly selected in order to efficiently detect vehicles at different locations and in different sizes.

Since the cueing stage is the first stage of the vehicle detection system, a large image area needs to be processed. For this reason, a fast detection technique is required. Although the phase-based symmetry detection approach can accurately detect the local symmetry without any prior object segmentation, it requires a high computational cost and therefore is not suitable to be used at the cueing stage. The accuracy of the symmetry axes detection is also not very critical at this stage since the hypothesised vehicles will be further verified.

3.3.1 Previous Work

Zielke *et al*. [25] proposed a method to detect the centreline of the leading vehicle based on the image intensity symmetry. The vehicle's bounding box is estimated by finding pairs of edges that are mutually symmetric over the detected vehicle's centreline. Symmetry detection based on image intensity is useful for isolating the vehicle from its background. However, it is very sensitive to influences of illumination variations.

Kuehnle *et al*. [29] proposed a system that uses the histogram produced by contour, gray level and horizontal line symmetries for locating a vehicle. Three symmetry axes are calculated and a set rule is used to predict the vehicle's centreline. This method is more robust to noise and illumination variation, but it requires a high computational cost since three symmetry calculations are needed.

Du *et al*. [30] detect a vehicle by contour symmetry. They proposed a technique for detecting horizontal reflectional symmetry by letting each pair of pixels on the same row to vote for their preferred symmetry axis. The location with the highest vote is considered to be the most significant symmetry axis. A drawback of this technique is that it detects the global symmetry axes of the road image. Symmetry axes of distant vehicles may be obscured by the axes produced by the road edges and markings, which could be symmetric over the whole image.

Bin *et al*. [27] use the same symmetry operator to detect a vehicle's vertical axis and its vertical edges at the same time. They assumed that the image of a vehicle is most symmetric at the centre and least symmetric at the edges. However, in order to optimally detect the vehicle and its edges, they need to calculate multiple symmetry values over the whole image, which is a very time consuming process.

In [28], Wei *et al*. used the Saturation (S) component from the Hue-Saturation-Value (HSV) or the Hue-Saturation-Lightness (HSL) colour space to calculate the local symmetry scores of the image. Like the gray scale symmetry, the S-component symmetry is also sensitive to the changes in the lighting conditions. They overcome this problem by calculating a second symmetry measure, the contour symmetry of the shadow underneath the vehicle. However, this will incur extra computational load for extracting the shadow and calculating its contour symmetry.

A colour symmetry based technique was proposed by Hao *et al*. [99]. They calculate the symmetry measure using the three colour components, r, g and b of each pixel. This method can exploit the colour property of a vehicle for the symmetry calculation. However, using three colour components in the calculation will require three times the computational load.

Although different techniques were proposed in the literature, most of them used the principle of the voting based approach to compute the symmetry score. In general, the symmetry score is calculated by summing up the absolute differences in pixel values that are equidistant but in opposite directions from a symmetry centre. The pixels' intensity and contour are the two most common properties used in the calculation.

This review showed that symmetry has been successfully used for vehicle detection. However, most of the techniques proposed in the literature have the following weaknesses:

- They are time consuming since the symmetry calculation is performed on every image row
- The selection of the window size for the symmetry calculation is critical since it will affect the location of the detected symmetry axis and the size of the symmetric objects that can be optimally detected. However, most of the proposed techniques use the same size of symmetry search window for the symmetry detection. This is ineffective to detect vehicles with different sizes in the image
- Most of the proposed technique are not able to locate the centre point of a vehicle. They can only detect the image's column with high symmetry score and use it to hypothesise the lateral positions of a vehicle

In this research, a novel technique for rapid detection of all possible vehicles in the image is proposed. The technique does not require any prior segmentation of the input image. Yet, it is able to locate the centre points and the bounding boxes of multiples vehicles in the image.

3.4 A Variable Window Size and Scan-line Based Symmetry Search Technique

3.4.1 Variable-size Symmetry Search Window

For the local symmetry detection, the selection of the symmetry search window size is very critical since it will determine the effective size of the symmetric objects to be detected. The detection is optimal when the size of the symmetry search window matches the size of the symmetric object.

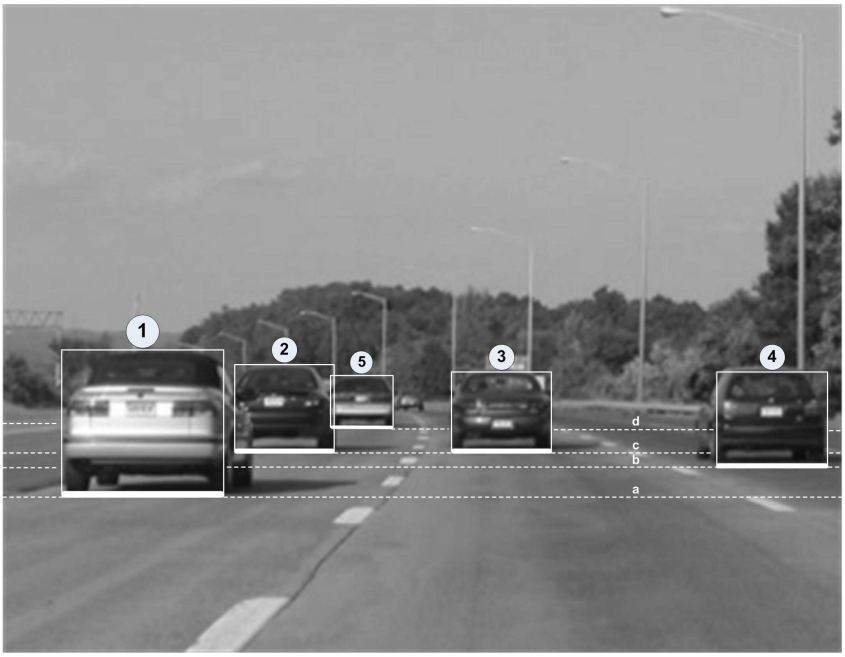


Figure 3.7: A normal road scene with vehicles at different distances: Close vehicle (1) has a bigger image size and its bottom edge is closer to the bottom of the image; the image size of a distant vehicle (5) is smaller and its bottom edge is near to the horizon; vehicles (2) and (3) are at almost the same distance, they show up in the image at almost equal size and their bottom edges are on the same image row. (Road photo taken from [100]).

The road images captured by a forward looking camera may contain vehicles from different distances. Due to the perspective distortion, vehicles at different distances will appear at different locations and in different sizes in the image. An example road image is shown in Figure 3.7. For a distant vehicle, it has a smaller image size and its bottom edge is closer to the horizon. Whereas for a close vehicle, its image is bigger and its bottom edge is nearer to the front boot of the test vehicle. This suggests that we can

estimate the most probable image size of a vehicle when it appears at different rows in the image.

Figure 3.8 shows the plot of the vehicle's bottom row versus the vehicle's width in the image. The data was manually measured from a road video captured during the experiment. As can be seen, the relationship between the vehicle's position and width in the image is almost linear. Based on these data, the best size for the symmetry search windows for different locations in the image can be determined.

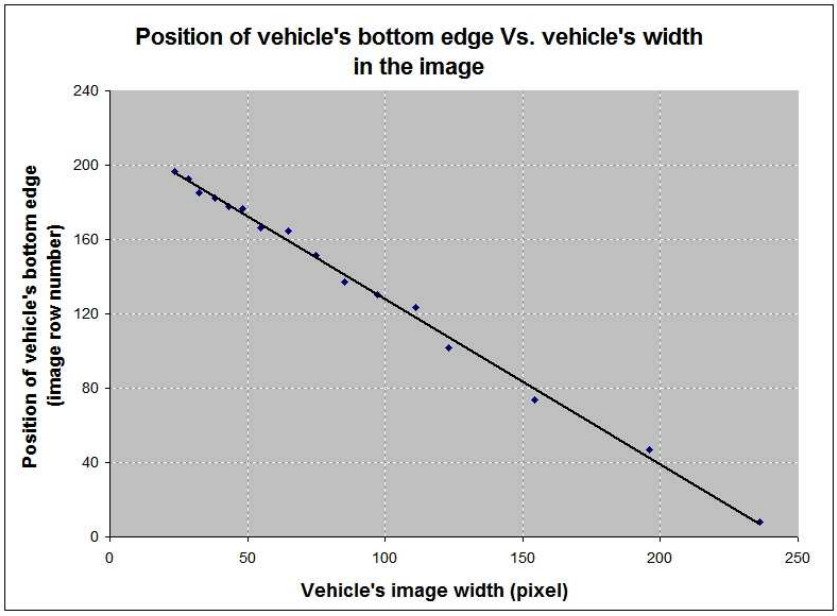


Figure 3.8: Plot of vehicle's bottom position versus vehicle's width in the image.

3.4.2 Scan-line Based Symmetry Search

For a vision-based object detection system, most of the techniques require a full search on the whole image to find the objects of interest. Some symmetry based detection techniques even require the object to be segmented before any symmetry calculation can be performed. These techniques are too time consuming for the vehicle cueing stage. To overcome this drawback, this research proposed a scan-line based sparse symmetry search technique.

The image of a vehicle is normally spanned over several image rows. Based on this

fact, it is possible to detect the vehicle by only processing on several scan lines that pass through the vehicle. A symmetry search window is used to calculate the symmetry profile along each scan line. The peak symmetry points of the profile are clustered and the results are used to hypothesise the centre point of the possible vehicles. This technique eliminates the redundancy of processing all image rows and thus significantly reduces the processing time.

3.4.3 The Proposed Symmetry Cueing Technique

The proposed vehicle cueing technique operates on the input image without the need of prior object segmentation. It uses scan-line based symmetry search and takes advantage of the variable-size symmetry windows to optimally detect vehicles at different distances. From the experiment, it was observed that the proposed symmetry calculation does not require fine image details and thus was carried out on a scaled down image to further reduce the computation time.

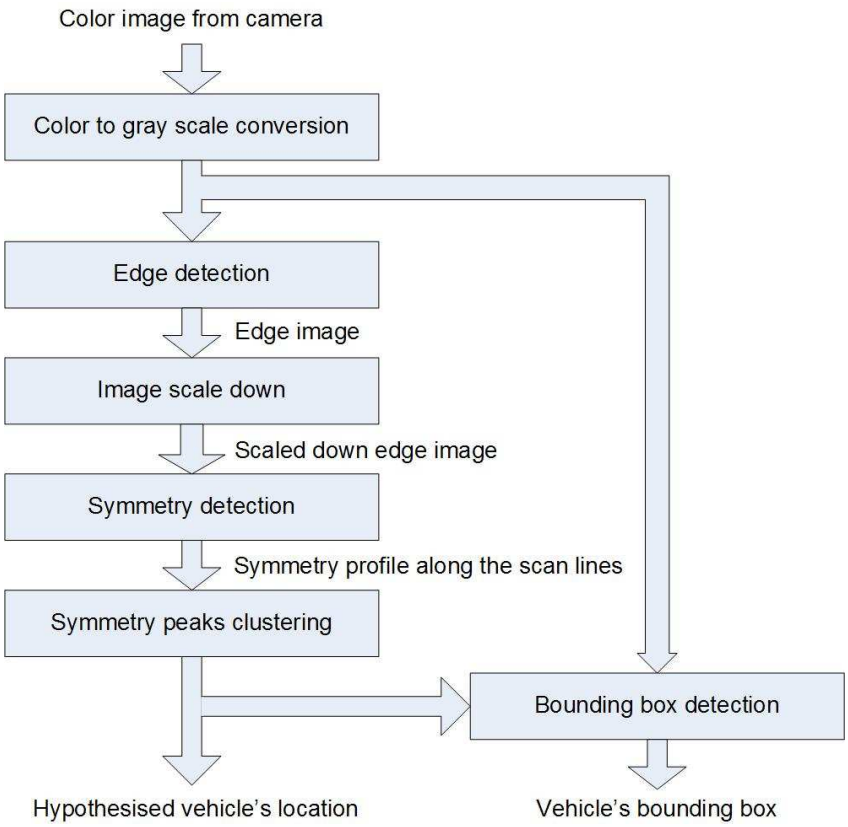


Figure 3.9: Flow chart of the proposed symmetry cueing technique

Figure 3.9 shows the flow chart of the proposed vehicle cueing technique. First, the input colour image is converted to gray scale. The contour image is then generated using an edge detector. Contour-based symmetry detection is used since it is less sensitive to noise and intensity changes in the image. The next step is to scale down the contour image to one quarter of its size. Symmetry calculation is then performed along several scan lines on the scaled down contour image to detect regions with high horizontal symmetry. Finally, the peak symmetry points are clustered and the mean position of each cluster is used as the hypothesised vehicle location. A bounding box for the vehicle can then be estimated around each hypothesised location. This step requires finer image details and therefore is carried out on the original contour image. The following paragraphs explain in detail each step in the proposed vehicle cueing technique.

3.4.4 Colour to Gray Scale Conversion and Edge Detection

The proposed technique operates on gray scale image. Therefore, the colour image from the camera is first converted to gray scale using the following equation [101]:

$$GrayScaleValue = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \tag{3.1}$$

where R, G and B are the red, green and blue components of the colour image.

Next, a global contour image is generated from the gray scale image. Canny edge detector [35] is used for the edge detection. The selection of the threshold values for the edge detection is not very critical as long as it generates enough edges for the symmetry detector. Edge detection is performed on the whole image since the resulting edge image is also needed by the subsequent bounding box detection stage. Figure 3.10 shows the resulting edge image for a typical road scene captured by a forward looking camera.

3.4.5 Determination of Feasible Regions for Vehicle Search

Based on the camera position and perspective constraints, the region in the captured image that most probably contains vehicles is identified. The horizontal limits of this region are between the horizon and the beginning of the road surface at the bottom of the image. Although there are techniques for detecting the horizon [102] and road surface

Figure 3.10: Resulting image after Canny edge detection

[103], these limits were fixed in this implementation to allow a better comparison of algorithms in different road scenes. The vertical limits are set based on the locations of the road boundaries on the left and right hand sides of the image.

Symmetry search is performed in the above selected region to locate potential vehicles. In order to reduce the computation time, scan-line based symmetry search is employed. This is done by performing symmetry calculations on 15 evenly spaced horizontal lines as shown in Figure 3.11. The number of scan lines was chosen based on the number of pixels separating each line and the image size of a distant vehicle to be detected.

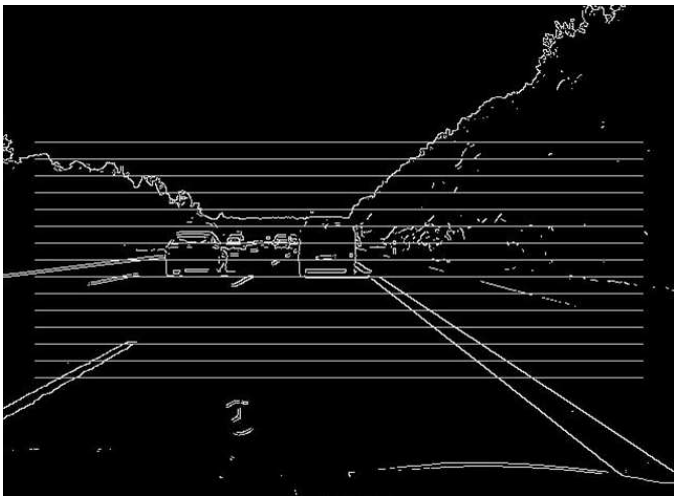


Figure 3.11: Symmetry scan lines

The resolution of the edge image is not very critical for symmetry calculations. This is because the algorithm only needs to detect the symmetry peak points along the scan lines. Base on this fact, the symmetry calculation is carried out on a scaled down edge image to further reduce the processing time. The experiment results in section 3.5.4 show that reducing the resolution of the edge image from the original 640×480 pixels to one quarter of its size (320×480 pixels) does not deteriorate the performance of the symmetry detector.

3.4.6 Contour-based Symmetry Detection

The symmetry value is calculated along the scan lines using the following equation:

$$SymVal(x,y) = \sum_{x'=1}^{W/2} \sum_{y'=y-H/2}^{y+H/2} S(x,x',y'), \tag{3.2}$$

where:

$$S(x,x',y') = \begin{cases} 2, & \text{If } I(x-x',y') = I(x+x',y') = 1, \\ -1, & \text{If } I(x-x',y') \neq I(x+x',y'), \\ 0, & \text{Otherwise} \end{cases} \tag{3.3}$$

W and H are the width and height of the symmetry search window. $I(x,y)$ is the pixel value at location x and y. The value of W needs to be properly set in order to effectively detect symmetric objects of different sizes. From the experiment in different environments and lighting conditions, it was found that the best values for W are between 8 and 20 depending on the position of the scan lines. With this search window's width, it is able to get a high symmetry value for most of the vehicles in the image. Figure 3.12 shows the plot of symmetry values along the symmetry scan lines.

3.4.7 Detection of Peaks in the Symmetry Plots

Non-maximum suppression is used to locate symmetry peaks in the generated symmetry plots. Peaks with symmetry value below a predefined threshold are removed. The

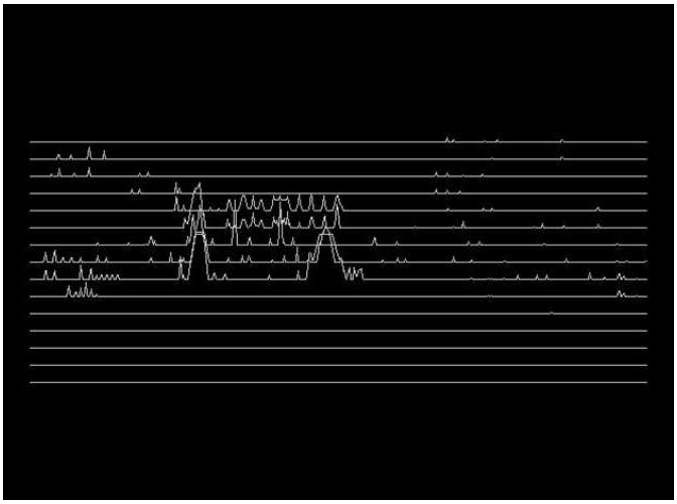


Figure 3.12: Symmetry plots along the scan lines

threshold value was empirically chosen from the experiments conducted on different road images. This value can be easily determined since the peaks generated from most background objects are a lot smaller compared to a vehicle.

$$SymPts(x,y) = \begin{cases} 1, & \text{If } SymVal(x,y) > Threshold, \\ 0, & \text{Otherwise.} \end{cases} \tag{3.4}$$

Figure 3.13 shows the resulting symmetry points overlaid on the captured image. It can be seen that most of the points are located around the vertical centreline of the vehicles.

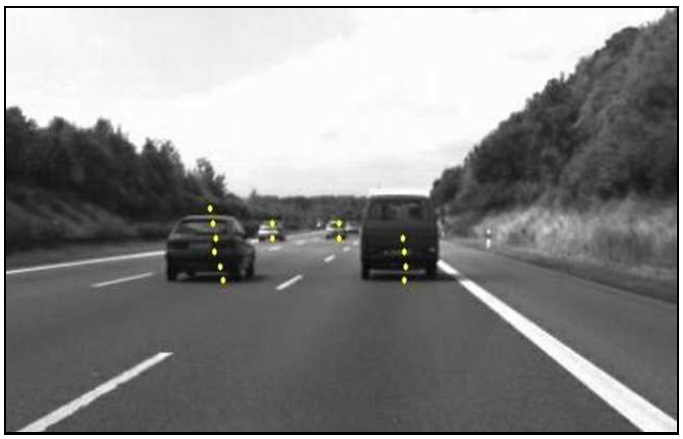


Figure 3.13: Peak symmetry points on each scan line (image enlarged)

3.4.8 Clustering the Symmetry Points

The identified symmetry peak points are then clustered using the k-means clustering technique [104]. The clustering is done in the $x/y-$space to group all points that are close to each other. Since the number of possible vehicles in the image is not known in the beginning, we start the process by clustering the points into two groups. Then, the position variation, σ^2 for each group is evaluated based the following equation:

$$\sigma^2 = \sum_{i=1}^{n} \frac{(x_i - x_{mean})^2 + (y_i - y_{mean})^2}{n}, \tag{3.5}$$

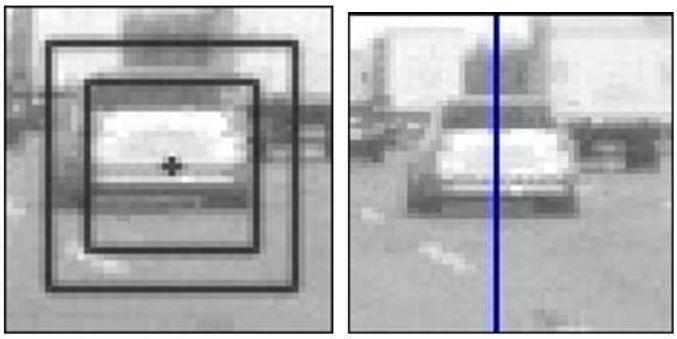
where n is the number of points in a cluster. x_{mean} and y_{mean} are the mean positions of the points.

If the variation is too high, it is most likely that the group consists of more than one vehicle and it will be clustered again into another two groups. This process continues until the variation for each cluster is less than a predefined threshold. The threshold was chosen such that the distance between the mean points, (x_{mean}, y_{mean}) of any two clusters is not less than 20 pixels, which is about the minimum distance between two cars to be detected in the image. Clusters with less than two points will also be removed since they are unlikely to belong to a vehicle's centreline. Finally, the mean points (x_{mean}, y_{mean}) for all the final clusters are calculated and used as the centre points of the hypothesised vehicles. As shown in Figure 3.14, these points provide a good estimate for the centre position of vehicles in the image.

Figure 3.14: Result of clustering the symmetry points. Four clusters were found for this image. The bigger dots are the mean points for the clusters (image enlarged)

3.4.9 Vehicle Bounding Box Estimation

The previous step provides the hypothesised position of the vehicles. Further verification is needed since they might belong to other symmetrical objects on the road. One of the appearance features of a vehicle in the image is strong vertical and horizontal edges. These edges will be detected and used for estimating the bounding box of the vehicle. To do this, a region of interest (ROI) is defined around each detected symmetry point. Since the size of the vehicle is unknown, an initial ROI size of 20×20 pixels is used. This size is then gradually increased up to a fixed limit if no bounding box with acceptable aspect ratio is detected (Figure 3.15(a)). Figure 3.15(b) shows an example of the selected ROI and its vertical symmetry line.



(a) (b)

Figure 3.15: Resizing the ROI for a vehicle's bounding box detection: (a) the ROI centred at the symmetry point is enlarged if no vehicle is detected. (b) An example of ROI, the vertical line is the symmetry axis of the ROI

Figure 3.16(a) shows the edge image of the ROI generated using the Canny edge detector. As shown in the figure, there are many edges not belonging to the vehicle of interest in the centre. These edges need to be removed in order to obtain a more accurate bounding box estimation. Since most of the edges of the vehicle are symmetrical along the vertical symmetry line, we can remove the outlier edges by removing the edge points that are not symmetrical. This technique will also make the selection of threshold values for the Canny edge detection less critical since most of the outlier edges will be removed. The result of this processing is shown in Figure 3.16(b). It can be seen that most of the outlier edges have been removed. The outline of the vehicle is now more apparent.

The next step is to find the vehicle's bounding box. This is done by calculating the

(a) (b)

Figure 3.16: (a) Edge image of the ROI. (b) Enhancing the vehicle's edges by removing the points that are non-symmetric along the symmetry line

vertical and horizontal projection maps of the processed edge image and then evaluating the presence of a rectangular object. The projection vector in the horizontal, **h** and vertical, **v** directions are described in the following equations:

$$\mathbf{h} = (h_1, \ldots, h_n)^T \qquad \text{and} \tag{3.6}$$

$$\mathbf{v} = (v_1, \ldots, v_m)^T \tag{3.7}$$

where

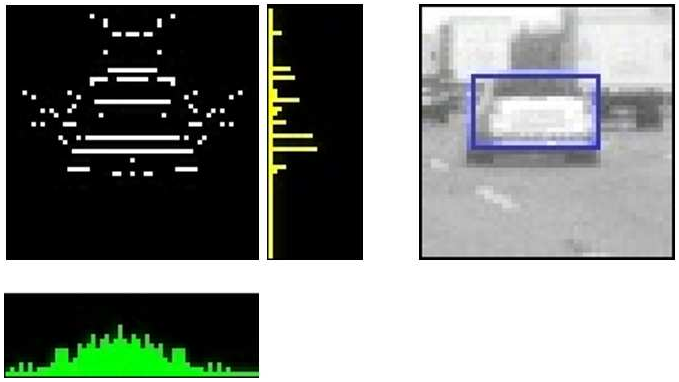
$$h_i = \sum_{j=1}^{m} I(x_j, y_i),$$

$$v_i = \sum_{j=1}^{n} I(x_i, y_j),$$

I(x,y) is the pixel value at location *(x,y)*, *m* and *n* are the width and height of the ROI respectively.

Figure 3.17(a) shows the plot of the horizontal and vertical projection maps. The maximum values of the vertical projection map, θ_v and the horizontal projection map, θ_h are recorded. The bottom boundary of the bounding box, y_B is obtained by finding the position of the first horizontal projection map from the bottom that is higher than $0.5\theta_h$. The top boundary, y_T is determined by the position of the first horizontal projection map from the top that is higher than $0.5\theta_h$. The same technique is used to find the left, x_L and the

right, x_R boundaries of the bounding box from the vertical projection maps and θ_v. The bounding box of the potential vehicle can then be defined by its upper left coordinate (x_L, y_T) and lower right coordinate (x_R, y_B). (Figure 3.17(b)).



(a) (b)

Figure 3.17: Bounding box detection: (a) the horizontal (right) and the vertical (bottom) projection maps of the edge enhanced image. (b) The estimated vehicle's bounding box

The aspect ratio of the bounding box is calculated and will be used as a criterion to check the validity of a vehicle.

$$AspectRatio = \frac{x_R - x_L}{y_T - y_B} \tag{3.8}$$

Most vehicles have an aspect ratio between 0.4 and 1.6. If the calculated ratio falls within this range, then the bounding box is considered to be a potential vehicle. Otherwise, the ROI size will be increased and the same steps for bounding box detection are repeated. The process will stop when an acceptable bounding box is detected or when the maximum allowable size for ROI is reached.

The result of the cueing stage is the position and size of the detected symmetric objects with acceptable aspect ratio. However, not all symmetric objects in the image belong to a vehicle. A rectangular sign board or a building may be symmetric and have an aspect ratio close to a vehicle. These are false detections that need to be removed by the verification stage. This will be discussed in more detail in Chapter 4. But for now, some evaluations on the proposed cueing technique are given in the following section.

3.5 Experiments and Results

This section presents the experiments for evaluating the performance of the proposed vehicle cueing system. The system was implemented on the ImprovCV image processing framework [105, 106] using the C/C++ programming language. The Open Source Computer Vision (OpenCV) Library [107] is used in the implementation. For comparison purpose, all experiments in this section use the same video stream taken on a normal highway. The size of the video is 640×480 pixels (VGA). It was recorded from a forward looking camera installed behind the windscreen of the test vehicle. The performance criteria being evaluated are the detection rate and the processing speed.

Figure 3.18: Implementation of the symmetry-based vehicle cueing system on the ImprovCV image processing framework

Detection Rate

The detection rate was calculated by counting the total number of vehicles detected over the total number of actual vehicles that appeared in the video frames. For comparison of different symmetry algorithms, the centre point of the hypothesised vehicle is used in the detection rate's calculation instead of the bounding box. The rule to decide whether a

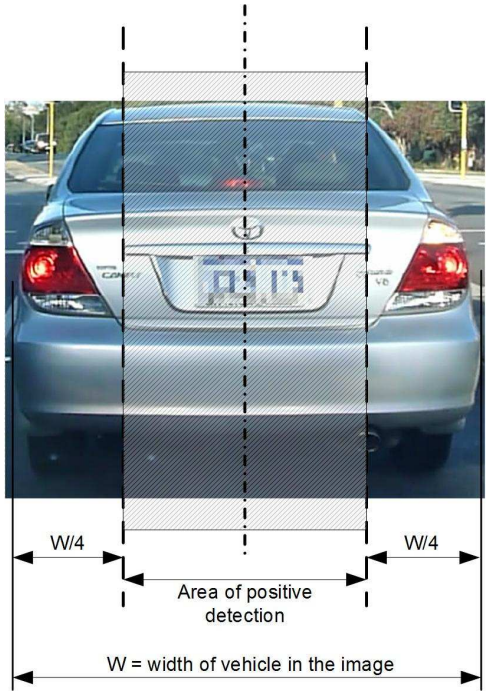


Figure 3.19: Area of positive detection. If the centre point of the hypothesised vehicle fall within the shaded area, it is counted as a positive detection

vehicle is detected is explained in Figure 3.19; A hypothesised vehicle with centre point that falls within the shaded area is considered as a positive detection.

Processing Speed

The experiments were carried out on a standard laptop with an Intel Core Duo 2.0 GHz processor. The processing time was measured by taking the average over several image frames. In order to have a fair comparison between different algorithms, all algorithms were coded without any specific software optimisation.

3.5.1 Performance of the Proposed Vehicle Cueing System

Table 3.1 shows the processing time taken by each component in the vehicle cueing system. The highest processing time is required by the symmetry detection. It includes the calculation of the symmetry profile for every scan line as well as the identification of symmetry peak points. The timings for symmetry peaks' clustering and bounding box detection depend on the number of symmetric objects in the image. If there are more symmetric objects, a higher number of symmetry peaks will be detected and thus a

longer clustering time is needed. Similarly, the process of bounding box detection needs to be repeated for every detected cluster. For comparison purpose, the timings for these two components shown in Table 3.1 are for one detection.

Table 3.1: Breakdown of the processing time for each component in the vehicle cueing system

	Average processing time (ms)
Edge detection	8
Symmetry detection	16
Symmetry peak points clustering	4
Bounding box detection	2
Total	**30**

Table 3.2: The performance of the proposed symmetry-based vehicle cueing system

Number of vehicles appearing in the video frames	755
Detection rate (vehicles detected)	98.94% (747)
Missed detection (vehicles missed)	1.06% (8)
False detection (non-vehicle detected)	5.80% (46)
Processing time (ms)	28

The performance of the proposed vehicle cueing system is given in Table 3.2. The results show that the system is able to detect almost 99% of the vehicles that appeared in the video frames. However, there was also a 5.8% of false detections recorded. These are mainly from the symmetric structures on the road such as sign boards, guardrails and road dividers. Some of the examples are shown in Figure 3.20. These false detections will need to be removed in the verification stage.

(a) (b) (c)

Figure 3.20: Examples of false detections: (a) road divider, (b) guardrail and (c) sign board. The big white dots are the mean points of the symmetry clusters.

3.5.2 Performance Comparison of Scan-line Based Symmetry Search and Full Symmetry Search

In this experiment, the performance of the system using the proposed scan-line based symmetry search and a full symmetry search is compared. For the full symmetry search method, symmetry calculation is carried out on every pixel in the symmetry search region. The results is given in Table 3.3.

Table 3.3: Performance comparison for using scan-line based symmetry search and whole image symmetry search

	Scan-line search	**Whole image search**
Number of vehicles appearing in the video frames	755	755
Detection rate (vehicles detected)	98.94% (747)	100% (755)
Missed detection (vehicles missed)	1.06% (8)	0% (0)
False detection (non-vehicle detected)	5.80% (46)	36.67% (437)
Processing time (ms)	28	168

As expected, the full search method has a major drawback in the processing time; it took almost six times longer compared to the scan-line method. Even so, both methods show comparable high detection rates. One interesting finding is that the scan-line method has a significantly lower false detection rate compared to the full search technique (lower by 30.9%).

These results show that using the proposed scan-line symmetry search technique not only reduces the processing time, but it also lowers the number of false detections while maintaining a reasonably high detection rate.

3.5.3 Performance Comparison for Using Variable Size and Fixed Size Symmetry Search Windows

The proposed system uses different window sizes for each scan line. The size spans from 8 to 20 pixels, depending on the position of the scan line in the image. In this experiment, the performance of this system is compared against systems with a fixed size symmetry search window. Three fixed sizes, which are 8, 20 and 40 pixels were tested. The results are given in Table 3.4.

Table 3.4: Performance comparison for using variable size and fixed size symmetry search windows

Size of symmetry search window (pixels)	**Variable 8-20**	**Fixed 8**	**Fixed 20**	**Fixed 40**
Number of vehicles appearing in the video frames	755	755	755	755
Detection rate (vehicles detected)	98.94% (747)	78.15% (590)	96.03% (725)	74.83% (565)
Missed detection (vehicles missed)	1.06% (8)	21.85% (165)	3.97% (30)	25.17% (190)
False detection (non-vehicle detected)	5.80% (46)	43.70% (458)	5.48% (45)	2.75% (16)
Processing time (ms)	28	26	31	56

From the results, it was found that the variable size search method produced the best detection rate (98.9%). This is followed by the 20 pixels fixed size method (96.0%). The latter has a slightly lower detection rate because it is unable to detect distant vehicles which appear small in the image. These vehicles can only be detected using a smaller window size. The 8 and 40 pixels fixed size methods have poor detection rates (78.2% and 74.2% respectively) mainly because their window sizes for the symmetry search are non optimal for detecting vehicles in the images.

The highest false detection was recorded from the 8 pixels fixed size method (43.7%) while the lowest came from the 40 pixels fixed size method (2.8%). This indicates that using a small symmetry search window will pick up a lot of false detections. However, the proposed variable window size symmetry search technique is able to achieve a high detection rate while maintaining a reasonably low number of false detections because it only performs small window size symmetry searches at locations where distant vehicles could possibly appear in the image.

3.5.4 The Effect of Image Resolution on the Performance of the Vehicle Cueing System

To reduce the processing time, the proposed vehicle cueing system uses a lower resolution image for symmetry detection. In this experiment, the performance of the proposed

lower resolution system (320×240 pixels) was compared with a system that uses full resolution (640×480 pixels). For proper comparisons, the size of the symmetry search window in the full resolution system is also doubled to match the size of vehicles in the image.

Table 3.5: The effect of input video resolution on the performance of the vehicle cueing system

	Reduced resolution (320x240)	**Full resolution (640x480)**
Number of vehicles appearing in the video frames	755	755
Detection rate (vehicles detected)	98.94% (747)	99.07% (748)
Missed detection (vehicles missed)	1.06% (8)	0.93% (7)
False detection (non-vehicle detected)	5.80% (46)	6.97% (56)
Processing time (ms)	28	76

The results in Table 3.5 show that using a lower resolution image for the symmetry detection has little effect on the detection rate (deteriorated by $< 1\%$). But it has an obvious advantage of lower processing time (reduced by 48 ms). The proposed symmetry detection algorithm does not require fine image details since it only detects the symmetry peak points from the edge image. It can, therefore, take the advantage of using a lower resolution image to reduce the processing time. In contrast, the bounding box detection and the verification stage require fine image details and therefore they have to operate on the full resolution image.

3.5.5 Performance Comparison of the Proposed Symmetry Based and the Motion Based Cueing Techniques

In this experiment, the performance of the proposed symmetry cueing technique is compared with a motion based detection technique. To do this, a motion based vehicle cueing system was implemented. Motion based technique detects moving objects in the video by analysing the optical flow fields generated by the objects. The most popular algorithms to calculate the optical flow vectors are the Horn-Schunck [13] (dense optical flow) and the Lucas-Kanade [108] (sparse optical flow) techniques. However, in this implementation, the flow vectors were calculated using the pyramidal Lucas-Kanade

[109] technique which is available in the OpenCV [107] image processing library. The main advantage of this technique over the Horn-Schunck or the original Lucas-Kanade methods is its shorter processing time.

The motion based cueing system first calculates the optical flow vectors from the consecutive video frames. Then the flow vectors are clustered using the K-mean clustering technique. The clustering was done in the magnitude, flow direction and x/y spaces. This is because the flow vectors originated from the same object are usually close to each other and have a similar magnitude and flow direction. Finally the mean position of each cluster is used as the hypothesised vehicle location.

From the experiment, it was found that most of the detected features for the optical flow calculation came from the vehicle's edges and corners. This has resulted in a lot of flow vectors being generated at the edges instead of the centre of the vehicles. For this reason, the rule for positive detection has to be loosened to allow proper comparison. As long as the hypothesised location touches the image of a vehicle, it is considered as a positive detection.

Table 3.6: Performance comparison of the symmetry-based and the optical flow-based cueing systems

	Symmetry based	**Optical flow**
Number of vehicles appearing in the video frames	755	755
Detection rate (vehicles detected)	98.94% (747)	95.63% (722)
Missed detection (vehicles missed)	1.06% (8)	4.37% (33)
False detection (non-vehicles detected)	5.80% (42)	48.47% (679)
Processing time (ms)	28	91

The results are shown in Table 3.6. The detection rate for the motion based technique is slightly inferior (95.6%) compared to the symmetry based technique (98.9 %). However, one key finding is that the motion based technique generated an excessive number of false detections (48.5%). Figure 3.21 shows one of the detection results from the motion based system. Since the video was taken from a moving vehicle, stationary objects such as side posts, guardrails and road markings will also generate flow vectors. This has caused the high number of false detections.

Another major disadvantage of the motion based technique is its higher computational

load. It required three times longer processing time compared to the symmetry based technique. On the whole, the proposed symmetry based technique showed better performance compared to the motion based technique. It will be implemented in the cueing stage of the proposed vehicle detection system.

Figure 3.21: Image shows one of the video frame from the result of motion based vehicle cueing. The arrows are the detected optical flow vectors and the white dots are the hypothesised vehicle's locations

3.6 Summary

This chapter discussed a novel vehicle cueing technique developed in this research. The technique uses multi-sized symmetry search windows to locate symmetry points along several scan lines in the image. The high symmetry points are then clustered and their mean locations are used as the hypothesised vehicle's locations. The proposed algorithm can be directly applied to the input image without needing an object segmentation stage. It also does not require fine image details and thus can be performed on a scaled down image to speed up the processing. The main contribution of this technique is the use of the scan-line based symmetry search which can significantly reduce the processing time. Experiments have shown that this does not affect the detection rate, yet, it is able to reduce the number of false detections. Another key advantage of this technique is the use of multi-sized symmetry search windows for optimal detection of vehicles at different locations in the image. Test results have shown that this can provide a better

detection rate compared to a fixed size symmetry search technique.

A fast and efficient vehicle cueing technique is important for a vehicle detection system. Its main task is to identify all candidate vehicles in the image. The identified candidates will then be further processed in the subsequent verification stage using more sophisticated algorithms. These algorithms provide more accurate results but at the expense of a higher computational cost. However, they only need to be performed on the identified ROIs, which are a small subset of the whole image. In the following chapter, the development of a two-class classifier for vehicle verification is described.

Chapter 4

A Classifier-based Vehicle Verification Technique

This chapter describes the development and evaluation of a vehicle classifier using the pattern recognition technique. The classifier is intended to be used in the proposed vehicle detection system to verify the hypothesised vehicles identified by the cueing stage. A systematic approach was taken to evaluate the performance of different classifiers and image features for vehicle classification. This chapter begins by providing some introduction and theoretical background on several types of classifiers and image features that are commonly used for object recognition. Two techniques of reducing feature dimensions for improving the speed performance will also be discussed. Finally the experiments and the results of the evaluation are presented in the performance evaluation section.

4.1 Introduction

The vehicle segmentation task proposed in this research takes the following two-step approach: vehicle cueing and vehicle verification. The cueing technique uses a simpler but faster algorithm to rapidly identify all possible vehicles in the image. This algorithm is tuned toward low missed rates, but as a consequence, will incur a high number of false detections. The false detections will then be removed in the verification stage using a

more sophisticated algorithm.

Two common techniques for vehicle verification are template matching and image classification. The template matching approach verifies a vehicle by measuring the similarity between the hypothesised image's region and some fixed vehicle's templates. Usually a library of templates consisting of all the standard vehicle's images is created. During the verification process, one or more templates are selected and used in the similarity check. The commonly used similarity checks are the normalised cross correlation and distance measures [110].

However, due to the large variability of a vehicle's appearance, it is difficult to define a set of common templates that can effectively represent all vehicles. This is the major drawback for the template matching approach. For this reason, in most of the literature employing template matching for vehicle verification, a 'loose' or generic shape is selected as the vehicle's template [51, 111]. However, this compromises the accuracy of the algorithm since other objects on the road may also satisfy this shape's constraint.

The learning based approach overcomes this drawback by training a classifier to distinguish between vehicles and non-vehicles. To determine the best classifier for the vehicle verification problem, two most commonly used image features were evaluated in this research. These features are the Histogram of Oriented Gradient (HOG) and Gabor. Their performances were tested on three different classifiers: the Support Vector Machine (SVM), the Multilayer Perceptron Neural Network (MLP) and the Mahalanobis distance classifiers. The evaluations allow a comparative analysis of the features and classifiers under the same experimental setups. The results will enable the selection of the most satisfactory technique to be use in the vehicle verification stage of the final system.

4.2 Vehicle Verification Using a Classifier

This method treats vehicle verification as a classification problem. It involves the training of a classifier based on a large number of examples. The advantage of this technique is that it is not required to explicitly construct a model for the object to be classified.

Defining a generic model for a vehicle is a difficult task due to the large inter-class variability of vehicles.

Figure 4.1 shows the block diagram of the vehicle verification stage. The inputs to this stage are the Region of Interests (ROI) identified by the previous cueing stage. The verification process consists of two successive steps: feature extraction and pattern classification. The feature extraction step extracts some representative features of the object to be classified. In this research, the HOG and Gabor features were considered. Both features have the potential to capture the salient visual patterns of a vehicle, which are basically the local edges and lines at different orientations. A feature selection step may follow to reduce the dimension of the features for improving the classification speed.

The training of the classifier is done offline. For this, a large number of positive and negative training images were collected. During the training process, the classifier finds an optimum decision boundary between the vehicles and non-vehicles. The output of the training is a model file that can be directly implemented in the verification module.

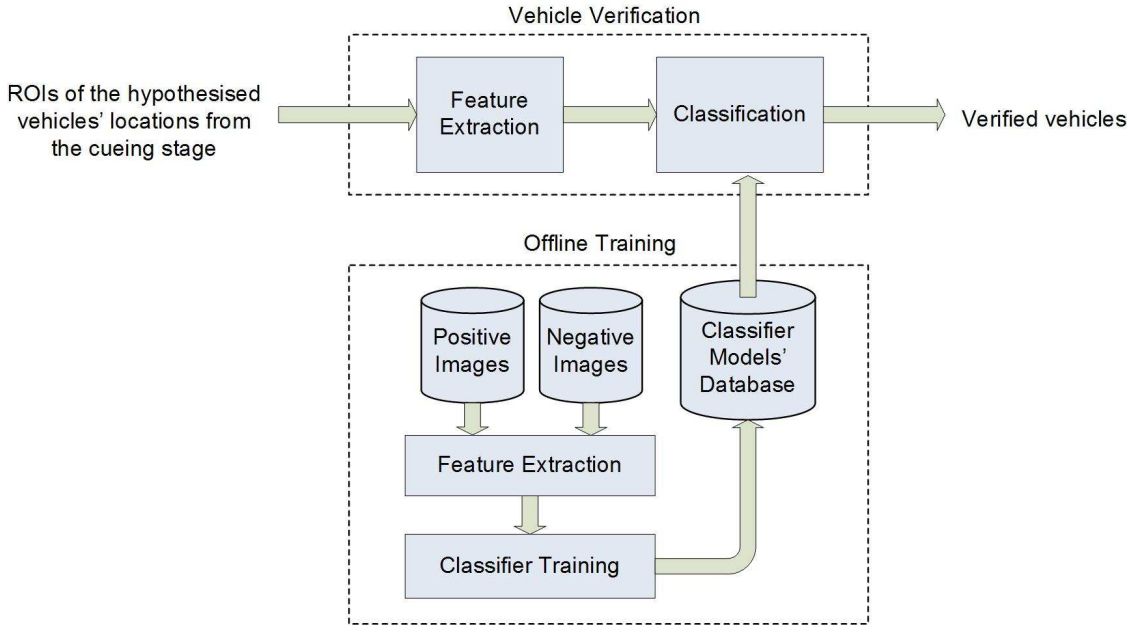


Figure 4.1: The block diagram of a classifier based vehicle verification system

The following subsections describe the three classification techniques that will be evaluated in this research.

4.2.1 Support Vector Machine

SVMs [53, 112, 113] are developed based on the statistical learning theory of Vapnik [53]. It is primarily a two-class classifier. The basic idea of the SVM is to map the training data of two object classes from the input space into a higher dimensional feature space. This is done via a mapping function, ϕ. Then an optimal separating hyperplane with maximum margin is constructed in the feature space to separate the two classes.

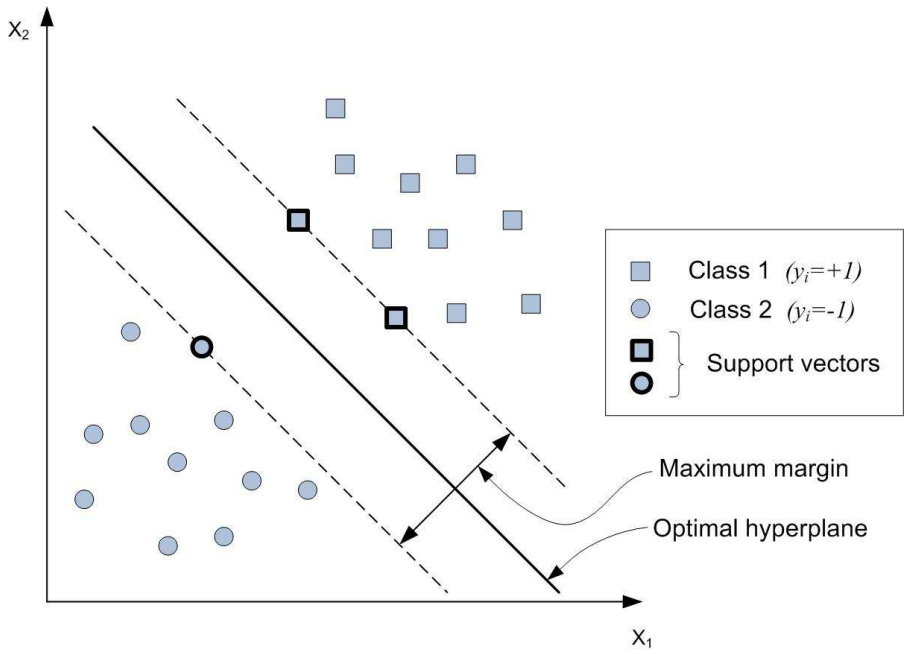


Figure 4.2: An example of the optimal separating hyperplane and margin for a two-dimensional feature space

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

$$(\mathbf{x_i}, y_i), i = 1, 2, ..., l \tag{4.1}$$

where

$\mathbf{x_i} \in \mathbb{R}^N$ 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 defined as [114]:

$$f(\mathbf{x}) = \sum_{i=1}^{l} y_i \cdot \alpha_i \cdot k(\mathbf{x}, \mathbf{x_i}) + b \tag{4.2}$$

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, α_i and bias, b are estimated from the training data. This is done by solving the constrained optimisation problem with the aim of finding a separating hyperplane with maximum margin. Training data that are associated with a non-zero α_i are the support vectors from Class 1 and Class 2. They are the data instances that sit on the boundary in the hyperspace (see Figure 4.2). The optimisation with the SVM resulted in a sparse representation of the input pattern since only a fraction of the training data are support vectors and used in the decision.

$k(\mathbf{x}, \mathbf{x_i})$ is the kernel function. The trick of using the kernel function is that we do not need to explicitly derive the mapping function, ϕ, which in many cases is a non-trivial task. Instead, only the kernel function, which can be evaluated efficiently is used in the training and classification.

The following are some commonly used kernel functions:

1. Linear

$$K(x_i, x_j) = {x_i}^T x_j \tag{4.3}$$

2. Radial Basis Function (RBF)

$$K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2), \gamma > 0 \tag{4.4}$$

3. Polynomial

$$K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0 \tag{4.5}$$

4. Sigmoid

$$K(x_i, x_j) = tanh(\gamma x_i^T x_j + r) \tag{4.6}$$

where γ, r and d are the kernel parameters.

By using different kernel functions, SVMs have the flexibility of implementing different learning machines. In the thesis, the RBF kernel was chosen due to its high performance

reported in the literature [115]. It also requires a simpler parameter selection since there is only one parameter (γ) to be optimised.

The correct choice of kernel parameters is critical for good classification results. In this experiment, we conducted an extensive search on the parameter space and used cross validation to find the parameters that give the best performance. More details about the experiments on the parameters selection are presented in section 4.5.3.

4.2.2 Multilayer Perceptron Feed-forward Artificial Neural Network

Multilayer perceptrons (MLP) are a common type of feed-forward artificial neural network (ANN). Its basic structure consists of one input layer, one output layer and one or more hidden layers. The nodes (also called neurons or units) in each layer are connected in the feed-forward fashion without any feedback paths or connections within the same layer (Figure 4.3). Each connection 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, $g(x)$. Different activation functions can be used, the most common types being linear, sigmoid and Gaussian functions. In [116], Hornik *et al*. showed that an MLP can be trained to approximate any function of interest to any arbitrary precision, provided that there is a sufficient number of hidden units.

The number of input 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 the hidden layers, there is no definitive rules for determining the number of layers or nodes to be used. Having too few hidden layers may cause inaccurate classification. On the other hand, having too many hidden layers will slow down the classification process and may result in overfitting. In practice, the best solution is usually obtained through extensive experiments or based on past experiences for a particular application [117, 52].

All ANNs for supervised learning need to be trained to perform a specific task. The

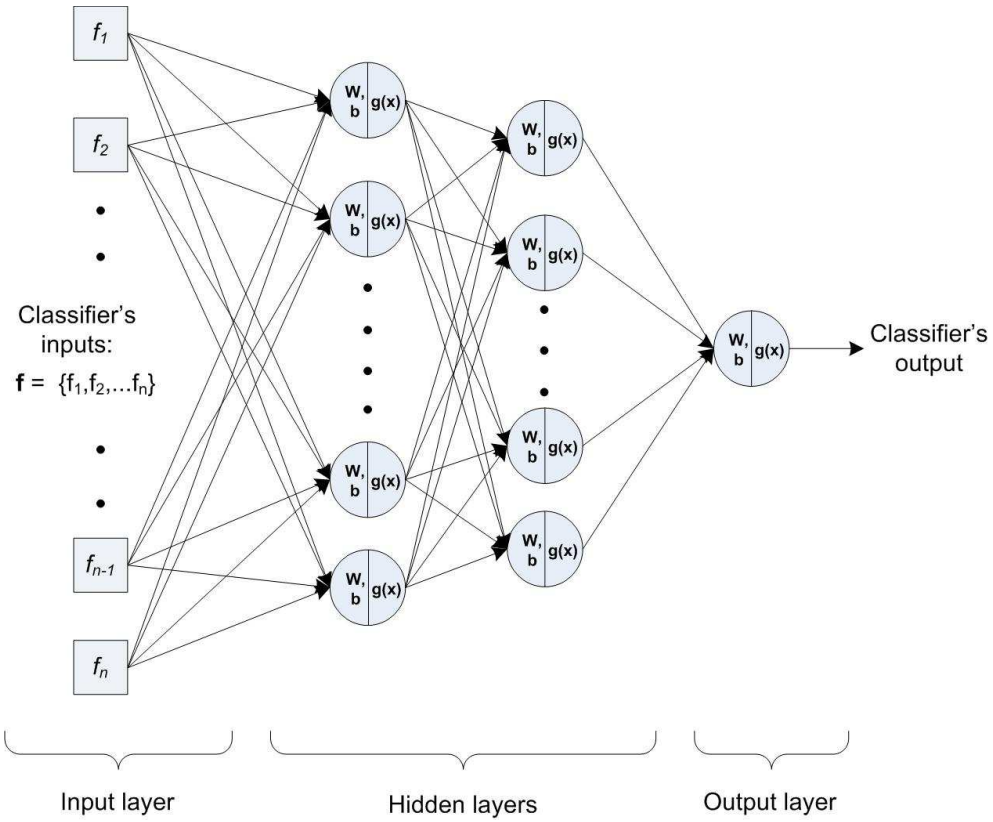


Figure 4.3: An example of the MLP network topology with n inputs, 1 output and two hidden layers

training process updates the connections' weights based on the given training patterns. The most common learning algorithm is the back-propagation technique [118, 119]. This technique performs gradient descent search in the weight space to minimise the errors produced by the network's output. The training process can be divided into two stages: (a) Feed-forward the input training patterns through the MLP to generate the network's output, (b) Compute the error and back-propagate the network to adjust the connections' weights with the goal of reducing the classification error. This process repeats itself for a fixed number of iterations or until the desired minimum error is achieved.

In this chapter, the MLP for vehicle classification is trained using the back-propagation technique. Symmetry sigmoid function (Figure 4.4) is used as the activation function for the nodes in all layers. The performances of the MLP using different networks' configurations were systematically evaluated to find the best network's topology for vehicle classification. The MLPs are trained using the vehicle data set and their cross validation detection rates are compared. The configuration with the best result is then used to

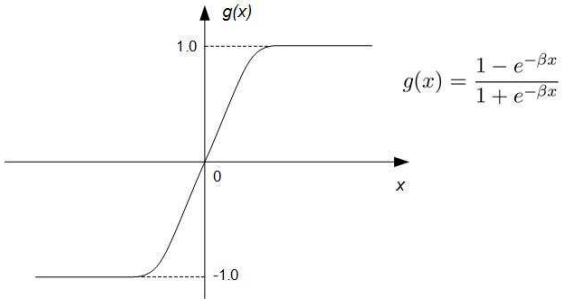


Figure 4.4: The response of the symmetrical sigmoid function. β is the slope parameter

generate the ROC curve.

4.2.3 Mahalanobis Distance

Mahalanobis distance [54] is a measure that can be used to assess the dissimilarity between two sets of variables. It differs from the Euclidean distance in that it takes into account the correlation among the variables in calculating the distance. This is more useful since most of the variables for classification are dependant on each other. Normalising the variables by their covariance makes the comparison among the variables more realistic.

Mahalanobis distance is commonly used as a minimum distance classifier. Usually the distances between an unknown sample and several object's classes are calculated and the sample is classified into a class with the shortest distance. It has been successfully used for applications such as text analysis [120, 121], object classification [122, 74, 123] and medical diagnostics [124–126]. In this study, we used Mahalanobis distance as the baseline for comparing the performances of different classifiers.

The Mahalanobis distance, $D_{Mahalanobis}$ between an N-dimensional vector, $\mathbf{x} = (x_1, x_2, \ldots, x_N)^T$ and a group of vectors with mean, $\boldsymbol{\mu} = (\mu_1, \mu_2, \ldots, \mu_N)^T$ and covariance matrix S is defined as:

$$D_{Mahalanobis}(\mathbf{x}) = \sqrt{(\mathbf{x} - \boldsymbol{\mu})^T S^{-1} (\mathbf{x} - \boldsymbol{\mu})} \tag{4.7}$$

In our experiment, $\boldsymbol{\mu}$ and S represents the vehicle class's distribution. They were calculated from the positive training samples which consists of 5000 vehicle images.

To classify a test image, the Mahalanobis distance between the image and the $\boldsymbol{\mu}$ of each vehicle's class is calculated. If the distance is below a set threshold, the image is classified as a vehicle. In the experiments, the Mahalanobis distances for 800 positive images and 1000 negative images from the evaluation data set were calculated. Then by setting the threshold at different levels, different values of true detection and false detection rates can be calculated. These values are the possible operating points of the classifier. They will be used to plot the Receiver Operating Curve (ROC).

4.3 Image Features Extraction

Usually the image data is not fed directly to the classifier, but it goes through an image preprocessing stage to extract certain image attributes that are pertinent to the given classification task. By removing the irrelevant data which can be the noise for the classification, the performance of the classifier can be improved. Another main reason for features extraction is to reduce the dimensionality of the data to be processed by the classifier. This will not only reduce the computation time for the classification process, but also the time needed for training the classifier. A good feature should be able to capture the distinctive characteristics for the object to be classified. It should also have small variation over different background conditions.

In this study, we investigated the performance of two different image features for vehicle classification. These features, Histogram of Oriented Gradient (HOG) and Gabor, are commonly used for object classifications. The HOG feature has been successfully used for person detection [57, 127–133] and the Gabor feature is often used for texture analysis [67–69, 134]. Although there are several literature reporting the use of these features for vehicle detection [38, 72, 135, 136, 60, 137], there is no comparative evaluation on their performances. In this study, we compared the performance of these two features under the same experimental setups. The results are then used for deciding which classifier is more suitable to be implemented in the vehicle detection system. The following two subsections discuss the HOG and Gabor features and the features extraction process used in this experiment.

4.3.1 Histogram of Oriented Gradient Feature

The HOG feature is obtained by computing the gradient's orientation for each pixel in an image region. The orientations are then quantised into a predefined number of ranges (see Figure 4.5) and their histograms are calculated and used as the feature vector for classification.

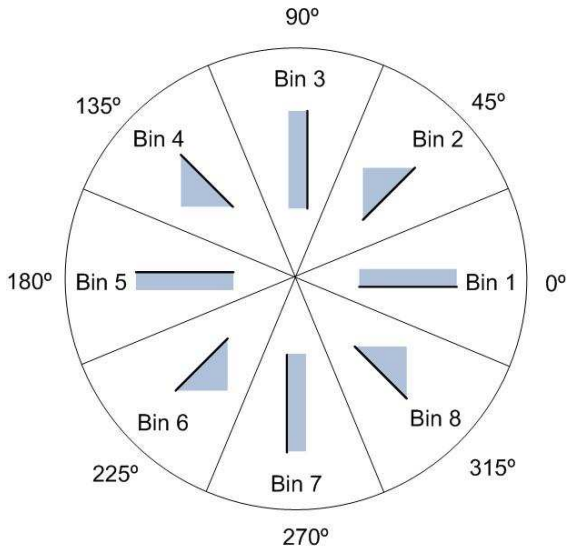


Figure 4.5: The 8 histogram bins for quantising the gradient orientations

We follow the configuration described in [138] to divide the image subregion into cells and blocks to facilitate the HOG feature extraction. The detailed steps are listed below:

1. Scale the image to 32x32 pixels and smooth it with a Gaussian filter
2. Divide the resulting image into 16 cells, with each cell 8x8 pixels
3. Each group of 2x2 neighbouring cells forms a block. A total of 9 overlapping blocks are generated (see Figure 4.6)
4. Calculate the horizontal and vertical gradients (dx and dy respectively) for each pixel, $I(x, y)$ in the block using the following equation:

$$dx = I(x + 1, y) - I(x - 1, y) \tag{4.8}$$

$$dy = I(x, y + 1) - I(x, y - 1) \tag{4.9}$$

5. Calculate the gradient orientation, θ for each pixel:

$$\theta(x, y) = tan^{-1}\left(\frac{dy}{dx}\right) \tag{4.10}$$

6. Accumulate the orientations into 8 histogram bins

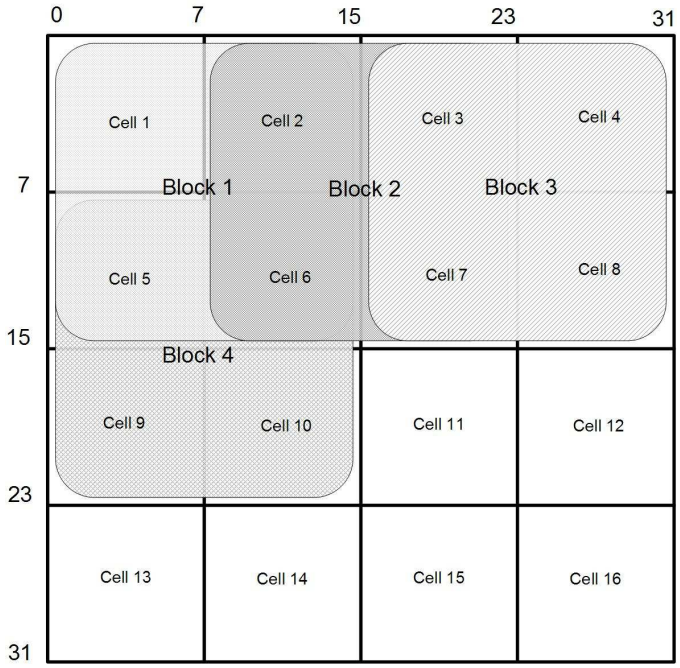


Figure 4.6: Dividing the 32x32 subimage into 16 cells and 9 overlapping blocks for the HOG feature extraction (for clarity, only 4 blocks are drawn in the picture)

The eight histogram values from each block are concatenated to form a 72-dimension HOG feature vector. This feature vector is used for the classifier training.

Variant of HOG Features

Several variants of HOG features were evaluated in the experiment. The variants use the same technique for feature extraction but differ in the following parameters:

1. The number of orientation angles
2. The number of histogram bins
3. The method for calculating the pixel's gradient

The parameters that produce the best classification results were determined through experiments. Figure 4.7 shows the summary for the HOG feature extraction.

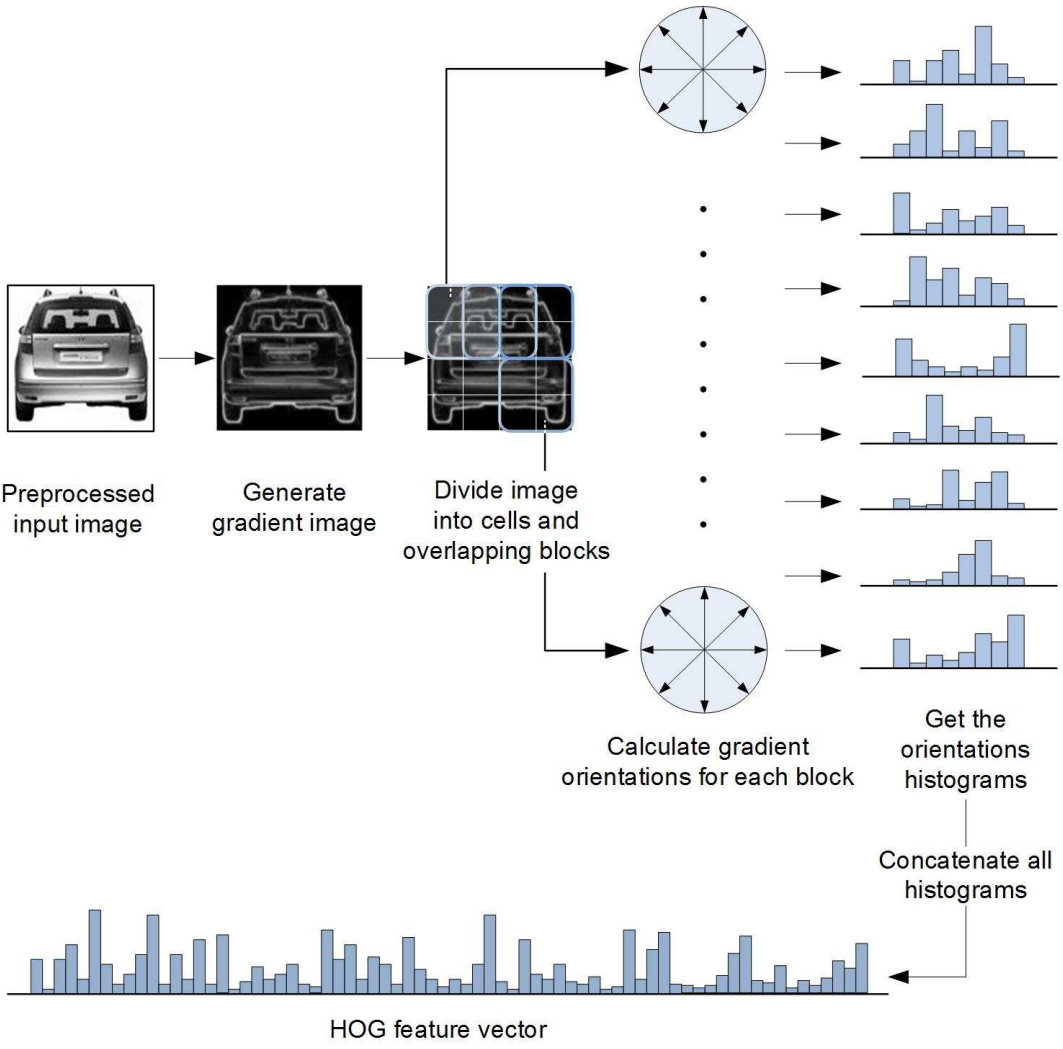


Figure 4.7: Summary of the steps for HOG feature extraction

4.3.2 Gabor Filter

The two dimensional Gabor filter was first introduced by Daugman [139]. He showed that 2D Gabor filters fit well in modelling the simple cells found in the receptive field of the mammalian visual cortex. This finding has inspired many researchers to apply the 2D Gabor filter for image analysis and computer vision. Several literature has reported the successful use of the 2D Gabor filter for texture analysis [67–70], object segmentation [134, 140] and classification [141–144].

In the spatial domain, a 2D Gabor filter is represented as a Gaussian function modulated by a complex sinusoidal wave. It can be formulated as [145, 70]:

$$\psi(z) = \frac{||k||^2}{\sigma^2} \exp\left(-\frac{||k||^2||z||^2}{2\sigma^2}\right) \text{esp}(jkz) \tag{4.11}$$

where $z = (x, y)$ is the location parameters. σ is the standard deviations of the Gaussian factor and $||\cdot||$ denotes the norm operator. The wave factor, k is defined as:

$$k = 2\pi f \, \text{esp}(j\theta) \tag{4.12}$$

where f (in cycle/pixel) is the central frequency of the sinusoidal plane wave and $\theta \in [0, \pi)$ is the orientation of the Gabor filter.

A Gabor filter will respond most strongly to a bar or an edge with a normal that coincides with θ. Different values of f will cause the filter to respond differently to lines or bars with different widths. By changing these parameters (θ and f), a Gabor filter can be tuned for optimal detection of bars or edges at different orientations and scales.

Gabor Feature Extraction

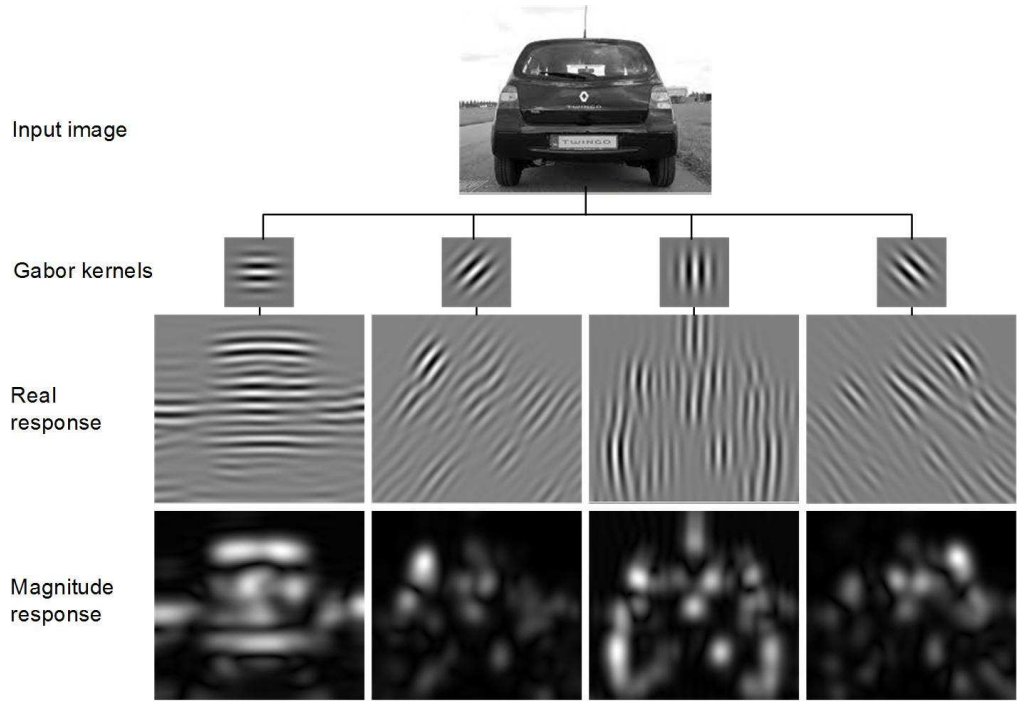
For a given input image, $I(z)$ the Gabor response, $O(z)$ can be obtained by convolving the image with a Gabor kernel:

$$O(z) = I(z) * \psi(z) \tag{4.13}$$

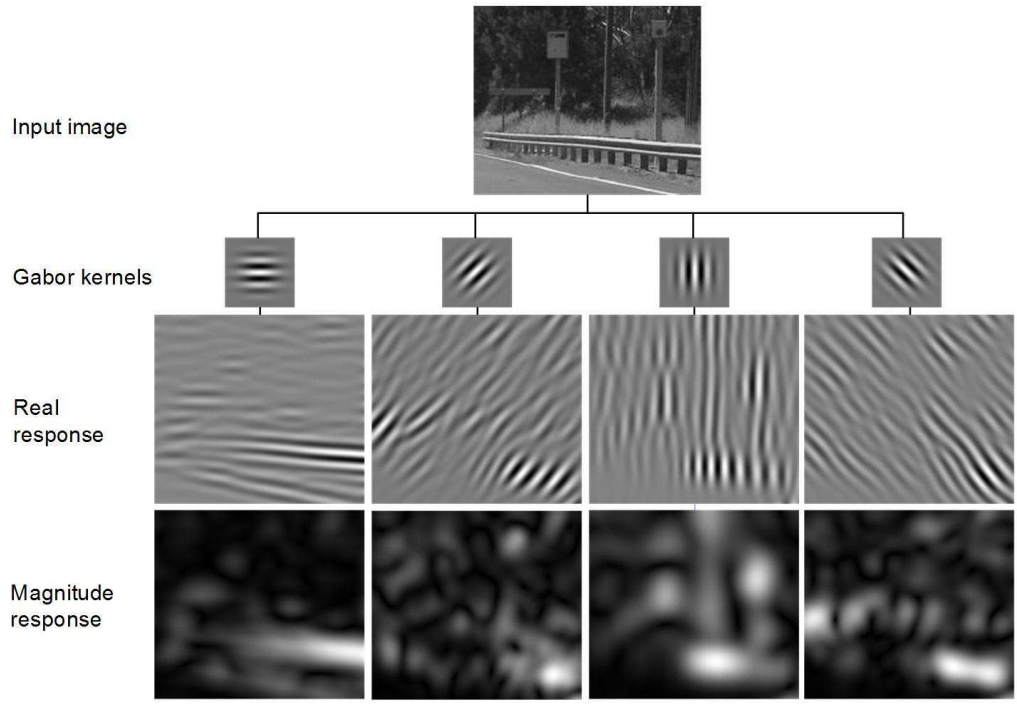
where $*$ denotes a convolution operator.

Usually a filter bank that consists of several Gabor filters with different preferred orientations and scales are used. The convolution results from each filter are put together to form the Gabor features. Note that the Gabor function is a complex-valued equation. This produces a response that consists of a real part and an imaginary part. Figure 4.8 shows some examples of the Gabor responses (real part and magnitude) for a vehicle (a) and a non-vehicle (b). Four different Gabor filters with θ equal to $0°$, $45°$, $90°$ and $135°$ were used in this example.

The choice of values for the filter's parameters is important in order to efficiently extract the vehicle's features. In this study, a set of Gabor filters with different orientations and



(a)



(b)

Figure 4.8: Example of Gabor response for: (a) a positive image and (b) a negative image. Four Gabor kernels with orientations 0°, 45°, 90° and 180° were used in this example.

central frequencies were generated using the following wave factor equation [143]:

$$k_{\mu,\upsilon} = 2\pi f_{\mu} \mathrm{esp}(j\phi_{\upsilon}) \tag{4.14}$$

where $\mu \in \{0, \ldots, m-1\}$ and $\upsilon \in \{0, \ldots, n-1\}$ with m and n represent the number of scales and orientations used in the filters bank. $f_{\mu} = \frac{f_{max}}{\sqrt{2}^{\mu}}$ and $\phi_{\upsilon} = \frac{\pi\upsilon}{n}$ define the different values of scales and orientations. f_{max} is the maximum frequency, it was set to 0.5 pixel/cycle in the experiment. $\sqrt{2}$ (half octave) is used as the spacing factor between different central frequencies. In this experiment, the performance of the Gabor bank, generated using a different number of orientations, m and scales, n were tested.

The Gabor response has the same size as the input image. This generates features with large dimensionality. For example, using a Gabor filter bank with four different orientations and three different scales on an image with size 32×32 pixels will generate $4 \times 3 \times 32 \times 32 = 12288$ features. Due to the large data size, a post-Gabor processing is usually required to produce a more compact feature set.

In this study, we use the statistical moments from the magnitude of the Gabor response to construct the feature vector. The first three moments are considered, they are the mean, μ, the standard deviation, σ and the skewness, γ. A similar technique is used in [70], but they only consider the first two moments (mean and standard deviation). The skewness has been included in our evaluation since it is also a discriminative feature for vehicle classification [71].

Using the statistical moments as the image feature means that the location information will be lost. In order to retain some spatial information, the image region to be processed is divided into cells and overlapping blocks using the same technique as the HOG feature extraction (section 4.3.1). The three moments are then calculated on each block. The feature vector is constructed by concatenating all the results: μ_{ij}, σ_{ij} and γ_{ij}, where i and j represent the filter's number and the block's number respectively.

For example, the feature vector generated using a Gabor filter bank with four orientations and two scales can be represented as:

$$\boldsymbol{f} = \{\mu_{11}, \sigma_{11}, \gamma_{11}, \mu_{12}, \sigma_{12}, \gamma_{12}, \ldots \mu_{89}, \sigma_{89}, \gamma_{89}\}$$

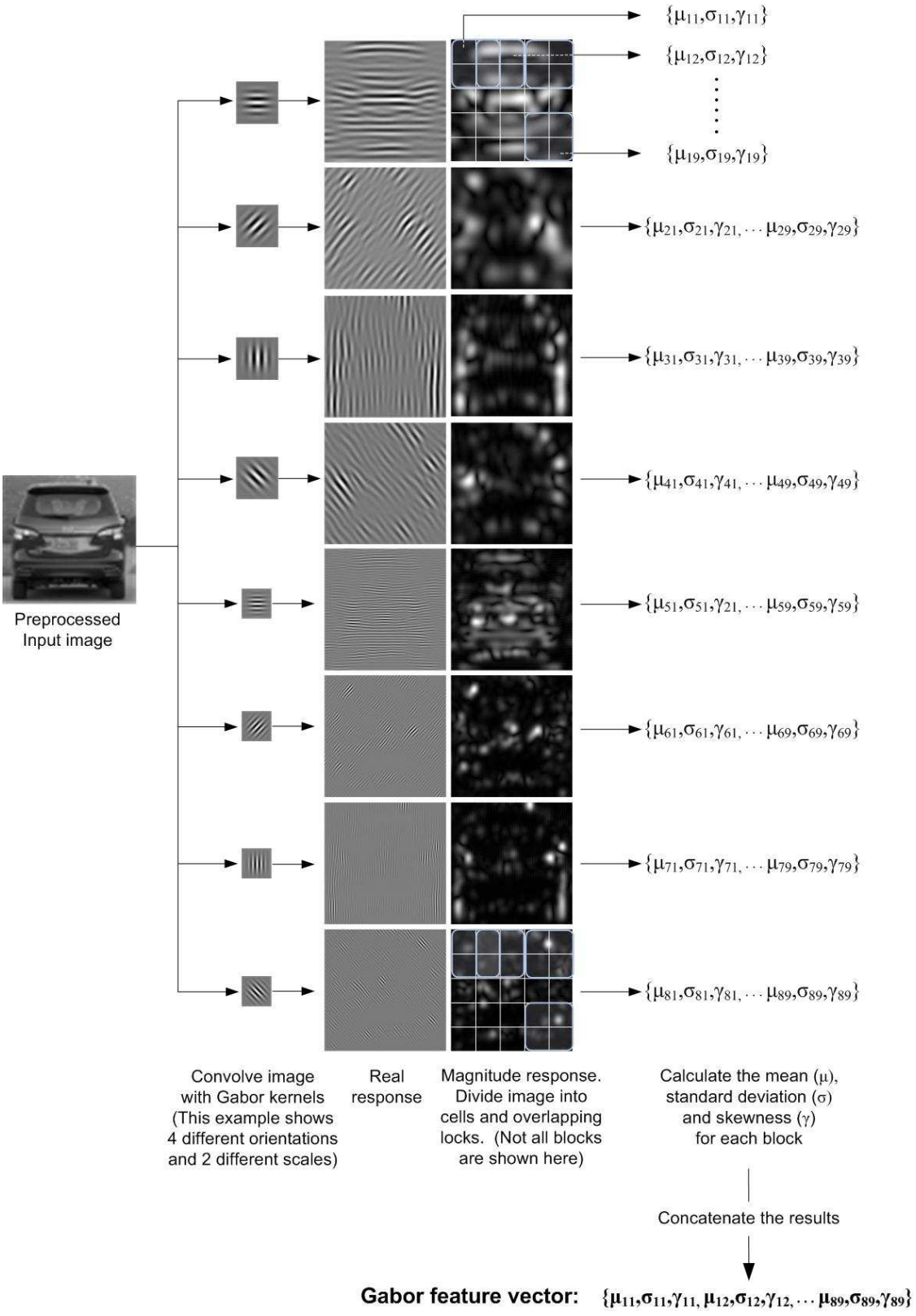


Figure 4.9: Summary of Gabor feature extraction used in this experiment

The size of the feature vector is 216 (4 orientations $\times$ 2 scales $\times$ 3 moments $\times$ 9 blocks). Figure 4.9 summarises the procedures for the Gabor feature extraction used in the experiment.

4.4 Feature Dimension Reduction

The comparatively high dimensionality of image features often causes difficulties to the standard object classification techniques. Therefore, feature dimension reduction is sometimes an essential part for an image classification system. The principal aim for feature reduction is to find a set of lower dimensional features which can still retain most of the information relevant to the classification problem.

There are several advantages for reducing the dimension of the features [146]:

1. It will speed up the classification process due to the lower number of inputs fed into the classifier and thus reduce the amount of data to be processed;
2. It will shorten the time required for the classifier training and reduce the storage required for the training data;
3. It will reduce the cost of features extraction. This can be either the computational cost for calculating the features or in some cases, the cost of the physical sensors;
4. Possible improvement in accuracy due to the removal of irrelevant or unimportant features; and
5. It may reduce the risk of over-fitting by training on a set of 'cleaner' and more compact features;

Feature reduction can be achieved through either feature transformation or feature selection. Feature transformation finds a mapping from the original features' space to a lower dimensional space. It creates a new set of features with a smaller dimension and ideally these features will still be able to preserve most of the relevant information of the original data. Common feature transformation techniques are Principal Components Analysis (PCA) [147], Independent Component Analysis (ICA) [148] and Factor Analysis (FA) [149].

In some applications it is more desirable to use feature selection rather than transformation. Feature selection finds an optimal subset of features from the original features' pool and uses them for the classification. Obviously the main benefit of this technique over feature transformation is the saving of computation time for calculating the transformation and for extracting the unnecessary features. Most feature selection techniques involve the ranking of the original features according to their importance. A subset of features is then selected based on their positions in the ranking.

In this study, we aim to improve the classification speed by reducing the number of features. Feature selection is used to find a subset of the most relevant features for the vehicle classification. Two different features ranking techniques were explored. One is based on the information inferred by the coefficients of the PCA and the other uses the statistical F-score measure.

4.4.1 Feature Selection Using Principal Component Analysis

PCA [147, 150] is a common technique for features' dimension reduction. Conventionally it is used as a feature transformation technique. It first calculates the covariance matrix, C of the original n-dimensional features vector extracted from the training images. Next, it computes all the eigenvectors and eigenvalues from C. There are n possible numbers of eigenvectors that can be calculated. They are then sorted according to the size of their corresponding eigenvalues. These eigenvectors are called the Principal Components (PCs). Each of them consists of n numbers of coefficients and each coefficient is associated with a feature from the original feature space.

It can be proved that the PCs are orthogonal to each other [151] and as a whole they form n orthogonal axes for the data space. When projecting an observation onto these axes, a new set of n uncorrelated features can be derived. The feature derived from the first PC has the highest variance and the subsequent PCs produce features with lower variance. The essence of PCA is that the features derived from the first few PCs are enough to retain most of the variation present in the original data. Therefore, feature reduction can be achieved by discarding the rest of the PCs.

However, the calculation for each derived feature requires $n \times n$ multiplications and the

use of all original features. This will further increase the time for feature extraction and thus deteriorate the classification speed.

In this study, we chose to use a feature selection technique proposed in [147] which utilises the information inferred in the PCs' coefficients for features ranking. This technique can be summarised as below:

1. Calculate the Covariance Matrix and the PCs as described above;
2. Start from the first PC, choose the feature that is associated with the largest coefficient and place it at the top of the ranking;
3. Do the same for each successive PC, but each time picks a new feature that have not been chosen and place the feature in the next position in the ranking.

This approach is reasonable because each PC is associated with a group of highly correlated features. A single dominant feature is enough to preserve most of the information of that group. A feature picked from the first PC has the highest ranking since the projection onto that PC retains the highest variance and thus its underlying group of features are more important.

In this study, we conducted the following experiments to systematically evaluate the performance of different features' subsets:

1. Extract the original feature set. Rank the features according to their importance using the technique described above;
2. Remove four feature with the lowest ranking. Use the remaining to train a classifier;
3. Measure the performance of the classifier using cross validation;
4. Repeat 2-3 on the remaining features. Stop when the classification's performance falls below an undesirable level.

From the experiment, a set of reduced features with comparable performance as the full feature set was identified. The experiments and the results are presented in section 4.5.6.

4.4.2 Feature Selection Using F-Score

F-score is a simple statistical method to measure the discrimination of two sets of real numbers. In [152], Chen *et al*. proposed to use F-score to rank a feature based on the degree of discrimination between the feature and the class labels. Features with a high F-score are more discriminative and therefore are considered to be more important. Several of the literature have also reported the successful use of this technique to reduce the feature's dimension for some biological classification problems [153–155].

Given a set of m positive instances ($x^+_{i,j}$) and a set of n negative instances ($x^-_{i,j}$), where i and j denote the i^{th} sample and j^{th} feature respectively, the F-score of the j^{th} feature is defined as:

$$F(j) \equiv \frac{\left(x_j^+ - x_j\right)^2 + \left(x_j^- - x_j\right)^2}{\frac{1}{m-1}\sum_{i=1}^{m}\left(x_{i,j}^+ - x_j^+\right)^2 + \frac{1}{n-1}\sum_{i=1}^{n}\left(x_{i,j}^- - x_j^-\right)^2} \tag{4.15}$$

where x_j, $x_j{}^+$ and $x_j{}^-$ are the average of the jth feature of the whole data set, the positive set and the negative set respectively.

In general, F-score calculates the ratio of the inter-class variance over the sum of within-class variance. The down side of this technique is that it considers each feature separately and therefore could not reveal the mutual information among the features. However, it is simple to calculate and can be effective for most cases [152].

In this study, F-score is used as a selection criterion to find the important features for vehicle classification. First, the features were ranked based on their F-score values. Then the experiment to evaluate the features' subset as described in the previous section was carried out to find a subset with the best classification performance. The details of the experiments and the results are presented in section 4.5.6.

4.5 Performance Evaluation

In this section, the performance of the vehicle classifier using different combinations of image features and classifiers were evaluated. Two types of image features were tested. They are the HOG and the Gabor features, extracted using the techniques described in

section 4.3. These features were trained and evaluated on three different classifiers: the SVM, the MLP and the Mahalanobis distance classifiers.

The software for this experiment was written in C/C++ programming language. The OpenCV [107] and the libSVM [156] libraries were used in the implementation. There are two main modules in the software. One is for the feature extraction and the other for the classification and the calculation of detection rates. The feature extraction module extracts the required image features for the training and evaluation sets. The extracted features are saved in a feature file using the format described in Appendix 1. The file can then be read by the classification module to train a classifier and to evaluate the classification performance using different performance matrices.

Since the classifier is targeted for used in a real-time vehicle detection system, the speed of the classification will also be measured and compared. All timing measurements were done by the software running on a standard laptop with an Intel Core Duo 2.0 GHz processor. Also, for fair speed comparison, all codes were written without any specific software optimisation.

4.5.1 Performance Matrices

Two performance matrices were used in the experiment. They are the ROC curve and cross-validation. These matrices provide some quantitative measures to enable proper comparison among different classifiers.

ROC Curve

The ROC curve is a common technique for assessing the performance of a classifier. It is a plot of operating points showing the possible tradeoff between a classifier's true positive (TP) rate and false positive (FP) rate. The operating points are obtained by varying some underlying parameters of the classifier and then measuring its TP and FP rates. An example of an ROC plot is shown in Figure 4.10.

In the figure, the classifier with an ROC curve that lies closer to the top left corner (classifier 1) has better performance compared to the other classifier. The area under the ROC curve can also be used as an overall measure for the classifier's performance. The

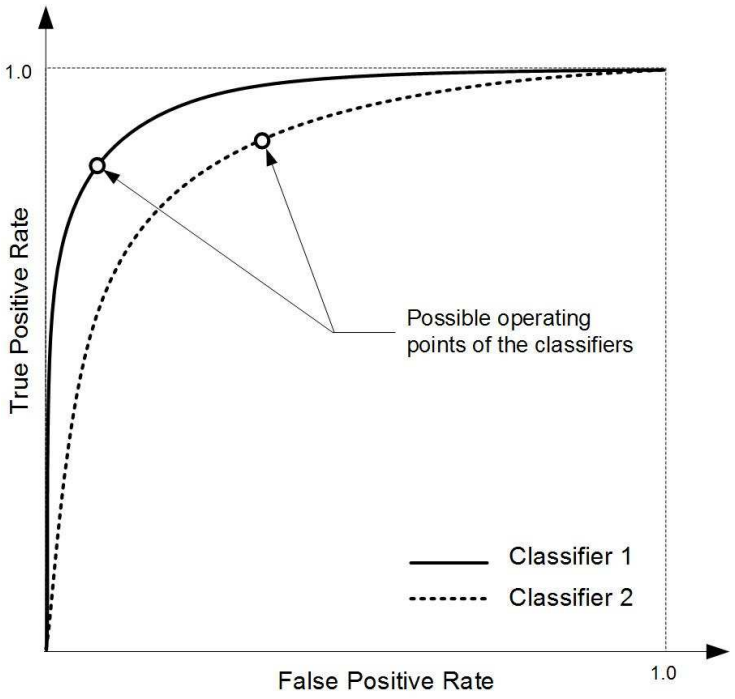


Figure 4.10: An example of the ROC curves

analysis of the ROC curve can also help in the selection of a suitable operating point for the classifier implemented in the final vehicle detection system.

Cross Validation

A classifier that generalises well on the training data might not perform well on the validation set. In such cases, cross validation can provide a more accurate estimate on the classifier's performance. For a *v-fold* cross validation, the data is first divided into v subsets. v-1 subsets are used for the training and the final subset is used for the validation. This process repeats for v times, each time a different subset is used as the validation set. The average of all the validation results is then calculated and used as a quantitative measure for the classifier's performance.

4.5.2 Vehicles Data Set

Due to the large inter-class variability for vehicles, a large training sample was collected for the classifiers training and evaluation. The positive images were either taken around the City of Perth using a digital camera or downloaded from the internet using the image

search engine. The images are randomly split into two groups: the training set and the evaluation set. Some examples of positive and negative images are shown in Figures 4.11 and 4.12.

Figure 4.11: Example of positive training images

Figure 4.12: Example of negative training images

Training Set

The training set consists of 5000 positive images and 6000 negative images. The positive images were originated from 2500 properly cropped vehicle's rear view images. Since the vehicles' rear views are approximately symmetrical, the images were horizontally flipped to generate the 5000 positive training samples. The negative images were selected to cover as many as possible of the non-vehicle objects that could appear on the

road. These include 1200 symmetric objects such as sign boards, buildings and overhead bridges and 4800 non-symmetric objects such as road markings, road dividers, shadows on the road surface and vegetation. The non-symmetric objects were also included in the training set to make the classifier more robust to remove any false positives picked up by the cueing stage. A 3-fold cross validation procedure is used in the training. The classifier's model created from the training is then tested on the evaluation set.

Evaluation Set

This is an independent set of positive and negative images not used in the classifier training. They are reserved for evaluating the performance of the generated classifier and the estimation of operating points for the ROC plot. This set of data consists of 800 positive and 1000 negative images.

4.5.3 Classifier Training

Before using the training images for training or classification, they need to go through a preprocessing stage for feature extraction. Two types of features will be evaluated in this experiment. They are the HOG and Gabor features.

Figure 4.13 shows the steps taken in the preprocessing stage. First, the image is histogram equalised to spread the brightness value. This step is important in order to account for the differences in lighting and contrast in the images. Next, the image is scaled to a fixed size of 32×32 pixels. Finally it goes through the feature extraction step, where different algorithms are applied to generate the feature vector. To maximise the classification performance, the feature values are normalised to the range from -1 to +1 [115]. The feature vectors are stored in a feature file and it can be used by any of the classifier modules for training or evaluation.

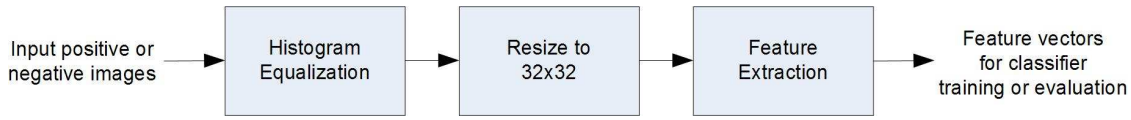


Figure 4.13: Image preprocessing and feature extraction

The output of the SVM or MLP classifier is a fractional value ranging from -1 to 1. An output of -1 represents a non-vehicle while an output of 1 represents a vehicle. The values in between these two limits tell how close the classification is toward the two object classes. For the Mahalanobis distance's classifier, the calculated distance is used in the same manner to assess the probability of an object being classified as vehicle. For all the three classifiers, a simple decision function based on thresholding is used to separate the vehicle from non-vehicle classes. The ROC curve can be constructed by applying different threshold levels at the output of a classifier and then plotting the true positive rate versus the corresponding false positive rate at each threshold level.

Parameter Selection for the SVM and MLP Classifiers

For the SVM classifier, the selection of several parameters in the classifier is very important in order to get good classification result. In this experiment, the values of the parameters are empirically determined using cross-validation. There are two free parameters to select: one is γ from the RBF kernel and the other is C, a noise parameter that controls the tolerance of classification error in learning. In this experiment, an exhaustive search is carried out to find the best γ and C. The search is done in two stages. The first stage performs a coarse search to scan for the best parameters' values across a wider range. Then the second stage makes a finer search around the values identified by the coarse search. Since it is very time consuming to run the parameters' search on the large training set using the 3-fold cross validation, the search was carried out on a subset of the training images. The subset consists of 1000 positive and 1000 negative images randomly selected from the training set. The values that produced the best cross validation result are then used to train the whole training set and generate the final model for the classifier.

For the MLP classifier, its discrimination ability can be highly affected by the network topology used. The topology defines the number of hidden layers and the numbers of nodes for each layer. The number of training cycles (epochs) may also affect the generalisation of the classifier since too many cycles will lead to overfitting. In the experiments, the best network topology and number of training cycles were determined through empirical evaluation. Many networks with different configurations were trained and the

results were systematically analysed. The configuration with the best cross validation result was chosen and used for plotting the ROC curve.

From the experiments, it was found that the best network configuration for the HOG feature consists of two hidden layers with 20 nodes in the first and 10 nodes in the second hidden layer. For the Gabor feature, the best topology from the evaluation is also using two hidden layers, but it needs 50 and 25 nodes in the first and second hidden layers. This difference is largely due to the different in the feature's size for the HOG and Gabor.

4.5.4 Performance Evaluation for the HOG Feature

In this section, the classification performance of the HOG feature is evaluated. Several variants of HOG were considered. All the variants use similar feature extraction technique as described in section 4.3.1. However, they use different parameters for the histogram calculation. The parameters under consideration in this experiment are:

1. The number of histogram bins;
2. The grouping of histogram bins; and
3. The methods for calculating the pixel's gradient.

The results are presented in the following subsections.

Performance of the HOG Feature Using Different Numbers of Histogram Bins

In the HOG feature extraction, the orientation angles are equally divided into several histogram bins. The number of histogram bins determines the fineness of the orientation's details captured by the features. Having too many bins may capture a lot of irrelevant orientations originated from the image's noise or the background clutter. It will also slow down the classification process. On the other hand, using too few histogram bins may miss out some important vehicle's features and thus degrade the classification performance.

In this experiment, the performance of the HOG features with 4, 8, 16 and 32 histogram bins were evaluated (see Figure 4.14). These numbers of histogram bins generate features with sizes 36, 72, 144 and 288 respectively.

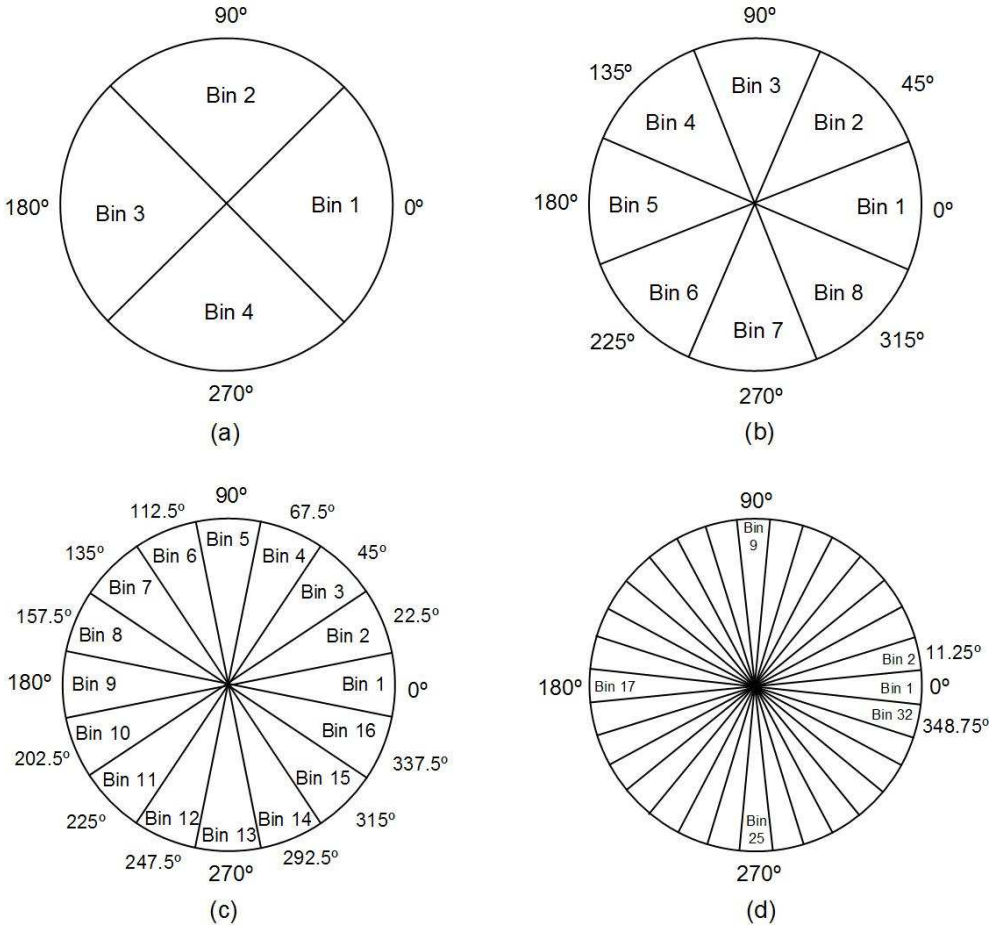


Figure 4.14: Dividing the orientation angles into 4, 8, 16 and 32 histogram bins (shown in (a), (b), (c) and (d) respectively)

Table 4.1: Performance of HOG feature with different number of histogram bins

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

The three-fold cross validation results and the ROC curves are shown in Table 4.1 and Figure 4.15 respectively. From the results, it can be seen that features with higher numbers of orientation bins perform better. When doubling the number of bins from 4 to

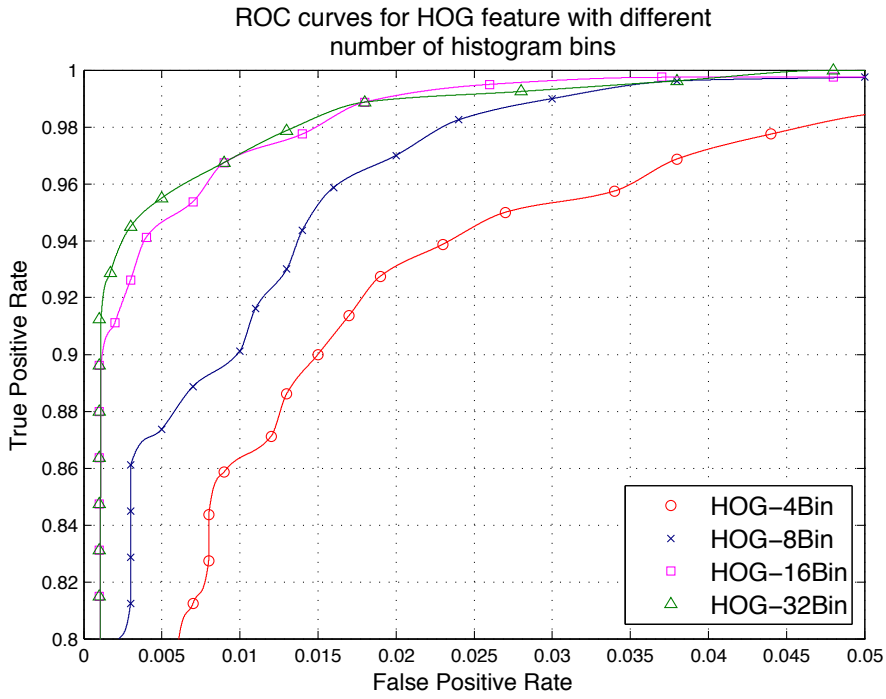


Figure 4.15: ROC curves for HOG feature with different number of histogram bins

8 and from 8 to 16, there are significant improvements in performance (better by 4.7% and 5.9% respectively). However, only a small improvement (<1%) is achieved when doubling the histogram bins from 16 to 32. These results show that some vehicle's features are represented in the finer orientation's details. However, at 16 bins, most of the salient orientation's features have been extracted. Therefore, increasing the histogram bins beyond this number will only gain a little improvement.

Another important finding from this experiment is that the processing time for the classification increases drastically for every increase in the number of features. This is because the complexity of the classifier increases exponentially with the number of inputs. The processing time is another critical factor to be considered when deciding the final feature to be used in the vehicle detection system.

Combining Orientations With Opposite Directions

In the previous experiment, the sign of the gradient's angles was taken into consideration when voting the orientations into the histogram bins. In this experiment, we considered

HOG features that ignore the sign information. This means that orientations with opposite direction are grouped into the same histogram bin (see Figure 4.16).

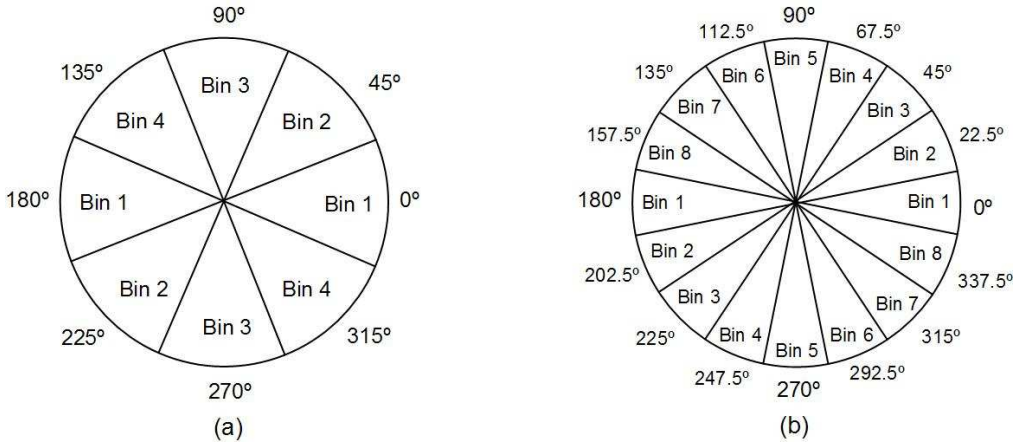


Figure 4.16: Figure shows the grouping of orientations with opposite direction into the same histogram bin. Two examples are shown here: (a) Grouping of 8 orientations into 4 bins and (b) Grouping of 16 orientations into 8 bins

Table 4.2: Performance of HOG feature with and without grouping of the opposite orientation bins

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

The test results are shown in Table 4.2 and Figure 4.17. It was found that in all the test cases, grouping of the opposite orientation's bins has improved the classification performance. One explanation for this is that the 'sign' of the pixel's orientation does not carry any critical information about the shape of the vehicle. By grouping the bins, the confusing 'signed' orientations coming from the image's background can be eliminated. This creates a cleaner set of features which in turn improves the classification performance. In addition, the grouping of the orientation bins reduces the number of features by half. This is a big advantage since it will significantly reduce the processing time.

Note that the feature with 32 orientations grouped into 16 bins is only slightly better than the feature with 16 orientations grouped into 8 bins (better by <0.5%). However,

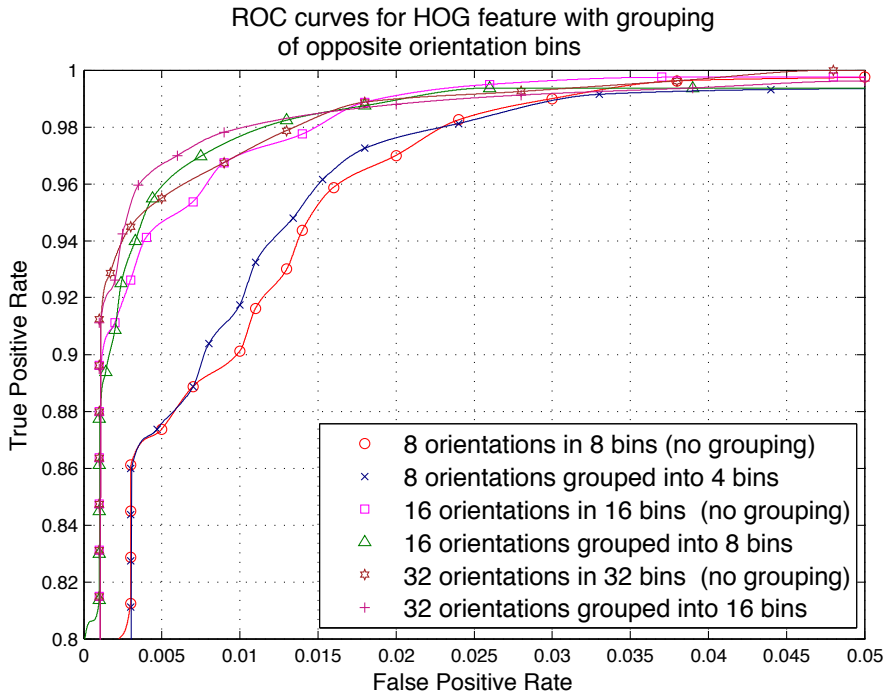


Figure 4.17: ROC curves for the HOG feature with and without grouping of the opposite orientation bins

the latter is more preferable since it has a smaller number of features and only requires about one third of the processing time. Therefore, this configuration is chosen to be used in the subsequent experiments.

Performance of the HOG Feature Using Different Methods of Gradient Calculation

The pixel's orientations are calculated from the horizontal and vertical gradients (dx and dy respectively). These gradients can be computed using different methods. In this experiment, the effects of using different gradient's calculations on the classification performance were evaluated. The following techniques for the gradient's calculation were considered:

1. **1D non-centred derivative:**

   $dx = I(x+1, y) - I(x, y)$

   $dy = I(x, y+1) - I(x, y)$

This is equivalent to convoluting each pixel with the following convolution masks:

$$G_x = \begin{bmatrix} -1 & 1 \end{bmatrix}, G_y = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

2. **1D centred derivative:**

   $dx = I(x+1, y) - I(x-1, y)$

   $dy = I(x, y+1) - I(x, y-1)$

   The equivalent convolution masks:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}, G_y = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

3. **3x3 Sobel masks:**

   $dx = I(x+1, y-1) - I(x-1, y-1) + 2 * I(x+1, y) - 2 * I(x-1, y) +$
   $I(x+1, y+1) - I(x-1, y+1)$

   $dy = I(x-1, y+1) + 2 * I(x, y+1) + I(x+1, y+1) - I(x-1, y-1) -$
   $2 * I(x, y-1) - I(x+1, y-1)$

   The equivalent convolution masks:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

The three-fold cross validation results and the ROC curves are shown in Table 4.3 and Figure 4.18 respectively.

Table 4.3: Performance of HOG feature with different methods of gradient calculation

Types of gradient calculations	Detection rate	Area under the ROC curve	Processing time (ms)
1D non-centred derivative	98.4%	0.9985	22.45
1D centred derivative	97.8%	0.9980	22.78
3x3 Sobel masks	96.2%	0.9968	22.95

The results show that the 1D non-centred derivative method gives better performance compared to the centred derivative and the sobel mask methods (better by 1.6% and 0.6% respectively). This implies that the gradient calculation using less neighbouring pixels provides better performance. One possible reason for this is that the local pixel's

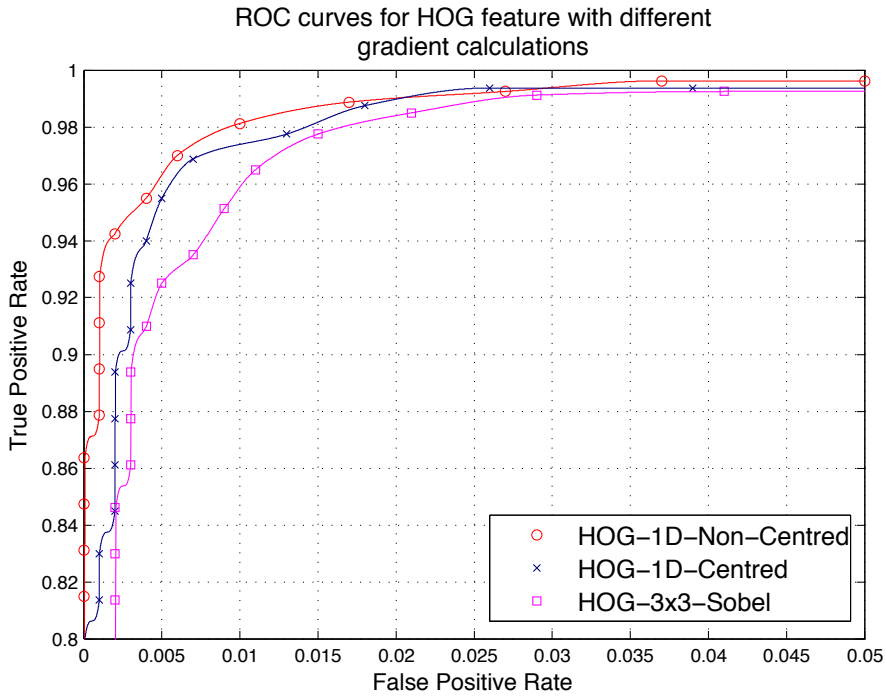


Figure 4.18: ROC curves for the HOG feature with different methods of gradient calculation

orientation that represents the shape of a vehicle can be better extracted using a smaller convolution mask. Larger convolution mask may dilate this information and cause the loss of some detailed features. For the processing speed, there is not much difference among the three techniques. Therefore, the 1D non centred method is considered to be the best method and will be used in the subsequent experiments.

Performance of HOG Feature on Different Classifiers

The previous experiments used SVM for classification. In this section, we investigate the performance of the HOG feature on two other classifiers introduced in sections 4.2.2 and 4.2.3. They are the MPL artificial neural network and the Mahalanobis distance classifiers. For this experiment, we used the HOG feature with 32 orientations grouped into 16 bins and 1D non-centred gradient calculation. This is the best feature identified so far from the previous experiments. The test results are shown in Table 4.4 and Figure 4.19.

From the results, it is obvious that the SVM outperformed the other two classifiers.

Table 4.4: Performance of HOG feature on different classifiers

Types of classifiers	Detection rate	Area under the ROC curve	Processing time time (ms)
SVM	98.4%	0.9985	22.45
MLP ANN	96.5%	0.9972	8.62
Mahalanobis distance	78.1%	0.9750	3.10

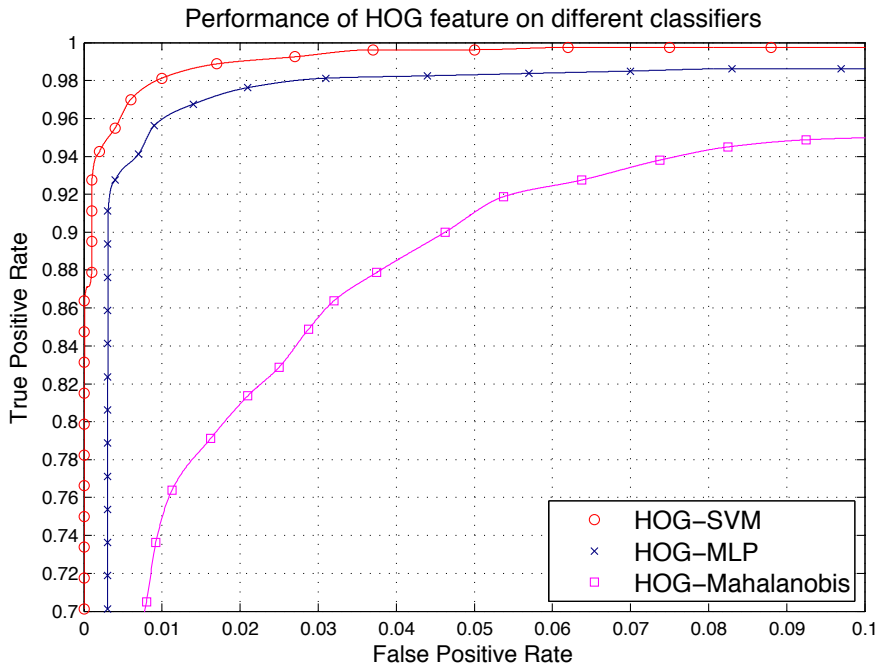


Figure 4.19: ROC curves for HOG feature using different classifiers

Mahalanobis distance is the fastest classifier. But unfortunately its classification performance is far inferior. Although the SVM is the best performing classifier, it requires the longest processing time. Yet, it is still considered to be the best candidate for the vehicle detection system since this processing speed is acceptable for real time implementation.

4.5.5 Performance Evaluation for the Gabor Feature

This section evaluates the performance of the Gabor features. First, we carried out some experiments to find the best parameters for the Gabor feature extraction. Then we select the best Gabor feature and analyse its performance on different classifiers. Finally we compare the performance of the best Gabor-based classifier and the best HOG-based classier identified in the previous section.

Performance of the Gabor Feature With Different Scales and Orientations

In this experiment, we evaluated the performance of the Gabor features extracted using different filter's parameters. To do this, we first created several Gabor filter banks with different numbers of orientations and scales. Then the filter banks were used as the kernels to generate different Gabor features using the technique described in section 4.3.2. For this experiment, the SVM classifier was used to evaluate the performance of the generated features.

Table 4.5: Performance of Gabor feature with different numbers of orientations and scales

Number of Orientations	Number of scales	Generated feature size	Detection rate	Area under the ROC curve	Processing time (ms)
2	2	108	92.8%	0.9937	163.42
2	3	162	93.1%	0.9928	286.40
4	2	216	95.0%	0.9953	468.66
4	3	324	95.0%	0.9951	854.16
6	2	324	95.8%	0.9970	838.44
6	3	486	96.0%	0.9973	1426.48
8	2	432	96.2%	0.9977	1234.50
8	3	648	96.3%	0.9975	1952.50

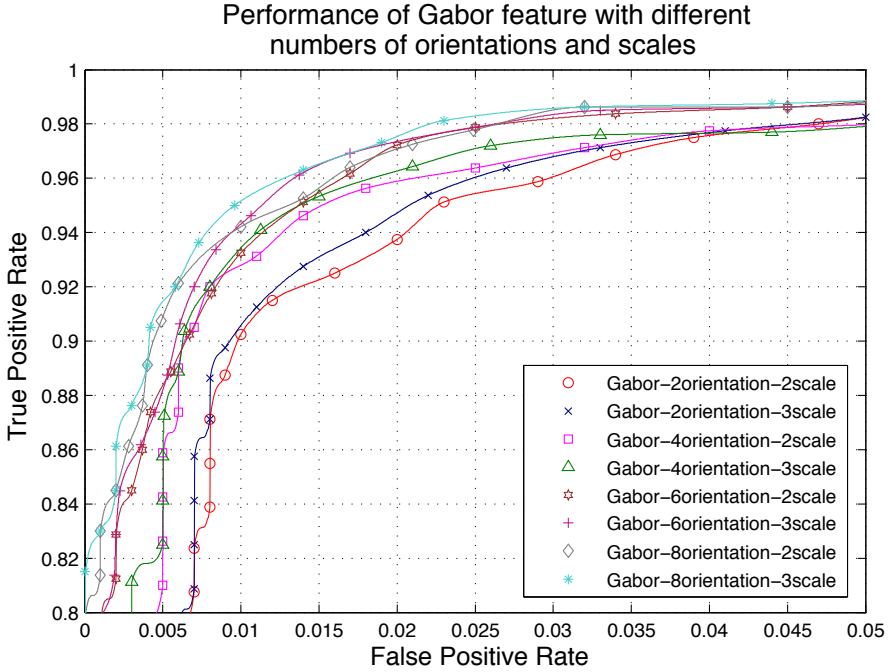


Figure 4.20: ROC curves for Gabor feature with different numbers of orientations and scales

Table 4.5 and Figure 4.20 show the evaluation results. In general, the classification performance increases with the number of orientations and scales used in the feature extraction. At two orientations, it is equivalent to detecting only the horizontal and vertical edges and therefore it has the worst performance. When using more orientations, more vehicle's edges at different angles were able to be detected and thus the classification performance improved. However, the improvement became less significant once the number of orientations exceeds four. This is possibly due to the fact that most of the important vehicle's edges have already been extracted. The results also show that the number of scales used in the feature extraction can affect the classification performance. Nevertheless, the impact is not as significant as the orientation. The overall classification time of the Gabor feature is much higher compared to the HOG feature. This is mainly due to the bigger feature's size and the more complex feature extraction process required by the Gabor feature. The best performing configuration (eight orientations and three scales) requires close to two seconds of processing time which is obviously not suitable for any real time implementation.

Performance of the Gabor Feature on Different Classifiers

This section compares the performance of the Gabor feature trained on the SVM, MLP and Mahalanobis distance classifiers. The best Gabor feature identified in the last section was used in the evaluation. The cross validation results and the ROC curves are shown in Table 4.5 and Figure 4.20 respectively.

Table 4.6: Performance of Gabor feature on different classifiers

Types of classifiers	Detection rate	Area under the ROC curve	Processing time time (ms)
SVM	96.3%	0.9975	1952.50
MLP ANN	89.4%	0.9882	620.72
Mahalanobis distance	65.2%	0.9261	237.60

Similar to the HOG feature, the Gabor feature trained on the SVM classifier produced the best result. This is followed by the MLP classifier and then the Mahalanobis distance classifier. However, the SVM classifier requires far higher processing time compared to the other two classifiers. For comparison reasons, the ROC curve for the best HOG-SVM classifier (from the experiment in section 4.5.4) is also plotted on the same graph. As

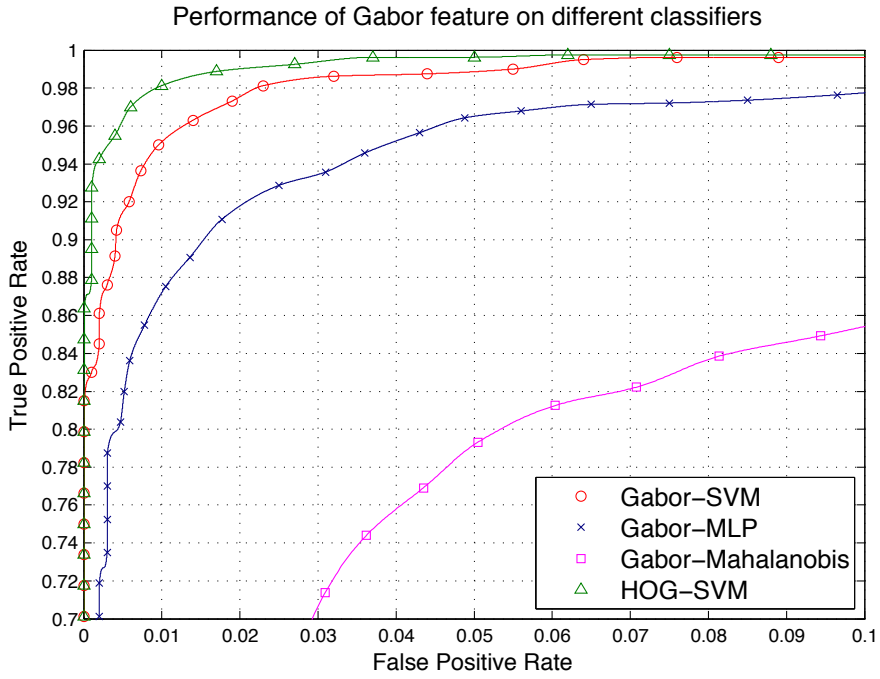


Figure 4.21: ROC curves for Gabor feature on different classifiers

can be seen, the performance of the Gabor-SVM classifier is inferior compared to the HOG-SVM classifier. Besides the classification performance, the Gabor-SVM classifier also required a considerably longer processing time (1.95 s) compared to the HOG-SVM classifier (22 ms).

4.5.6 Performance Evaluation for Using Reduced Feature Sets

In this section, we picked the best classifier identified from the previous sections (HOG-SVM) and attempted to improve its speed performance. This can be achieved by reducing the number of features using feature selection. Two feature selection techniques introduced in section 4.4 were evaluated. Both techniques rank the features according to their importance for the classification problem.

The first technique uses the information inferred in the principal components (PCs) from the Principal Component Analysis (PCA). In this experiment, first, the HOG features were extracted from 5000 positive training images. Then the PCA is performed on the extracted data. The next step is to analyse the coefficients of the PCs and pick one

dominant feature from each PC. The ranking of the features is based on the order of the PC where they came from.

In Figure 4.22, the percentage of variance accounted by each PC is shown in the bar chart. The line plotted on the same graph shows the cumulative percentage of the variance. It can be seen that the first few PCs accounted for most of the variance of the data. The cumulative plot indicates that 80% of the total variance came from the first 16 PCs. This suggests that it is possible to use a subset of the features to represent the full data set.

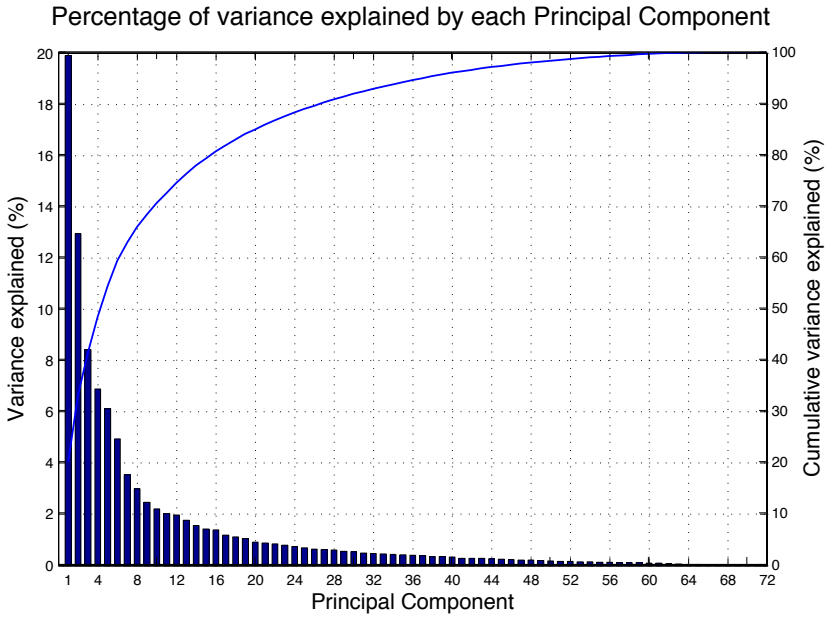


Figure 4.22: Percentage of variance explained by each Principal Component

The second feature selection technique is based on F-score. The F-score's value for each of the HOG feature were calculated and they were used for feature's ranking. A feature with a higher F-score is considered to be more important and it is placed at a higher rank. In Figure 4.23, the bar chart shows the F-score values for all the features arranged in descending order. The line plotted on the chart is the accumulated percentage of the F-score values. Similar to PCA, there are a few dominant features with very high F-score values. Also, the top 25 features have accounted for more than 80% of the total F-score values. This indicates that a subset of features with high F-score is enough to retain most of the information of the whole data set.

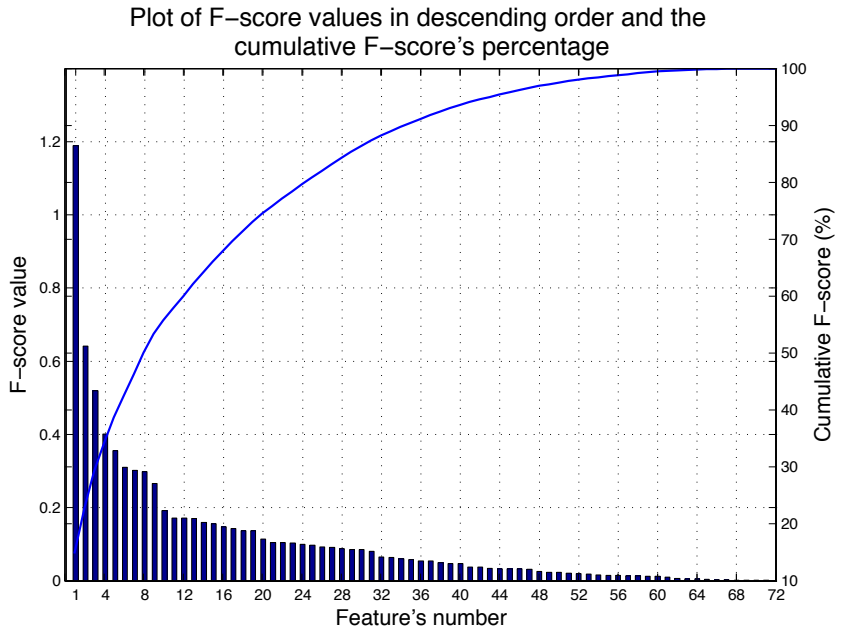


Figure 4.23: F-score values for the HOG features

To evaluate the performance of the above features selection techniques, we created several features subsets using the rankings proposed by both methods. Starting from the full features set, four features from the bottom of the ranking are removed to create a new subset. The same procedure is repeated for the remaining features until there are only twenty features left. This created thirteen features subsets for each of the feature selection technique. Each subset is then trained using the SVM classifier and the performance is evaluated. The results are shown in the ROC curves in Figures 4.25 and 4.26. The cross validation results are given in Table 4.7 and are also plotted in Figure 4.24.

From the results, it was found that all subsets with 32 or less features are non-optimal since their detection rates are significantly lower. Whereas the subsets with 40 or more features show little differences in their detection rates compared to the full feature set (<0.2 %). This means that the 40-features' subset selected by either of the two feature selection techniques is sufficient to represent the full features set. It is also interesting to note that in both the 40-features' subsets from the two feature selection techniques, there are 22 common features that mostly came from the horizontal and vertical orientation's bins. This is as expected since the horizontal and vertical edges are the most prominent visual characteristics of a vehicle.

Table 4.7: Cross validation detection rates for different features subsets

Number of features in the features subset	PCA-select's detection rate	F-Score's detection rate	Processing time (ms)
20 features	95.4%	96.3%	11.31
24 features	96.8%	97.4%	12.03
28 features	97.4%	97.7%	12.27
32 features	97.7%	97.8%	12.97
36 features	98.1%	98.2%	13.51
40 features	98.2%	98.3%	14.00
44 features	98.3%	98.3%	14.69
48 features	98.3%	98.2%	15.41
52 features	98.4%	98.3%	16.26
56 features	98.3%	98.4%	17.69
60 features	98.4%	98.4%	18.65
64 features	98.3%	98.4%	19.86
68 features	98.4%	98.3%	21.35
72 (all) features	98.4%	98.4%	22.45

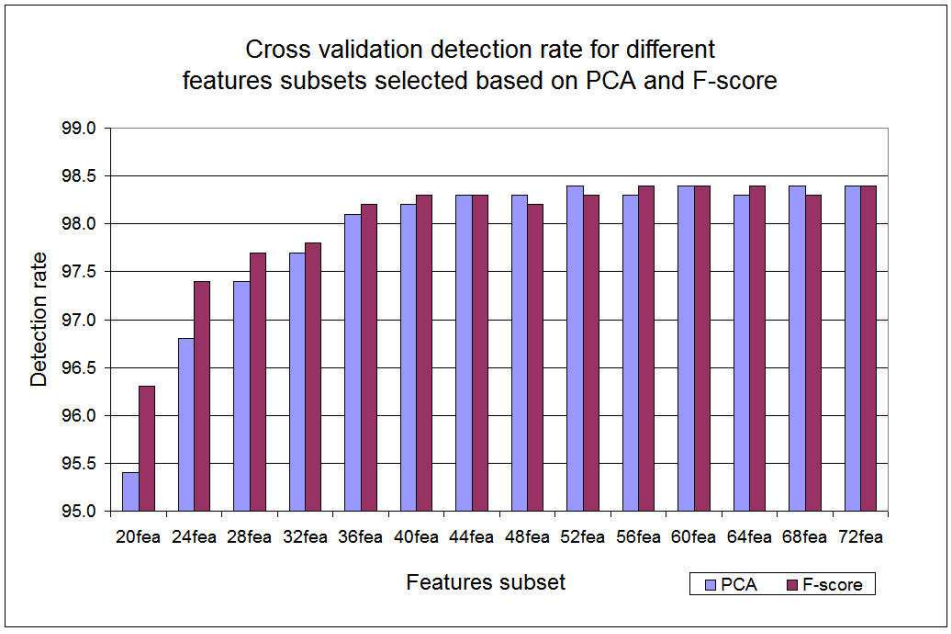


Figure 4.24: Bar chart showing the cross validation detection rates for the reduced feature sets selected by PCA and F-score

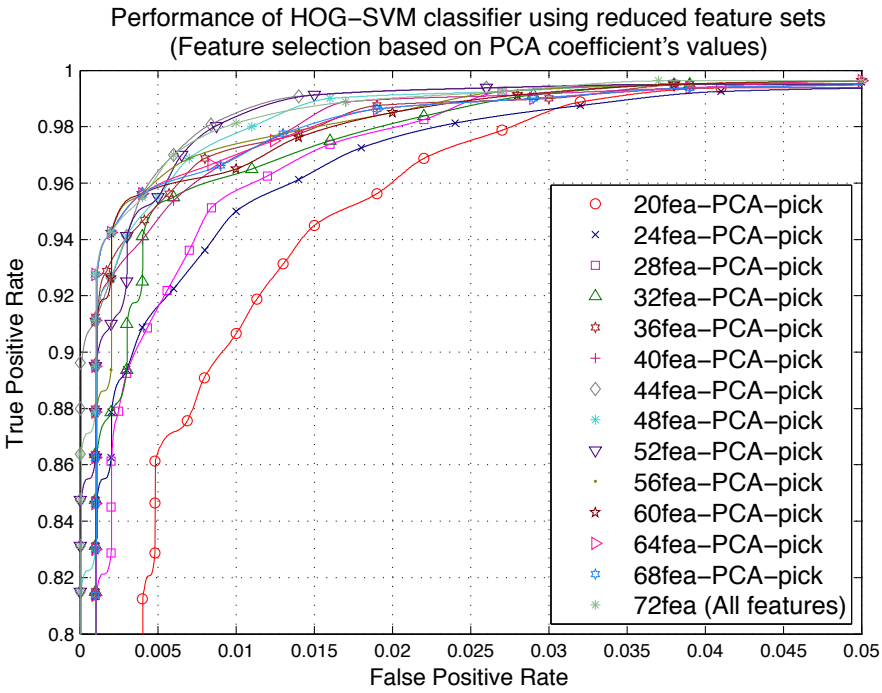


Figure 4.25: ROC curves for the reduced feature sets selected using PCA

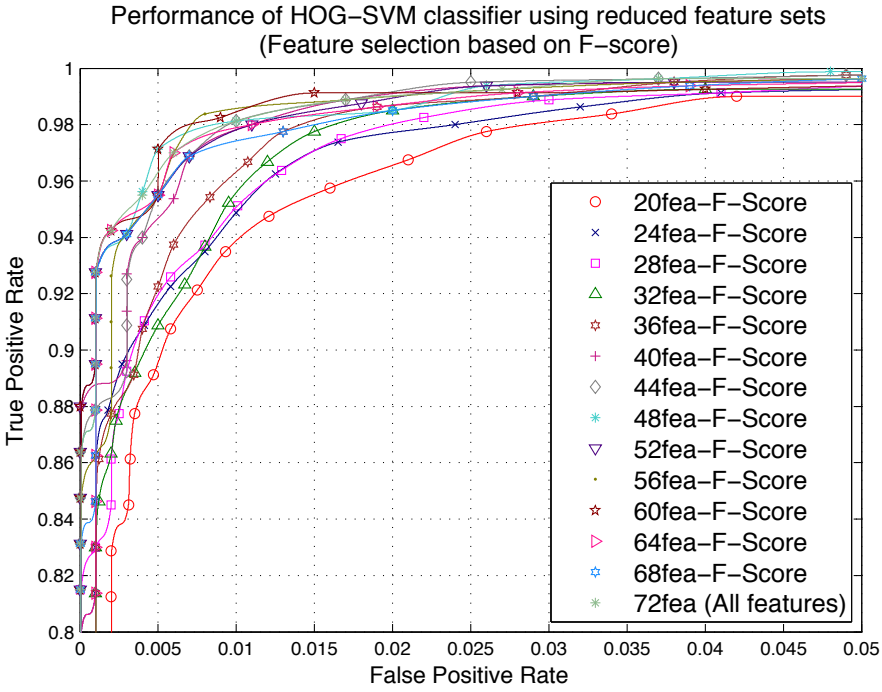


Figure 4.26: ROC curves for the reduced feature sets selected using F-score

From the experiments, it can be observed that the main advantage of using a reduced features set is the saving of processing time. Using either of the 40-features' subsets rather than all the features will reduce the classification time by around 38%. Yet both of them are able to provide comparable classification performance. Out of the two 40-features' subsets selected by the two different feature selection techniques, the F-score's subset was chosen since it has a slightly better performance compared to the PCA's subset. This subset in combination with the SVM classifier will be implemented in the vehicle detection system.

4.6 Summary

This chapter described the development of a two-class vehicle classifier. The purpose is to use the classifier to verify the hypothesised vehicle in the vehicle detection system. In the experiments, the discrimination ability of the HOG and Gabor features for vehicle classification were investigated. For each of these features, a systematic approach was taken to find the best parameters for features extraction. The performance of the features were compared on the SVM, MLP and Mahalanobis distance classifiers.

From the experiment it was found that the HOG feature extracted using 32 orientations grouped into 16 histogram bins produced the best result. Using a higher number of histogram bins will improve the classification performance. However, this comes with the cost of a higher computation time due to the larger feature's size. From the results, it was also observed that using the 1D non-centred technique for gradient calculation outperformed the 1D centred or Sobel mask techniques.

On the whole, the HOG feature performed better than the Gabor feature. It also requires a shorter processing time. Feature extraction for Gabor is slower since it requires the use of a larger convolution mask. It also needs a post processing step to calculate the image moments. On top of that, it generates a higher number of features and thus requires a much longer classification time.

In the experiment, two approaches for features selection have also been investigated. These approaches used either PCA or F-score to rank the features. An experiment was

carried out on the best performing HOG features to evaluate the effect of reducing the number of features. The results showed that for both the feature selection techniques, a subset of 40 highest ranking features is sufficient to represent the full 72 features. The 40 features' subset selected by F-score was chosen to be used in the vehicle detection system since it has a slightly better performance compared to the PCA's subset.

For both the HOG and Gabor features, the SVM classifier achieved better performance compared to the MLP and the Mahalanobis distance classifiers. However, it demands the highest computational cost. The MLP requires a slightly shorter processing time, but its classification performance is poorer compared to the SVM. The Mahalanobis distance classifier is the fastest, however, its performance is significantly inferior. Given the real time constraints for a vehicle detection system, it is critical to choose a fast technique for vehicle verification. Yet, the SVM classifier was chosen to be implemented in the vehicle detection system due to its good classification results and reasonable processing time.

In conclusion, this chapter has proposed a vehicle classifier based on the SVM and the HOG reduced features. The processing time of this classifier is 14 ms which is still acceptable for real time implementation. The proposed classifier will be used to verify the hypothesised vehicles identified in the cueing stage. Once a vehicle is verified, it is added to the tracking list. From there on, a tracking process will monitor the movement of the vehicle in the subsequent image frames. In the next chapter, the development of a vehicle's tracking function based on the Kalman filter and a reliability points system will be discussed.

Chapter 5

Vehicle Tracking Using Kalman Filter

Once a vehicle is verified, the movement of the vehicle in the subsequent video frames will be monitored by a tracking function. In this study, a tracking function based on a Kalman filter and a reliability point system is proposed.

There are several advantages for including tracking in the system. First, by exploiting the temporal coherence of the video frames, the region to search for re-detecting a vehicle can be narrowed. Secondly, the tracking process can monitor the movement of a detected vehicle even though it is momentary not detected in one or several consecutive video frames. This will prevent the sudden 'loss' of a vehicle from the detection output due to occlusion or poor contrast between the vehicle and the road background.

This chapter describes the development of the proposed vehicle tracking function. First, the overall structure of the tracking module is explained. Then the application of a Kalman filter for tracking vehicles in the image plane is described. Next, the integration of a reliability point system to assess the quality of tracking is given in section 5.3. Then the experiments and the evaluation results are presented in the experiment section. This is followed by a summary in the last section.

5.1 The Vehicle Tracking Module

After the vehicles are detected by the verification stage, they will be passed to the tracking function. The tracking function manages all the detected vehicles in a tracking list

and tracks their movement in the subsequent video frames. By having the tracking function monitor the detected vehicles, the relatively time consuming cueing and verification stages can be bypassed for certain video frames to improve the overall speed of the system. This will not cause huge impact to the detection rate since a vehicle usually comes gradually into the scene over several image frames.

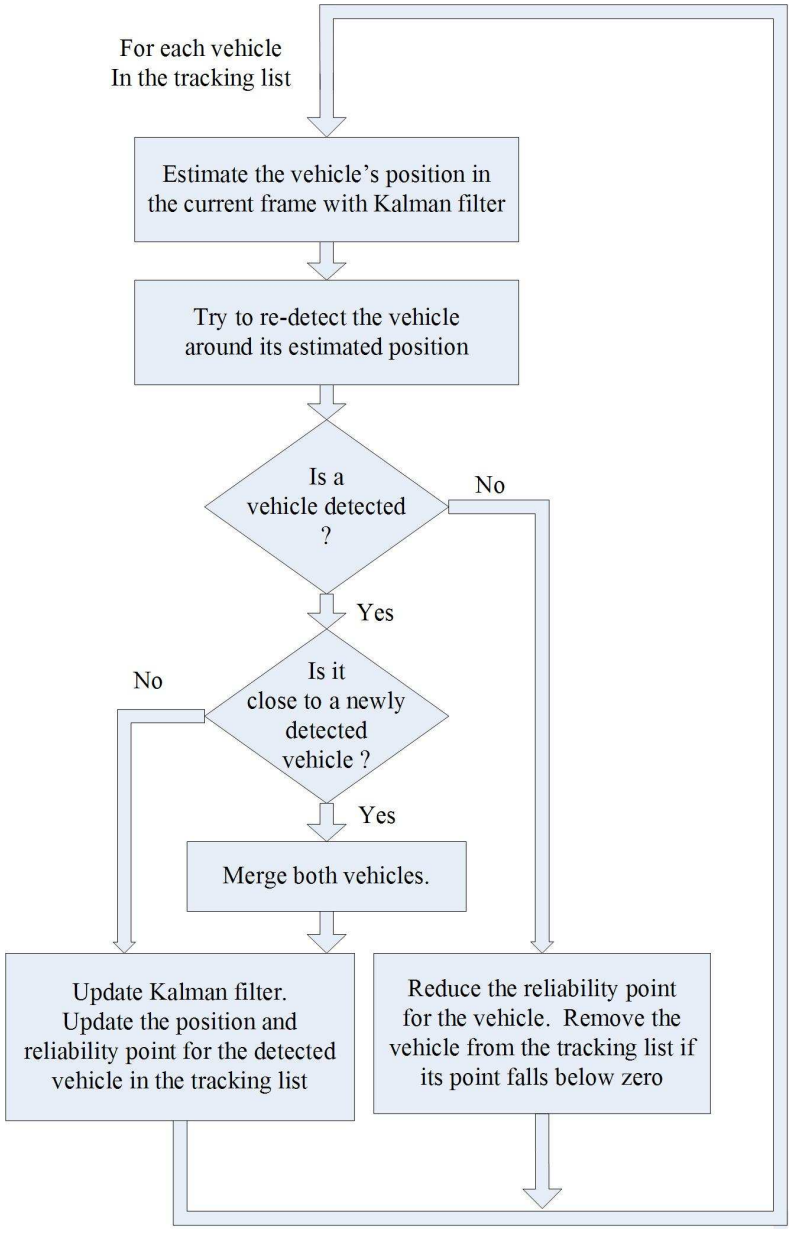


Figure 5.1: Flow diagram of the vehicle tracking process

Figure 5.1 shows the flow diagram of the tracking module. At every tracking cycle, the tracker updates the positions of the vehicles in the tracking list by trying to re-detect

the vehicles around their previous positions. This is assisted by a Kalman filter which predicts the trajectory and the most probable locations of the vehicles in the subsequent video frames. This will narrow down the search area for re-detecting a vehicle

If a vehicle is detected in the search area, the tracking function will check if it is near to any newly detected vehicle. Two nearby vehicles with size and aspect ratio close to each other will be merged and the result updated to the tracking list. Ideally, each tracked vehicle should be re-detected if it does not leave the scene. However, due to occlusion and the variable contrast between the vehicles and the road background, some vehicles are not being re-detected in every frame.

The quality of tracking is assessed using a reliability point system. When a vehicle is first detected, it is assigned with some initial points. During the tracking cycle, points are added if the vehicle can be successfully re-detected. Otherwise, points will be deducted. A vehicle is considered valid if its reliability point is above a certain threshold. However, if the point falls below zero, the vehicle is assumed to be no longer in the scene and will be removed from the tracking list. By incorporating the reliability point system into the tracking, wrong termination of a tracked vehicle due to momentarily unsuccessful detection in one or several frames can be avoided.

5.2 Application of Kalman Filter for Vehicle Tracking

A Kalman filter is integrated into the tracking function to predict the location of a vehicle in the future video frames. There are several advantages for including the Kalman filter into the tracking process:

1. It improves the re-detection rate since the search is carried out at the best estimated location of the vehicle in the next video frame.

2. It reduces the search area for vehicle re-detection and hence shortens the computation time.

3. By discarding other parts of the image in the search, the possible number of false detections can be reduced.

In addition, the smoothing effect of the Kalman filter will refine the tracking result from the uncertainty of the measurement noise and handle the situation of momentarily missed detections.

The following subsections explain the Kalman filtering and its application for tracking vehicles in the image plane.

5.2.1 The Discrete Time Kalman Filter

The process and the measurement models of a linear discrete-time system can be defined by the following equations [157, 158]:

$$\mathbf{x}_k = \mathbf{F}_{k-1}\mathbf{x}_{k-1} + \mathbf{w}_{k-1} \tag{5.1}$$

$$\mathbf{y}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k \tag{5.2}$$

where $\mathbf{x}_k$ and $\mathbf{y}_k$ are the state and measurement vectors at time k. $\mathbf{F}_k$ and $\mathbf{H}_k$ are the transition and measurement matrices.

The process noise, $\mathbf{w}_k$ and the measurement noise, $\mathbf{v}_k$ are assumed to be independent, zero-means, white Gaussian noise with covariance matrices $\mathbf{Q}_k$ and $\mathbf{R}_k$ respectively:

$$\mathbf{w}_k \sim (0, \mathbf{Q}_k) \tag{5.3}$$

$$\mathbf{v}_k \sim (0, \mathbf{R}_k) \tag{5.4}$$

The Kalman filter estimates the state of the process by recursively updating the system dynamics. This is done in two phases– the *time update* phase and the *measurement update* phase [159]. This is illustrated in Figure 5.2.

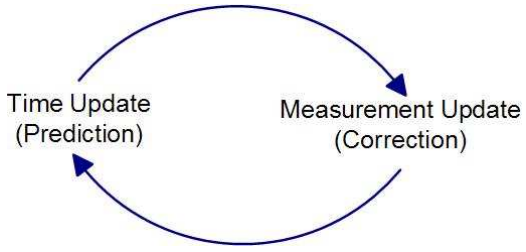


Figure 5.2: The two-phase Kalman filter cycle: The time update predicts ahead in time based on the current state. The measurement update adjusts the prediction with the data from the latest measurement

The *time update* phase projects forward in time the current state and error covariance to obtain the *a priori* estimates for the next time step. This projection does not take into account the latest measurement. The *measurement update* phase incorporates the latest measurement into the system's model to get the *a posteriori* estimates of the state and error covariance.

The *time update* and the *measurement update* algorithms are summarised in equations 5.5 to 5.9. In the equations, the subscript indicates the time step while the superscripts '-' and '+' indicate the *a priori* and the *a posteriori* estimates respectively. $\hat{\mathbf{x}}$ denotes the estimate for $\mathbf{x}$.

The Time Update Equations

The time update equation for calculating the *a priori* state estimate is given by:

$$\hat{\mathbf{x}}_k^- = \mathbf{F}_{k-1}\hat{\mathbf{x}}_{k-1}^+ \tag{5.5}$$

The time update equation for the *a priori* error covariance estimate is:

$$\mathbf{P}_k^- = \mathbf{F}_{k-1}\mathbf{P}_{k-1}^+\mathbf{F}_{k-1}^T \tag{5.6}$$

The Measurement Update Equations

After getting the newest measurement, $\mathbf{y}_k$, the *a posteriori* estimate, $\hat{\mathbf{x}}_k^+$ is calculated using the following measurement update equation:

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{y}_k - \mathbf{H}_k\hat{\mathbf{x}}_k^-) \tag{5.7}$$

The *a posteriori* error covariance is updated using this measurement update equation:

$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^- \tag{5.8}$$

where $\mathbf{K}_k$ is called the Kalman filter gain. It is given by the following equation:

$$\mathbf{K}_k = \frac{\mathbf{P}_k^-\mathbf{H}_k^T}{\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T + \mathbf{R}_k} \tag{5.9}$$

The cycle for the time and measurement updates repeats for every time step. The goal of the Kalman filter is to provide the best estimate for the state of the system based on the current available knowledge in the system's model and the measurement data.

5.2.2 Implementation of the Kalman Filter for Vehicle Tracking

In this study, a Kalman filter is used to predict the motion and the changes in image size of a vehicle in the image plane. At video frame rate (>10 fps), the movement of a vehicle's image between two consecutive video frames is small. Therefore, it is sufficient to model such movement as constant velocity [158]. These assumptions allow the system to be modelled using the linear discrete-time Kalman filter described in the previous section. The linear Kalman filter is simpler and not as computationally costly compared to the non-linear models such as the Extended Kalman filter or Particle filter.

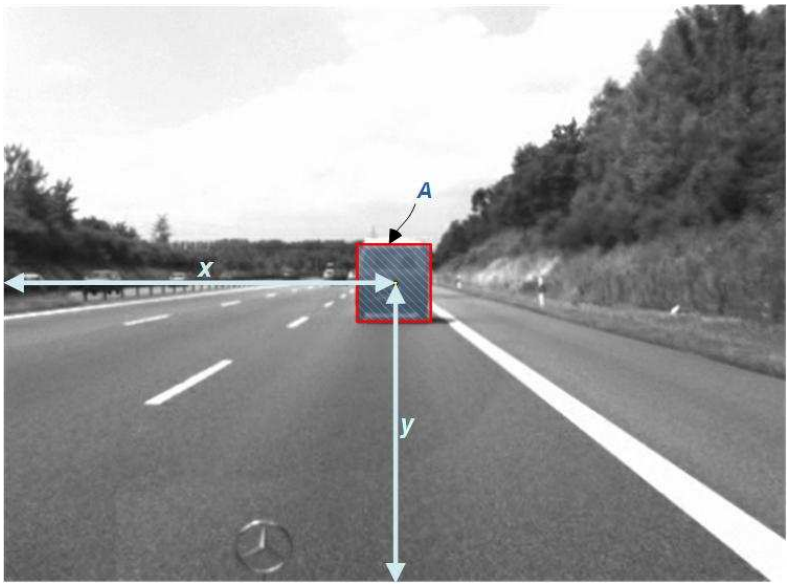


Figure 5.3: Figure shows the parameters used in the Kalman filter. They are the coordinates of the vehicle's centre point (x, y) and the area of the vehicle's image (A)

The variables that are integrated into the Kalman filter are the centre point, (x, y) and area, A of the tracked vehicle in the image plane (Figure 5.3). This has resulted in the following state and measurement vectors:

$$\mathbf{x}_k = [x, y, A, v_x, v_y, v_A]^T \tag{5.10}$$

$$\mathbf{y}_k = [x, y, A]^T \tag{5.11}$$

where v_x and v_y are the velocities in the movement of the vehicle's centre point in the x and y directions. v_A is the rate of change in the vehicle's image size.

With these state and measurement matrices, the Kalman transition matrix, $\mathbf{F}_k$ and measurement matrix, $\mathbf{H}_k$ can be constructed as follows:

$$\mathbf{F}_k = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.12}$$

$$\mathbf{H}_t = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{5.13}$$

where Δt is the time difference between the current and the last video frame.

The covariance matrices for the process and measurement errors ($\mathbf{Q}_k$ and $\mathbf{R}_k$) are assumed to be constant over time. This study follows a technique described in [158] to initialise their values. The technique was developed to model any translational motion with constant velocity and arbitrary acceleration in the image plane. Based on this technique, the initialisation values for $\mathbf{Q}_k$ and $\mathbf{R}_k$ are given as:

$$\mathbf{Q}_k = \begin{bmatrix} \frac{a_{e,x}^2 \Delta t^3}{3} & 0 & 0 & \frac{a_{e,x}^2 \Delta t^2}{2} & 0 & 0 \\ 0 & \frac{a_{e,y}^2 \Delta t^3}{3} & 0 & 0 & \frac{a_{e,y}^2 \Delta t^2}{2} & 0 \\ 0 & 0 & \frac{a_{e,A}^2 \Delta t^3}{3} & 0 & 0 & \frac{a_{e,A}^2 \Delta t^2}{2} \\ \frac{a_{e,x}^2 \Delta t^2}{2} & 0 & 0 & a_{e,x}^2 \Delta t & 0 & 0 \\ 0 & \frac{a_{e,y}^2 \Delta t^2}{2} & 0 & 0 & a_{e,y}^2 \Delta t & 0 \\ 0 & 0 & \frac{a_{e,A}^2 \Delta t^2}{2} & 0 & 0 & a_{e,A}^2 \Delta t \end{bmatrix} \tag{5.14}$$

$$\mathbf{R}_k = \begin{bmatrix} \sigma_{e,x}^2 & 0 & 0 \\ 0 & \sigma_{e,y}^2 & 0 \\ 0 & 0 & \sigma_{e,A}^2 \end{bmatrix} \tag{5.15}$$

where $a_{e,x}$, $a_{e,y}$ and $a_{e,A}$ are the spectral amplitudes of white noise and $\sigma_{e,x}^2$, $\sigma_{e,y}^2$ and $\sigma_{e,A}^2$ are the variances of the measurement errors (for x, y and area respectively).

The Kalman filter models the state's probability distribution as Gaussian. This means that each Kalman filter can only track a single vehicle. To overcome this limitation for tracking multiple vehicles, a separate Kalman filter is instantiated for each vehicle in the tracking list.

The *a priori* estimate of the Kalman filter suggests the location (vehicle's centre point) and size of the region where a vehicle could possibly appear in the video frame. This information is utilised by the tracking function to narrow down the search space for re-detecting a vehicle. Once the re-detection is accomplished, the new measurement data will be reconciled into the system's model. The *a posteriori* estimate is then calculated and used as the best estimate for the vehicle's location and size in the current video frame.

5.3 The Reliability Point System

A reliability point system is used to assess the quality of the tracking. The assessment is based on a simple rule-based technique.

When a vehicle is initially detected, it is assigned with some reliability points. The vehicle is then tracked over a sequence of video frames, during which time it may accumulate a number of reliability points. Points are added or deducted depending on how consistent a vehicle can be re-detected. The following are the rules used by the tracking function to update the reliability point of a vehicle:

1. When a vehicle is first detected, two reliability points are assigned to the vehicle;
2. If the vehicle is not re-detected in the following frame, one point is deducted;
3. If the vehicle is re-detected in the following frame, one to three points may be added. The number of points depends on how consistent the size and aspect ratio between the current and the last detection. However, the maximum of points a vehicle can accumulate is six;
4. If the reliability points of a vehicle are more than two, the vehicle is considered to be valid and will be displayed at the tracking output;

5. If the reliability point of a vehicle falls below zero, the vehicle will be removed from the tracking list.

Based on these rules, a newly detected vehicle has to be tracked for at least one cycle before it is displayed on the output. This is to prevent the display of false detections which usually appear and disappear in short intervals. On the other hand, if a vehicle has attained more than three points, it will keep on being displayed even though it is not being re-detected in the following video frame. This will prevent the 'disappearance' of a valid detection from the display due to momentarily missed detection. In a situation where a vehicle is not being re-detected, the estimates from the Kalman filter will be used to update the vehicle's status during the tracking cycle.

New Detected Vehicles

Once a new vehicle is detected and verified, its descriptors are passed to the tracking function, which are the vehicle's centre point, aspect ratio and size. The tracking function will then take the following steps to update the vehicles to the tracking list:

1. Calculate the distance, d of a vehicle in the tracking list to every new detected vehicle. The distances are calculated using the following equation, where (x_1, y_1) and (x_2, y_2) are the centre points of two vehicles:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{5.16}$$

2. Find the vehicle with the shortest distance.

3. If the distance is less than a predefined threshold θ_d, and the differences in size and aspect ratio of both vehicles are less than 10%, then merge both vehicles and update the result to the tracking list. The value of θ_d was empirically determined during the experiment. It was found that a value of 20 pixels can provide satisfactory results.

4. Repeat steps 1–3 for all the remaining vehicles in the tracking list.

5. Any new detected vehicles that have not been merged are added as new entries in the tracking list.

5.4 Experiments and Results

Two experiments were carried out to test the efficiency of the vehicle tracker. The first experiment investigated the tracking of a preceding vehicle from a following car. The second experiment tested the tracking of an overtaking vehicle. Both experiments used the pre-recorded video streams taken from a forward looking camera installed behind the windscreen of a test vehicle. In the test, the coordinate of the vehicle's centre point (x, y) and the size of the vehicle's image (A) were monitored over the consecutive video frames. Both the measured and the Kalman estimated values were recorded and compared. The purpose is to inspect the measurement error and to see how well the Kalman filter can regulate those errors.

5.4.1 Tracking of a Preceding Vehicle

In this experiment, a preceding vehicle was tracked over 500 video frames. The test video was taken by a host vehicle that followed closely behind another vehicle along a highway. At around frame 420, the preceding vehicle started to steer to the right lane. The results are plotted in Figure 5.4 to 5.6.

In the figures, the dotted lines represent the measured values while the darker lines are for the values of the Kalman filter's estimates. It can be seen that there are some random fluctuations in the measured values. This is mainly caused by the error in the bounding box detection. However, less fluctuations were observed in the Kalman estimated values. This can be seen in the figures where the plots for the Kalman filter's estimates are smoother. The results show that the Kalman filter can provide a good estimate for the position and size of the vehicle. Using these values to decide on the region for finding a vehicle in the subsequent video frame may improve the re-detection rate.

In Figure 5.4, the x-coordinate of the tracked vehicle increased substantially after frame 420. This happened during the preceding vehicle steered to the right lane. The size of the vehicle's image can provide some clues about the vehicle's distance. From the plot in Figure 5.6, it can be observed that the preceding vehicle was gradually moving away from the test vehicle but it got closer again when it started to change lane.

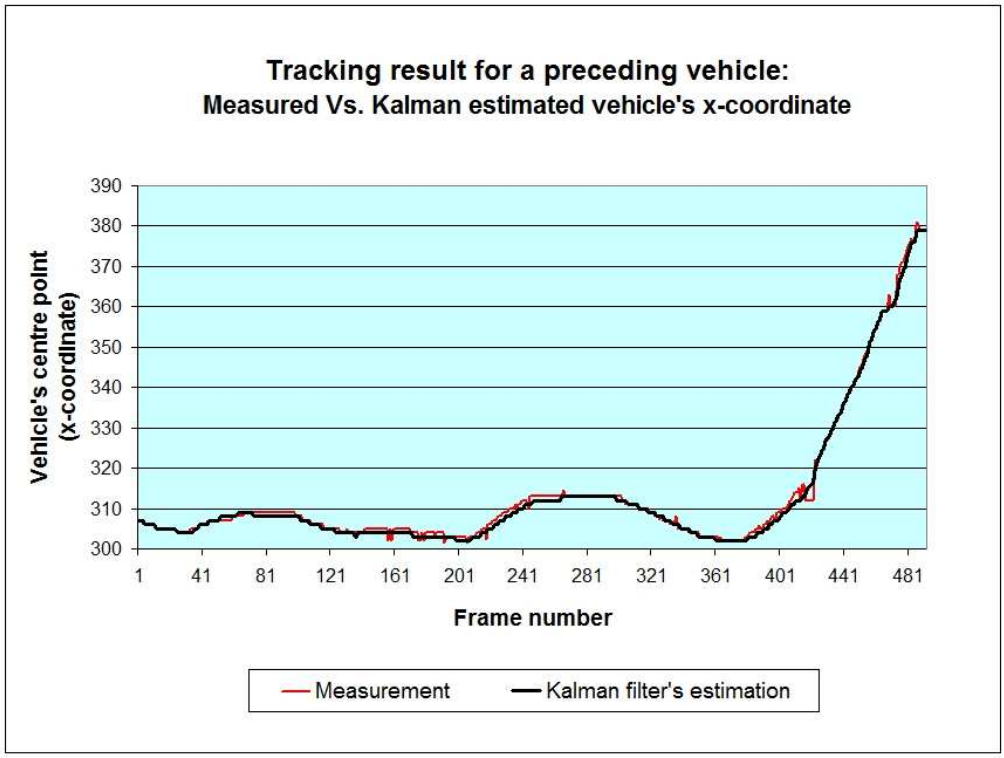


Figure 5.4: Preceding vehicle tracking: The measured and the Kalman's estimated values for the vehicle's x-coordinate

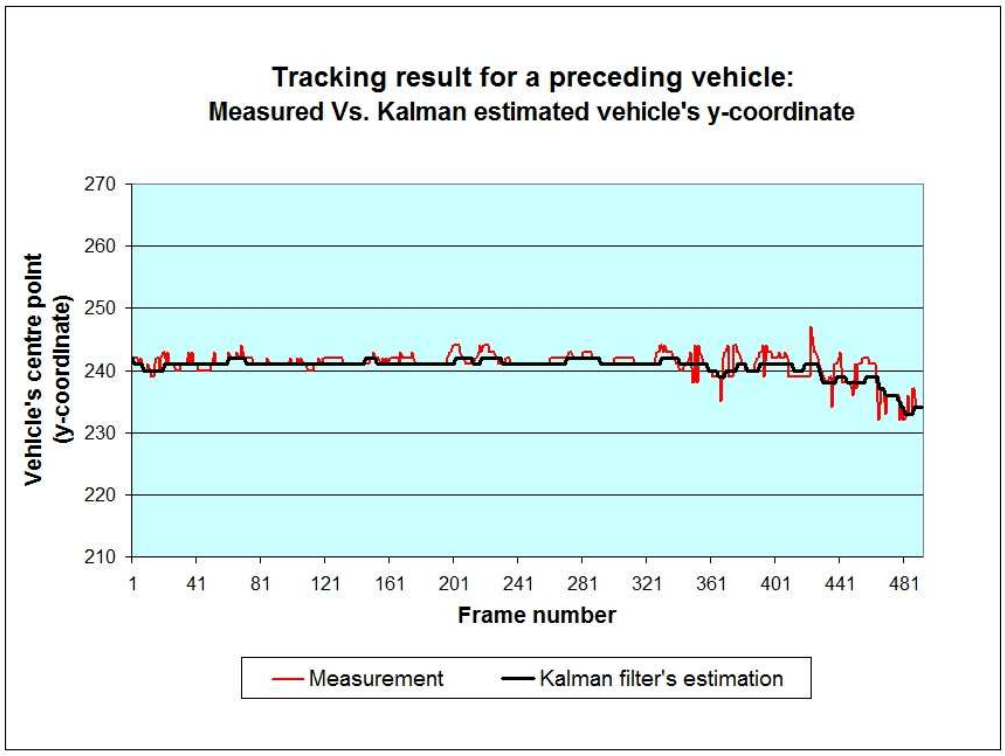


Figure 5.5: Preceding vehicle tracking: The measured and the Kalman's estimated values for the vehicle's y-coordinate

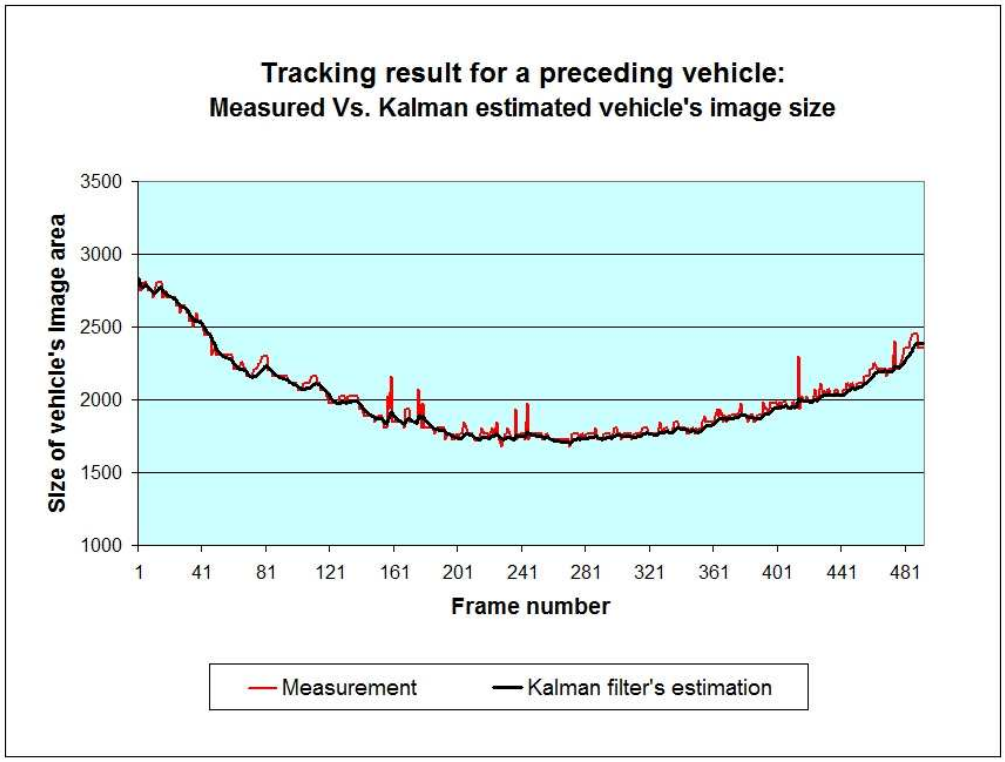


Figure 5.6: Preceding vehicle tracking: The measured and the Kalman's estimated values for the vehicle's image size

5.4.2 Tracking of an Overtaking Vehicle

The test video used in this experiment shows a vehicle overtaking the host vehicle from the left lane. The system tracked the overtaking vehicle in 85 video frames. The results are plotted in Figures 5.7 to 5.9.

Similar to the first experiment, there were random fluctuations in the measured values. In addition, this test also has recorded two instances of missed detection which happened during frames 8 and 47. These can be seen in the figures where there are two gaps in the measurements plots. However, despite these gaps, the graphs for the Kalman estimates still remained smooth throughout the whole test. The tracking process checks and updates the reliability points of the 'missing' vehicle and keeps tracking it using the Kalman's estimated values.

This result shows that a combination of the Kalman filter and the reliability point system can assure the continuous tracking of a vehicle even though there are some momentarily missed detections.

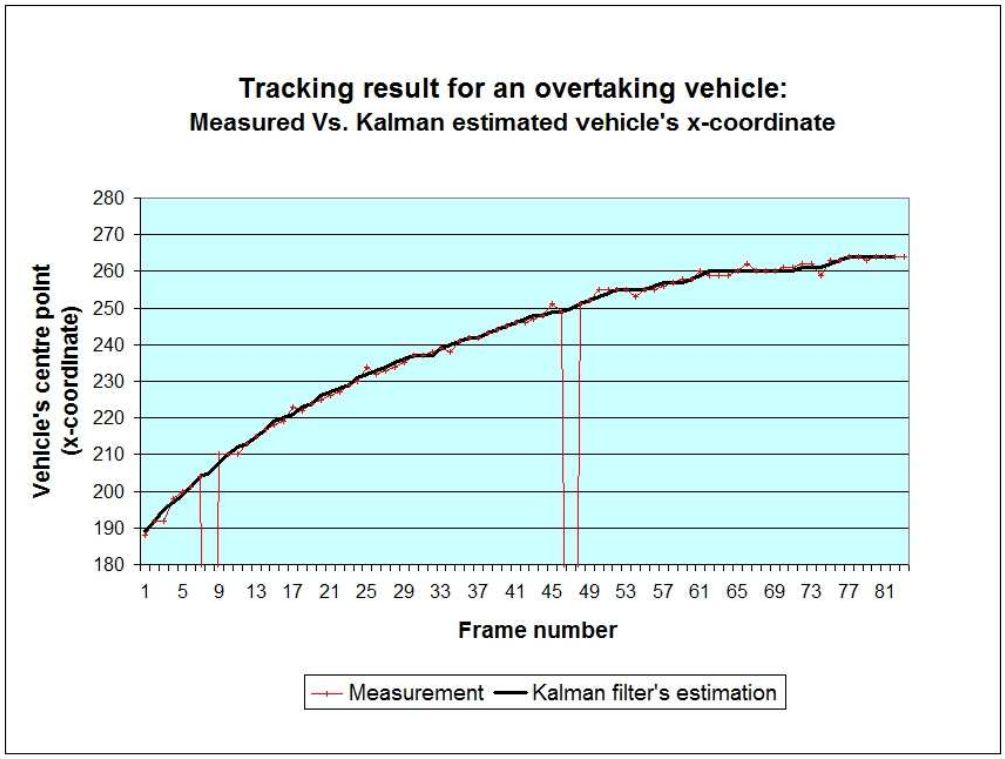


Figure 5.7: Tracking result for an overtaking vehicle: The measured and the Kalman's estimated values for the vehicle's x-coordinate

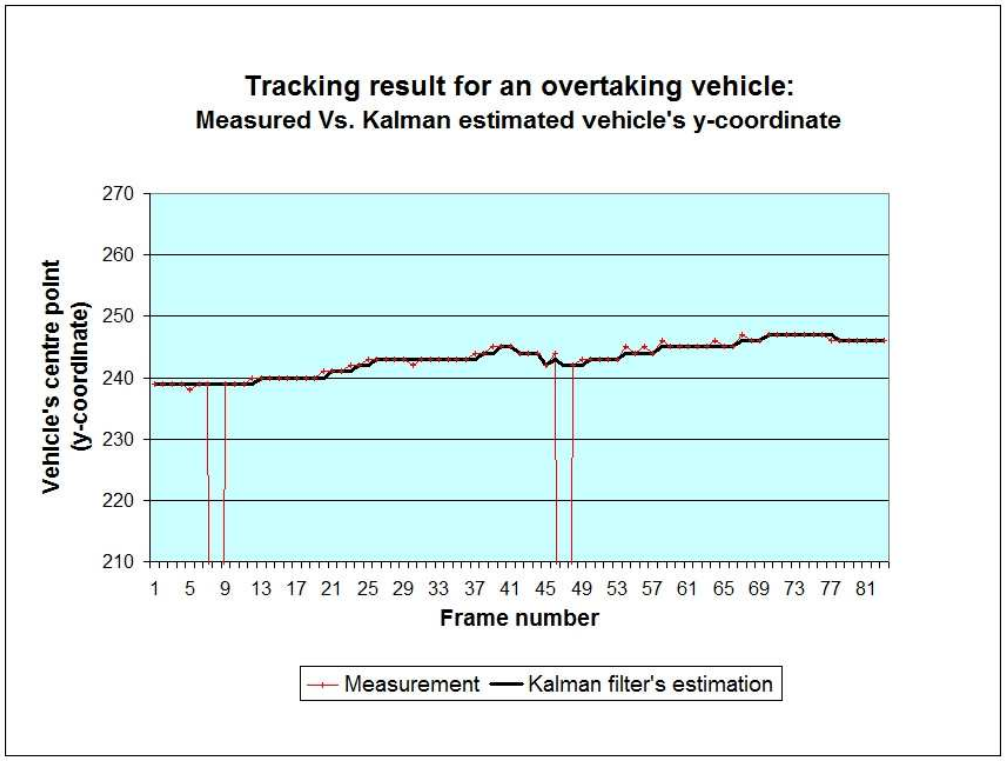


Figure 5.8: Tracking result for an overtaking vehicle: The measured and the Kalman's estimated values for the vehicle's y-coordinate

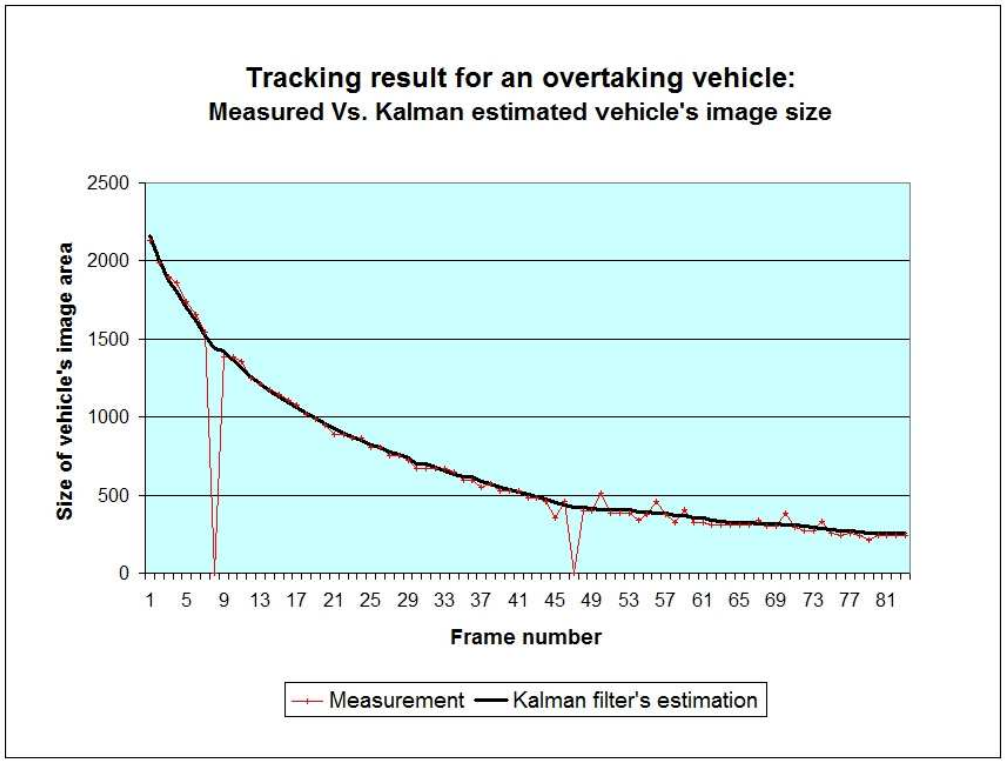


Figure 5.9: Tracking result for an overtaking vehicle: The measured and the Kalman's estimated values for the vehicle's image size

5.5 Summary

This chapter has described the development of a tracking function for the vehicle detection system. The function uses a Kalman filter and a reliability point system to improve the efficiency of tracking. The Kalman filter is used to smoothen the irregularity due to the error in the detection. It also predicts the most probable location and size of a tracked vehicle in the subsequent video frame. This information is useful for the tracking function to re-detect the vehicles. By narrowing down the search area, the re-detection rate can be improved while the processing time is reduced.

The reliability point system provides a simple and fast mechanism to monitor the quality of tracking for the vehicles in the tracking list. A detected vehicle has to acquire a certain reliability point before it is considered a valid detection. Also, a vehicle that has accumulated a high reliability point will not be suddenly removed due to momentarily missed detection. For these reasons, incorporating the reliability point system into the tracking function will improve the overall robustness of the vehicle detection system.

The experiment results have shown that the proposed tracking function can successfully track the preceding and the overtaking vehicles in consecutive video frames. The problem of momentarily missed or wrong detection can also be handled.

The tracking function developed in this chapter forms the third module of the proposed vehicle detection system. In the next chapter, the integration of all the three modules, namely, cueing, verification and tracking into the complete system will be discussed.

Chapter 6

System Integration, Experiments and Results

The previous three chapters described the development of the three main modules for the proposed vehicle detection system. The integration of these modules to form the complete system is explained in this chapter. The performance of the system was tested under different road conditions and the results are presented in section 6.2.

6.1 The Proposed Vehicle Detection System

The flow diagram of the proposed vehicle detection system is shown in Figure 6.1. The complete system was formed by the integration of the three core modules developed in the previous three chapters. The first module detects the symmetric regions in the image and uses them as the hypothesised vehicles' locations. An ROI is then defined around the centre point of each region where a possible vehicle's bounding box is searched. If a valid bounding box is detected, it will be verified by the classifier module to determine whether it belongs to a vehicle.

It is essential to keep the edges and the silhouette of the vehicle for the verification process. Therefore, the region of the bounding box for verification is enlarged by a few pixels in all directions. In order to compensate for possible errors in the bounding box detection, the verification is done by scanning the enlarged region at different locations

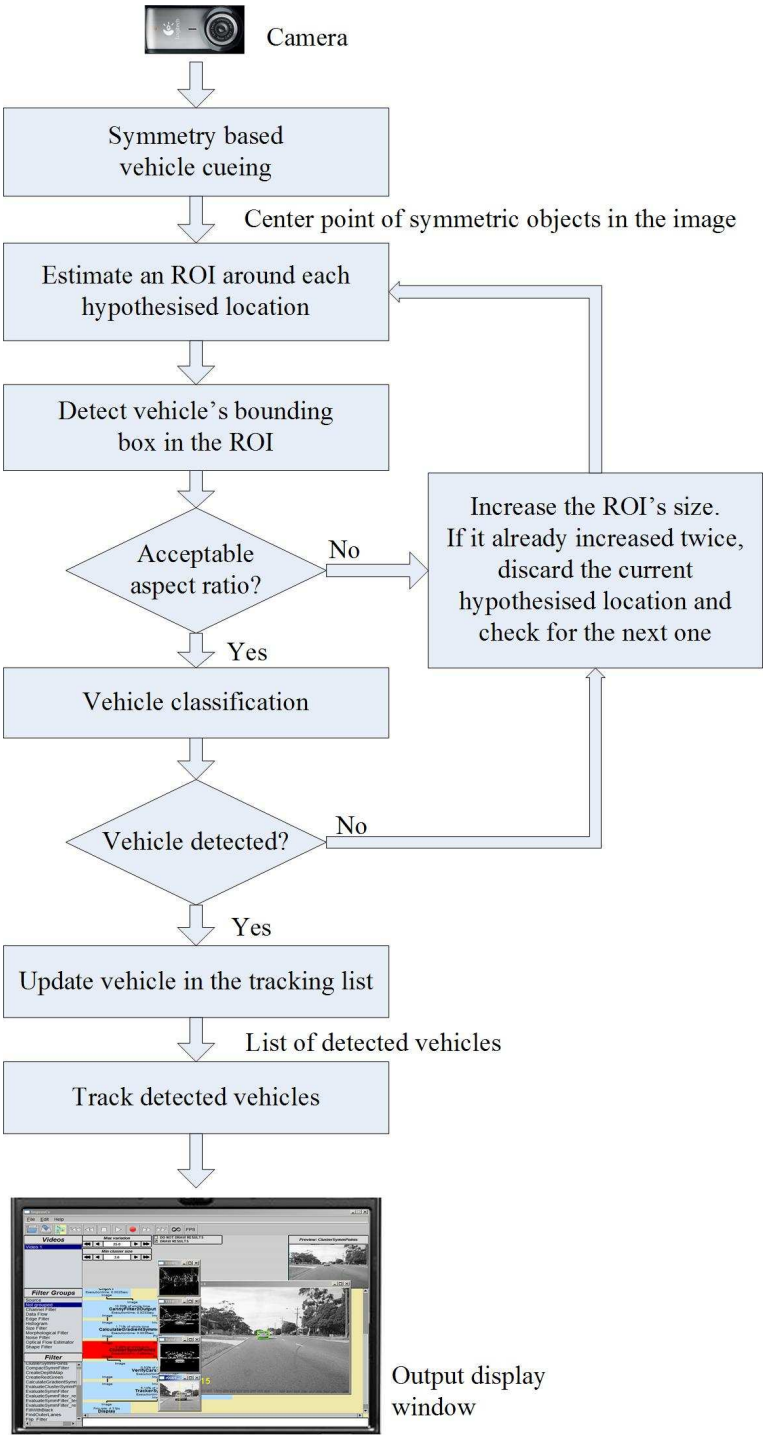


Figure 6.1: The flow diagram of the proposed vehicle detection system

and scales. Due to the high computational requirement of the classification process, only a coarse scanning with an average of three verifications per detection window is carried out.

The selection of a proper operating point for the classifier in the verification module

is critical, since it will determine the sensitivity of the system. For the experiments in this chapter, the best operating point is empirically selected during the evaluation phase. The goal of the selection is to find a point with the best trade off between the number of missed detections and the number of false detections. Increasing the sensitivity will reduce the number of missed vehicles, but at the same time it will generate a lot of false alarms. On the other hand, if the sensitivity is set to too low, the system might miss out some critical vehicles.

If a vehicle is detected, it will be passed to the tracking module. Otherwise, the ROI will be enlarged by 10% and the same process for bounding box detection and vehicle verification is repeated. However, this is allowed to continue for two iterations before the hypothesised location is considered a non-vehicle and consequently discarded.

When the tracking module receives a detected vehicle, it will either merge the vehicle with one in the tracking list or initiate a new tracking instance. It then monitors the movement of the tracked vehicles in the subsequent video frames and draws the results on the output display window.

A vehicle usually comes gradually into the scene over several image frames. Therefore, it is acceptable to regularly skip the comparatively time consuming cueing and verification stages to reduce the overall processing time of the system. This is illustrated in Figure 6.2. Different numbers of frames to skip were tested during the experiment. It was found that periodically skipping by three frames can significantly speed up the system without affecting the detection rate.

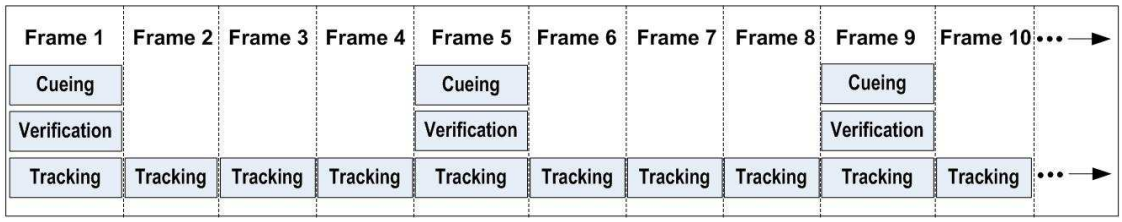


Figure 6.2: Regularly skipping the cueing and verification stages to reduce the overall processing time of the vehicle detection system

6.1.1 Software Modules

The proposed system was implemented on the ImprovCV [106, 105] image processing framework developed in the University of Western Australia, Computational Intelligence Information Processing Systems (CIIPS) research group. Several publicly available software libraries were also used in the implementation. They are the OpenCV [107] image processing library and the libSVM [156] Support Vector Machine library.

The software was written in C/C++ programming language. It was implemented as several add-on modules to the ImprovCV framework (see Figure 6.3). The modules were implemented based on the proposed algorithms from the previous three chapters. They were individually tested before integrating into the complete vehicle detection system.

The training of the classifier used in the verification module was done offline. Based on the evaluation in Chapter 4, the classifier model with the best classification result was used in the implementation.

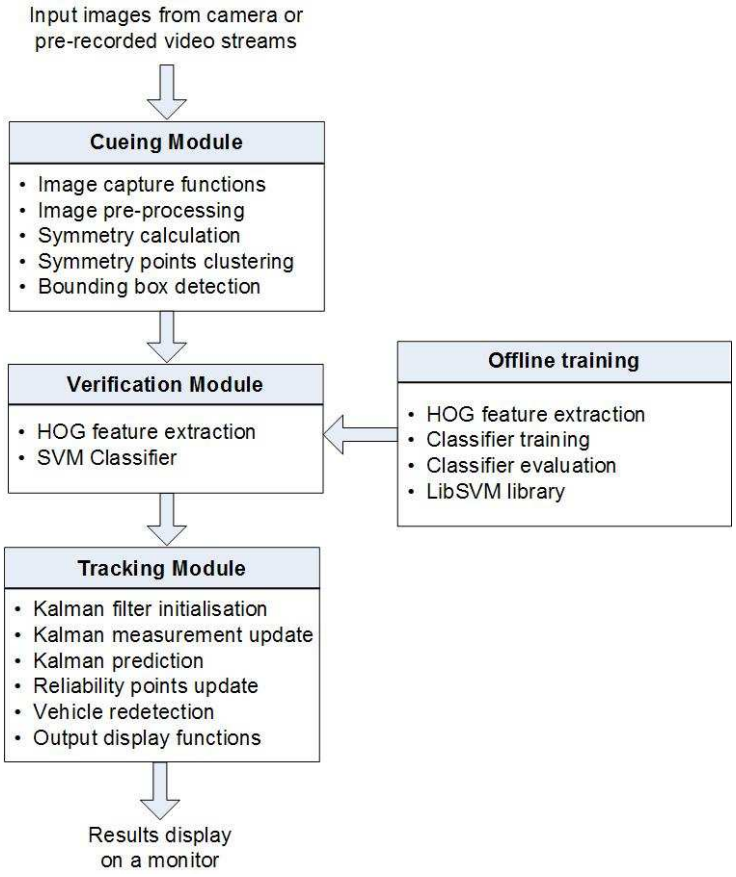


Figure 6.3: The main software modules for the vehicle detection system

6.1.2 Hardware Components

The main hardware for the vehicle detection system consists of a Logitech C905 USB camera and a Dell 6400 laptop computer. The camera is fixed to the windscreen using a suction cup. The image resolution of the camera was set to VGA (640 × 480 pixels). This resolution is sufficient to detect the preceding vehicles at distances less than 50m, which are the critical distances for collision avoidance.

The laptop computer was running at 2.0 GHz on an Intel Core Duo processor with 2GB RAM. Figure 6.4 shows the setup of the vehicle detection system in a BMW X5 vehicle. The software has the option of either getting the input images from the camera or a pre-recorded video stream.

Figure 6.4: Figure shows the setup for testing the vehicle detection system in a BMW X5 vehicle

6.2 Experiments and Results

6.2.1 Experiment Setup

The performance of the system was evaluated on several pre-recorded video streams. Pre-recorded videos were used instead of live camera feeds in order to allow careful analysis of the results in the laboratory. The accuracy of the system can be affected by different road environments. Therefore, several test videos taken on different roads and weather conditions were used in the evaluation.

The outputs of the vehicle detection system were manually analysed frame by frame to calculate the detection rates. To determine whether a vehicle is correctly detected, the bounding box predicted by the system is compared with the area covered by the vehicle in the image. A vehicle is considered detected if the area of the bounding box covers more than 80% of the vehicle's image and the difference in size between the detected bounding box and the vehicle's image is less than 10%. This is given by equation 6.1 and 6.2 where A_V and A_D are the areas of the vehicle's image and the detected bounding box respectively.

$$\frac{A_V \cap A_D}{A_V} > 0.8 \tag{6.1}$$

$$\frac{|A_V - A_D|}{A_V} < 0.1 \tag{6.2}$$

The performance of the system is evaluated using the following three measures: True Positive Rate (TPR), False Positive Rate (FPR) and False Negative Rate (FNR). They are calculated using the following equations:

$$TPR = \frac{Number\ of\ vehicles\ detected}{Number\ of\ vehicles\ appearing\ in\ the\ video\ frames} \times 100\% \tag{6.3}$$

$$FPR = \frac{Number\ of\ false\ detections}{Number\ of\ vehicles\ detected\ +\ Number\ of\ false\ detections} \times 100\% \tag{6.4}$$

$$FNR = \frac{Number\ of\ vehicles\ missed}{Number\ of\ vehicles\ appearing\ in\ the\ video\ frames} \times 100\% \tag{6.5}$$

$$= (100 - TPR)\ \% \tag{6.6}$$

6.2.2 Results and Discussions

Figure 6.5 shows two examples of the output screen on the ImprovCV framework. The final output and the intermediate processing results can be displayed to analyse the operation of the image processing algorithms at different stages. However, for the timing measurement, all the intermediate displays are turned off to get the best speed of the system.

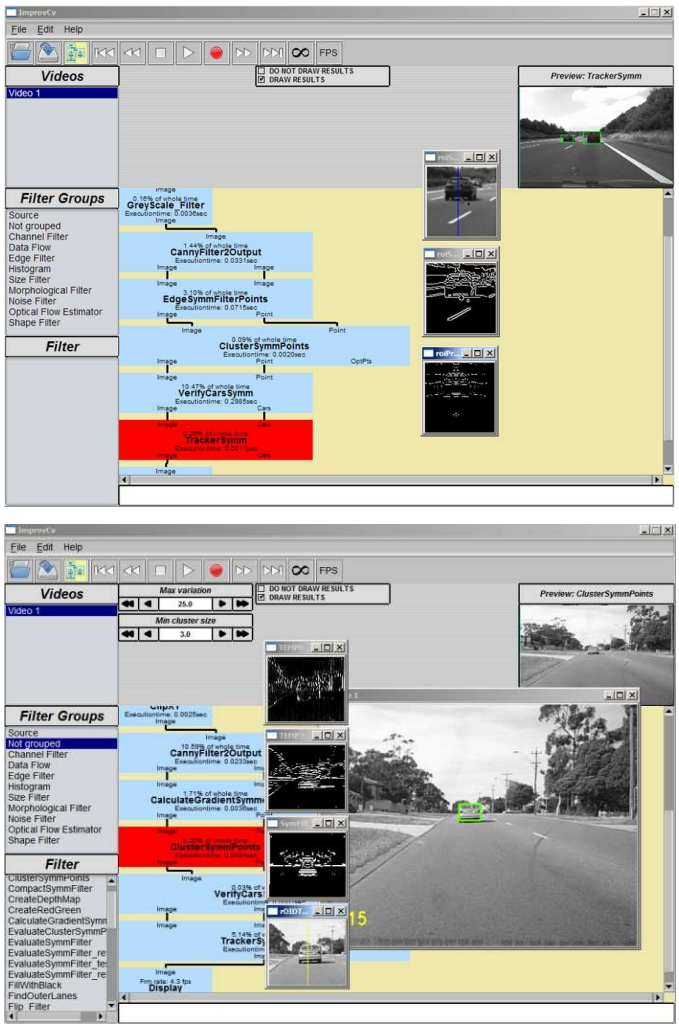


Figure 6.5: The ImprovCV image processing framework. Figures show two examples of the test results with some intermediate outputs displayed in the smaller windows

The Processing Time

Table 6.1 shows the processing time of the vehicle detection system. The system was tested on the 2.0 GHz Intel Core Duo laptop without any specific software optimization.

The verification stage requires the highest processing time since it uses a more sophisticated algorithms. This is followed by the symmetry detection stage but it only needs half of the time required by the verification stage. The overall average processing time for one frame is 80ms. Therefore, the system can achieve a frame rate of approximately 12.5 frames per second.

Table 6.1: Processing time of the vehicle detection system

	Average processing time (ms)
Edge extraction	8
Symmetry detection	22
Vehicle verification	40
Vehicle tracking	10
Total	80

The Detection Rate

The detection rate of the system was evaluated using four video streams. Two video streams were taken on a sunny day, one on a cloudy day and the other in the evening. One of the sunny day's videos was captured on a highway while the rest were taken on different urban roads. All videos have a resolution of 640×480 pixels (VGA) and their length varies from 700 to 900 frames. The results are presented in Table 6.2.

Table 6.2: The test results for the vehicle detection system

	Sunny day (Highway)	**Sunny day (Urban rd)**	**Cloudy day (Urban rd)**	**Evening (Urban rd)**
Number of vehicles appearing in the video frames	864	703	983	715
True Positive Rate (vehicles detected)	94.6%(817)	94.0%(661)	92.3%(907)	90.2%(645)
False Positive Rate (non-vehicles detected)	0.1%(1)	3.4%(25)	3.2%(33)	4.8%(36)
False Negatives Rate (vehicles missed)	5.4%(47)	6.0%(42)	7.7%(76)	9.8%(70)

Both the sunny day's videos recorded a high True Positive rate (94.4% and 94.0% respectively). This is followed by the cloudy day's video (92.3%) and the evening video

(90.2%). One explanation for this is because the videos, taken at different times of day and weather conditions, show different brightness contrast between the vehicles' images and their backgrounds. The image of a vehicle taken on a sunny day are clearer with visible edges and internal structures such as the bumper, rear windscreen and tail lights. These features are important for the bounding box detection and verification.

The results also show that the video taken on the highway has a lower False Positive rate (0.1%) compared to the three urban road's videos (3.2% – 4.8%). This is because the background of the urban road scenes is usually cluttered with objects such as buildings, road signs and bill boards. Some of these objects have aspect ratio and internal structures resembling a vehicle which may confuse the system and mistaken them as vehicles.

(a) (b)

Figure 6.6: Examples of complex road scenes. The white dots are the hypothesised vehicle locations. Figure (b) also shows the symmetry peak points (the smaller dots).

However, most of the non-vehicle symmetrical objects and edges picked up in the cueing stage can be successfully removed by the vehicle verifier. Some examples are shown in Figure 6.6 (a) and (b). The symmetry detector picked up some of the shadows and road edges as symmetric regions (white dots in the images). However, these regions were subsequently removed in the verification stage.

Figures 6.7 to 6.9 show a sequence of images from the output of the vehicle detection system.

(a) (b)

(c) (d)

(e) (f)

Figure 6.7: Some detection results for the highway scenes

(a) (b)

(c) (d)

(e) (f)

Figure 6.8: Some detection results for the urban road scenes

(a) (b)

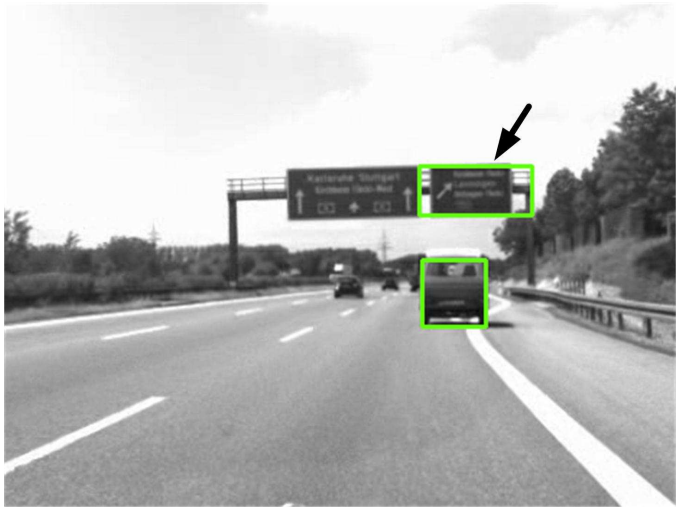
(c) (d)

(e) (f)

Figure 6.9: More detection results for the urban road scenes

The False Detections

Figure 6.10 (a) and (b) shows two examples of false detections. The sign boards (indicated by the arrows) have been falsely detected due to their rectangular shape with aspect ratio close to a vehicle.

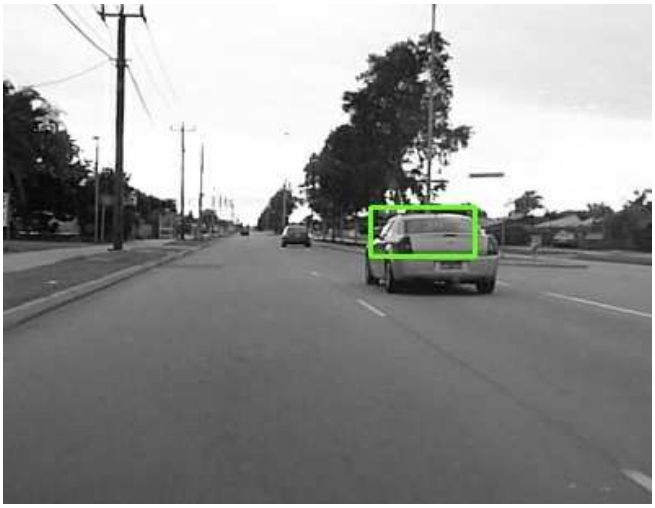


(a)

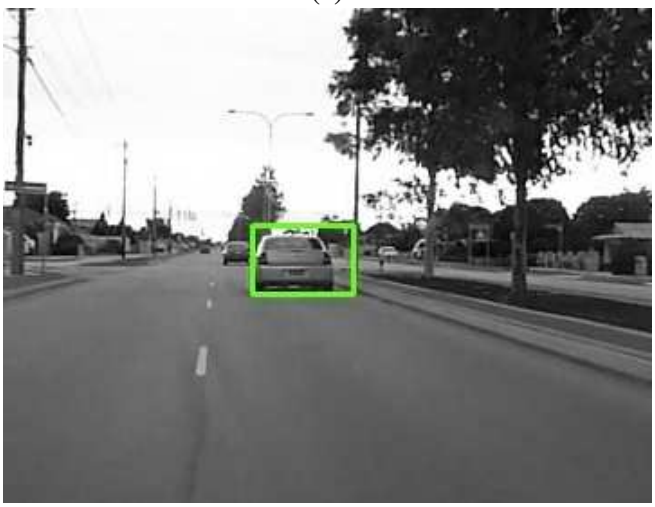
(b)

Figure 6.10: Examples of false positives (wrong detections) indicated by the arrows.

Two examples of wrong bounding box detections are shown in Figure 6.11 (a) and (b). In the first figure, the detected bounding box did not cover the whole vehicle. This is because the edges generated by the vehicle's bumper is stronger compared to the edges of the vehicle's bottom. The second figure shows that the system falsely detected a distant lamp post as the right boundary of the vehicle.



(a)

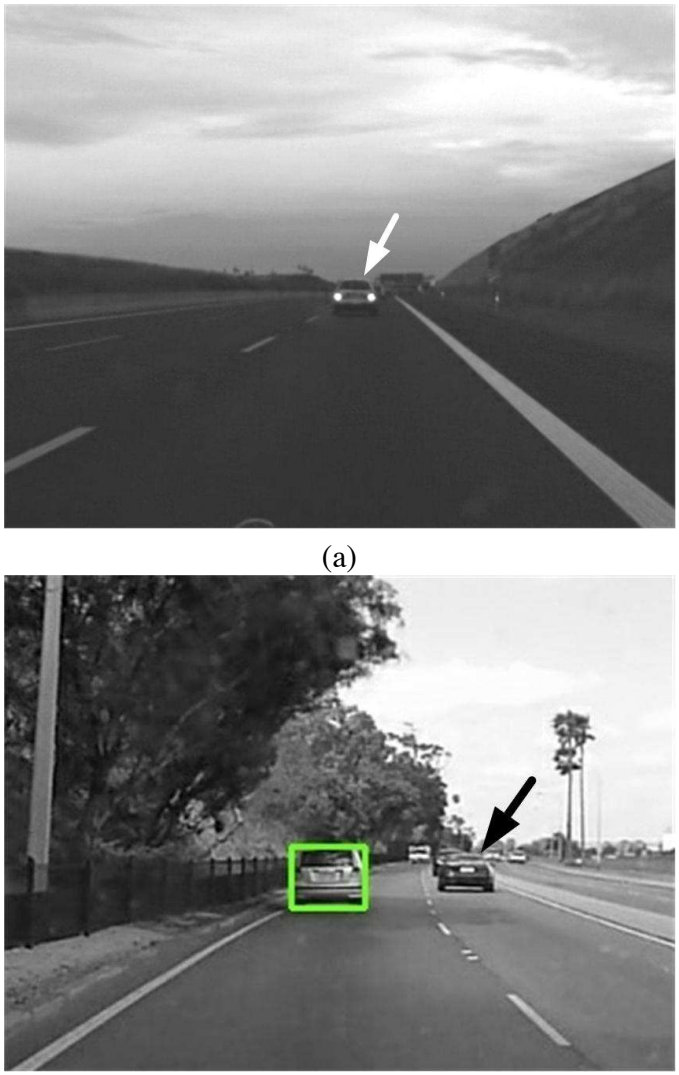


(b)

Figure 6.11: Examples of errors in the bounding box detection

Figure 6.12 (a) and (b) shows another two examples of missed detections. The vehicles were not detected due to a lack of strong edges for bounding box detection. This tends to happen more frequently on distant and dark coloured vehicles due to their low image contrast with the background.

Vision-based vehicle detection can capture the rich information about the road scene. This will enable the recognition of the road objects by just analysing the captured images. However, this is also its main disadvantage since the accuracy of the detection is largely dependent on the quality of the image. Poor lighting conditions, such as during the evening, may cause the edges of a vehicle to not being properly extracted and can result in false detection.



(a)

(b)

Figure 6.12: Examples of false negatives (missed detections) indicated by the arrows.

6.3 Summary

This chapter has described the integration of the cueing, verification and tracking modules proposed in the last three chapters to form a robust vehicle detection system. The experiments and the evaluation results on the complete system are also presented.

The test results showed that the system is able to detect multiple vehicles on the highway and the urban roads. The best detection rate was recorded on the sunny days' videos while the worst on the evening's video. This is mainly due to the differences in the road illumination. The urban road scenes are more complex compared to the highway because they are cluttered with different background objects and shadows. For this reason, it has

generated a higher number of false detections.

On the whole, the test results show that the final system can achieve close to real-time performance. It is able to detect more than 90% of the vehicles appearing in all the video frames tested in the experiment. It also yields a low number of false detections (< 5%). In conclusion, the complete system formed by the integration of the novel techniques proposed in the previous three chapters can provide a robust solution to monocular-based vehicle detection.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This thesis has proposed a technique for monocular-based vehicle detection and tracking. The technique combines several novel algorithms developed in this research. The following paragraphs provide a general conclusion for each of the stages in the proposed system, and highlight their major contributions and key findings.

- **A fast and efficient symmetry-based vehicle cueing technique**

  The vehicle cueing stage identifies the locations of all potential vehicles in the image. It requires a fast detection algorithm since a large area of the image needs to be processed. This thesis has proposed a fast and efficient symmetry-based vehicle cueing technique to address this problem.

  The technique uses multi-sized symmetry search windows to locate the symmetry points along several scan lines in the image. A fast voting-based symmetry detection algorithm is used in the calculation. The detected high symmetry points are clustered using the k-mean clustering technique. The mean location of each cluster is then used as the hypothesised vehicle's location.

  The main contributions from this part of the research are: (1) The introduction of the scan-line based symmetry search technique to significantly reduce the processing time; and (2) The use of multi-sized symmetry search windows for optimal detection of vehicles at different locations in the image. Experimental results have

shown that this technique is faster and more accurate compared to the conventional fixed size symmetry search technique or the optical flow technique.

- **Evaluation of different image features and classifiers for vehicle verification**
  The verification step verifies the hypothesised vehicles identified by the cueing stage. Vehicle verification based on pattern classification is generally more accurate compared to the template matching method. Although different types of image features and classification schemes have been proposed in the literature, there are very few comparative evaluations of their performances using the same data set.

  In this research, two efficient image features–HOG and Gabor, and three popular classifiers–SVM, ANN and Mahalanobis distance were studied and their performances for vehicle classification were systematically evaluated under the same experimental setups.

  The key findings from this part of the research are: (1) The classification performance of the HOG feature is generally better than the Gabor feature. The HOG feature also requires a shorter processing time since its feature set is more compact; (2) For both the HOG and Gabor features, the SVM classifier achieved better performance compared to the MLP or the Mahalanobis distance classifiers; and (3) The 40 highest ranking HOG features selected by the F-score feature selection technique are sufficient to represent the full 72 features set.

  Based on these findings, the best vehicle classifier that meets the performance and real-time requirement for a vehicle detection application is proposed. This classifier, the HOG reduced features set trained on the SVM, is selected to be implemented in the proposed vehicle detection system.

- **Vehicle tracking using the Kalman filter and a reliability point system**
  Once a vehicle is verified, the movement of the vehicle in the subsequent video frames will be monitored by a tracking function. The tracking function exploits the temporal coherence of the consecutive video frames to narrow down the areas for re-detecting a vehicle. This will improve the re-detection rate as well as reduce the processing time. In this research, a Kalman filter and a reliability point system are integrated into the tracking function to improve the efficiency of the tracking.

The key findings from this part of the study are: (1) A Kalman filter can efficiently predict the movement of a vehicle by only tracking its position in the image plane. The irregularities and errors in the detection can also be smoothed by the Kalman filter; and (2) The proposed reliability point system can provide a simple and fast solution to handle the problem of momentarily missed or wrong detection.

The experimental results have shown that the proposed tracking function can successfully track the preceding and the overtaking vehicles in consecutive video frames.

- **Integration of different components to form a robust vehicle detection system**
  Finally, a complete system is formed by the integration of the above three components. The system provides a novel solution to the monocular-based vehicle detection. Experimental results have shown that the system can effectively detect multiple vehicles on the highway and complex urban roads under varying weather conditions.

7.2 Future Work

In this section, some recommendations for possible areas of future work are given. These recommendations cover each of the stages in the proposed vehicle detection system.

- **Improvement for the cueing stage**
  The performance of the system can be improved by including a lane detection module. Knowing the lane boundaries will reduce the areas where the cueing stage needs to search for vehicles. It may also cut down the number of false detections by ignoring the objects outside the road regions, since these objects are not important for the purpose of collision warning. The lane detection module can also provide additional driver assistance functions such as lane departure warning.

- **Improvement for the verification stage**
  The verification stage can be improved by training different classifiers to identify different types of vehicles such as cars, SUVs and trucks. The selection of which classifier to be used during the verification process can be based on the aspect ratio

of the detected bounding box. Another possibility is to train different classifiers to verify vehicles at different viewing angles. The effects of combining multiple features on the classification performance can also be explored.

- **Improvement for the tracking stage**

  Although the tracking function has been shown to be sufficiently accurate to track the movement of vehicles in the image plane, it is still possible to improve the results by using a more complex model. For instance, the non-linear tracking techniques such as the Extended Kalman filter and Particle filter may provide more accurate tracking results. However, their performances in term of processing time and algorithm complexity should be evaluated and compared with the linear Kalman filter to find out their suitability for use in the vehicle detection system.

- **Night time vehicles detection**

  The proposed system has focused on the detection of vehicles during daytime. At nighttime, the only prominent visual feature of a vehicle is the tail lights. To make the system usable at nighttime, the detection algorithms have to be modified to detect these features. Some additional functions for detecting the road's brightness or checking the time of day can be added to allow the system to automatically switch to the most appropriate algorithms for effective detection.

- **Improving the processing speed**

  There are several possible ways to improve the processing speed of the proposed system: (1) Software optimisation– The current software was written without any specific optimisation. The frame rate can be improved by optimising some parts of the codes, for example using the integral image technique for the HOG feature extraction; and (2) Hardware acceleration– The system can be implemented on a platform with a hardware accelerator. Some of the low level functions such as the edge detection and the symmetry calculation can be implemented on FPGA or GPU to speed up their operation.

- **Driver assistance applications**

  The proposed monocular-based vehicle detection system can be used as one of the

core components for other higher level driver assistance applications such as collision warning, adaptive cruise control and brake assistance systems. The development of these applications on top of the proposed system in the same framework is another possible future extension to this research work.

References

[1] WHO: Global status report on road safety 2009. Accessed on 1st Sept 2011. URL http://www.who.int/violence_injury_prevention/road_safety_status/2009.

[2] Margie Peden et al. World report on road traffic injury prevention. Technical report, WHO, 2004.

[3] WHO: Decade of action for road safety 2011-2020: Global Launch. Accessed on 1st Sept 2011. URL http://www.who.int/roadsafety/publication/decade_launch.

[4] Olaf Gietelink. Design and Validation of Advanced Driver Assistance Systems. PhD thesis, Technical University of Delft, 2007.

[5] Soo Teoh and Thomas Bräunl. Symmetry-based monocular vehicle detection system. Machine Vision and Applications, 2011, (DOI) 10.1007/s00138-011-0355-7.

[6] Sun Zehang, G. Bebis, and R. Miller. On-road vehicle detection: a review. IEEE Transactions on Pattern Analysis and Machine Intelligence, 28(5):694–711, 2006.

[7] B. Steux, C. Laurgeau, L. Salesse, and D. Wautier. Fade: a vehicle detection and tracking system featuring monocular color vision and radar data fusion. IEEE Intelligent Vehicle Symposium, 2002, volume 2, pages 632–639 vol.2, 2002.

[8] N. Srinivasa, Chen Yang, and C. Daniell. A fusion system for real-time forward collision warning in automobiles. Proceedings of the IEEE Intelligent Transportation Systems, volume 1, pages 457–462 vol.1, 2003.

[9] U. Kadow, G. Schneider, and A. Vukotich. Radar-Vision Based Vehicle Recognition with Evolutionary Optimized and Boosted Features. IEEE Intelligent Vehicles Symposium, 2007, pages 749–754, 2007.

[10] Sergiu Nedevschi, Andrei Vatavu, Florin Oniga, and Marc Michael Meinecke. Forward collision detection using a Stereo Vision System. 4th International Conference on Intelligent Computer Communication and Processing, 2008. (ICCP 2008), pages 115–122, 2008.

[11] Kunsoo Huh, Jaehak Park, Junyeon Hwang, and Daegun Hong. A stereo vision-based obstacle detection system in vehicles. Optics and Lasers in Engineering, 46(2):168–178, 2008.

[12] M. Bertozzi, A. Broggi, A. Fascioli, and S. Nichele. Stereo vision-based vehicle detection. Proceedings of the IEEE Intelligent Vehicles Symposium, 2000. (IV 2000), pages 39–44, 2000.

[13] B. K. P. Horn and B. G. Schunck. Determining optical flow. Artificial Intelligence, 17: 185–203, 1981.

[14] S. M. Smith and J. M. Brady. ASSET-2: real-time motion segmentation and shape tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence, 17(8):814–820, 1995.

[15] B. Heisele and W. Ritter. Obstacle detection based on color blob flow. Proceedings of the Intelligent Vehicles '95 Symposium, pages 282–286, 1995.

[16] S. M. Smith and J. M. Brady. SUSAN - a new approach to low level image processing. International Journal of Computer Vision, 1(23):45–78, 1997.

[17] C. Harris and M. Stephens. A Combined Corner and Edge Detection. Proceedings of The Fourth Alvey Vision Conference, pages 147–151, 1988.

[18] Cao Yanpeng, Alasdair Renfrew, and Peter Cook. Vehicle motion analysis based on a monocular vision system. Road Transport Information and Control - RTIC 2008 and ITS United Kingdom Members' Conference, IET, pages 1–6, 2008.

[19] Cao Yanpeng, A. Renfrew, and P. Cook. Novel optical flow optimization using pulse-coupled neural network and smallest univalue segment assimilating nucleus. International Symposium on Intelligent Signal Processing and Communication Systems, (ISPACS 2007), pages 264–267, 2007.

[20] D. Willersinn and W. Enkelmann. Robust obstacle detection and tracking by motion analysis. IEEE Conference on Intelligent Transportation System, 1997. (ITSC '97), pages 717–722, 1997.

[21] Christos Tzomakas and Werner von Seelen. Vehicle detection in traffic scenes using shadows. Technical report, Institut fur Neuroinformatik, Ruhr-Universitat, Bochum, Germany, 1998.

[22] M. B. Van Leeuwen and F. C. A. Groen. Vehicle detection with a mobile camera: spotting midrange, distant, and passing cars. Robotics & Automation Magazine, IEEE, 12(1): 37–43, 2005.

[23] Margrit Betke, Esin Haritaoglu, and Larry S. Davis. Real-time multiple vehicle detection and tracking from a moving vehicle. Machine Vision and Applications, 12(2):69–83, 2000.

[24] M. Bertozzi, S. Broggi and A. Castelluccio. A real-time oriented system for vehicle detection. Journal of Systems Architecture, 43:317–325, 1997.

[25] T. Zielke, M. Brauckmann, and W. V. Seelen. Intensity and edge-based symmetry detection with an application to car-following. CVGIP: Image Understanding, 58(2), pages 177–190, 1993.

[26] A. Bensrhair, A. Bertozzi, A. Broggi, A. Fascioli, S. Mousset, and G. Toulminet. Stereo vision-based feature extraction for vehicle detection. IEEE Intelligent Vehicle Symposium, 2002, volume 2, pages 465–470 vol.2, 2002.

[27] Dai Bin, Fu Yajun, and Wu Tao. A Vehicle Detection Method via Symmetry in Multi-Scale Windows. 2nd IEEE Conference on Industrial Electronics and Applications, 2007. (ICIEA 2007), pages 1827–1831, 2007.

[28] Liu Wei, Wen XueZhi, Duan Bobo, Yuan Huai, and Wang Nan. Rear Vehicle Detection and Tracking for Lane Change Assist. IEEE Intelligent Vehicles Symposium, 2007, pages 252–257, 2007.

[29] Andreas Kuehnle. Symmetry-based recognition of vehicle rears. Pattern Recognition Letters, 12(4):249–258, 1991.

[30] Y. Du and N. P. Papanikolopoulos. Real-time vehicle following through a novel symmetry-based approach. Proceedings of IEEE International Conference on Robotics and Automation, 1997, volume 4, pages 3160–3165, 1997.

[31] T. Bucher, C. Curio, J. Edelbrunner, C. Igel, D. Kastrup, I. Leefken, G. Lorenz, A. Steinhage, and W. von Seelen. Image processing and behavior planning for intelligent vehicles. IEEE Transactions on Industrial Electronics, 50(1):62–75, 2003.

[32] Tsai Luo-Wei, Hsieh Jun-Wei, and Fan Kuo-Chin. Vehicle Detection Using Normalized Color and Edge Map. IEEE Transactions on Image Processing, 16(3):850–864, 2007.

[33] Chen Yen-Lin, Chen Yuan-Hsin, Chen Chao-Jung, and Wu Bing-Fei. Nighttime Vehicle Detection for Driver Assistance and Autonomous Vehicles. 18th International Conference on Pattern Recognition, 2006. (ICPR 2006), volume 1, pages 687–690, 2006.

[34] M.L. Eichner and T.P. Breckon. Real-time video analysis for vehicle lights detection using temporal information. IET Conference Publications, 2007(CP534):15–15, 2007.

[35] John Canny. A Computational Approach to Edge Detection. IEEE Transactions on PAMI-Pattern Analysis and Machine Intelligence, 8(6):679–698, 1986.

[36] M Y Siyal and A Solangi. A Novel Morphological Edge Detector Based Approach for Monitoring Vehicles at Traffic Junctions. Innovations in Information Technology, 2006, pages 1 –5, 2006.

[37] Li-sheng Jin, Bai-yuan Gu, Rong-ben Wang, lie Guo, Yi-bing Zhao, and Lin-hui Li. Preceding Vehicle Detection Based on Multi-characteristics Fusion. IEEE International Conference on Vehicular Electronics and Safety, 2006. (ICVES 2006), pages 356–360, 2006.

[38] Sun Zehang, R. Miller, G. Bebis, and D. DiMeo. A real-time precrash vehicle detection system. Proceedings of the Sixth IEEE Workshop on Applications of Computer Vision, 2002. (WACV 2002), pages 171–176, 2002.

[39] T. K. ten Kate, M. B. van Leewen, S. E. Moro-Ellenberger, B. J. F. Driessen, A. H. G. Versluis, and F. C. A. Groen. Mid-range and distant vehicle detection with a mobile camera. Intelligent Vehicles Symposium, 2004 IEEE, pages 72–77, 2004.

[40] Jinhui Lan and Meng Zhang. A new vehicle detection algorithm for real-time image processing system. 2010 International Conference on Computer Application and System Modeling (ICCASM), volume 10, pages V10–1 –V10–4, 2010.

[41] Qing Ming and Kang-Hyun Jo. Vehicle detection using tail light segmentation. 6th International Forum on Strategic Technology (IFOST), 2011, volume 2, pages 729 –732, 2011.

[42] Thomas Kalinke, Christos Tzomakas, and Werner V. Seelen. A Texture-based Object Detection and an adaptive Model-based Classification. in Procs. IEEE Intelligent Vehicles Symposium98, pages 341–346, 1998.

[43] Lin Peiqun, Xu Jianmin, and Bian Jianyong. Robust Vehicle Detection in Vision Systems Based on Fast Wavelet Transform and Texture Analysis. IEEE International Conference on Automation and Logistics, 2007, pages 2958–2963, 2007.

[44] Zhu Zhenfeng, Lu Hanqing, J. Hu, and K. Uchimura. Car detection based on multi-cues integration. Proceedings of the 17th International Conference on Pattern Recognition, 2004. (ICPR 2004), volume 2, pages 699–702 Vol.2, 2004.

[45] Wen Xuezhi, Zhao Hong, Wang Nan, and Yuan Huai. A Rear-Vehicle Detection System for Static Images Based on Monocular Vision. Control, Automation, Robotics and Vision, 2006. ICARCV '06. 9th International Conference on, pages 1–4, 2006.

[46] P. Parodi and G. Piccioli. A feature-based recognition scheme for traffic scenes. Proceedings of the Intelligent Vehicles '95 Symposium, pages 229–234, 1995.

[47] A. Bensrhair, M. Bertozzi, A. Broggi, P. Miche, S. Mousset, and G. Toulminet. A cooperative approach to vision-based vehicle detection. Proceedings od the IEEEIntelligent Transportation Systems, 2001, pages 207 –212, 2001.

[48] U. Handmanna, T. Kalinke, C. Tzomakas, M. Werner, and W.V Seelen. An image processing system for driver assistance. Image and Vision Computing, volume 18, pages 367–376, 2000.

[49] M. Betke, E. Haritaoglu, and L. S. Davis. Highway scene analysis in hard real-time. IEEE Conference on Intelligent Transportation System, 1997. (ITSC '97), pages 812–817, 1997.

[50] Lin Mingxiu and Xu Xinhe. Multiple Vehicle Visual Tracking from a Moving Vehicle. Sixth International Conference on Intelligent Systems Design and Applications, 2006. (ISDA '06), volume 2, pages 373–378, 2006.

[51] Z. Hu and K. Uchimura. Tracking cycle: a new concept for simultaneous tracking of multiple moving objects in a typical traffic scene. Proceedings of the IEEE Intelligent Vehicles Symposium, 2000. (IV 2000), pages 233–239, 2000.

[52] Ben Krose and Patrick van der Smagt. An introduction to Neural Networks. The University of Amsterdam, 1993.

[53] Vladimir N. Vapnik. The nature of statistical learning theory. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[54] P. C. Mahalanobis. On the generalised distance in statistics. Proceedings National Institute of Science, India, volume 2, pages 49–55, 1936.

[55] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. European Conference on Computational Learning Theory, pages 23–37, 1995.

[56] Sun Zehang, G. Bebis, and R. Miller. Boosting object detection using feature selection. Proceedings. IEEE Conference on Advanced Video and Signal Based Surveillance, 2003., pages 290–296, 2003.

[57] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. (CVPR 2005), volume 1, pages 886 –893 vol. 1, 2005.

[58] P.E. Rybski, D. Huber, D.D. Morris, and R. Hoffman. Visual classification of coarse vehicle orientation using Histogram of Oriented Gradients features. pages 921 –928, 2010.

[59] Sun Zehang, G. Bebis, and R. Miller. On-road vehicle detection using evolutionary Gabor filter optimization. IEEE Transactions on Intelligent Transportation Systems, 6(2):125–137, 2005.

[60] Ling Mao, Mei Xie, Yi Huang, and Yuefei Zhang. Preceding vehicle detection using Histograms of Oriented Gradients. International Conference on Communications, Circuits and Systems (ICCCAS 2010), pages 354 –358, 2010.

[61] Quoc Truong and Byung Lee. Vehicle Detection Algorithm Using Hypothesis Generation and Verification. Emerging Intelligent Computing Technology and Applications, volume 5754, pages 534–543, 2009.

[62] N. D. Matthews, P. E. An, D. Charnley, and C. J. Harris. Vehicle detection and recognition in greyscale imagery. Control Engineering Practice, 4(4):473–479, 1996.

[63] Constantine P. Papageorgiou and Tomasi Poggio. A Trainable Object Detection System: Car Detection In Static Images. Technical report, 1999.

[64] Paul Viola and Michael Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. Conference on Computer Vision and Pattern Recognition, (2001)., 2001.

[65] Rainer Lienhart and Jochen Maydt. An Extended Set of Haar-like Features for Rapid Object Detection. ICIP, 2002.

[66] Chunpeng Wu, Lijuan Duan, Jun Miao, Faming Fang, and Xuebin Wang. Detection of Front-View Vehicle with Occlusions Using AdaBoost. pages 1 –4, 2009.

[67] M R Turner. Texture discrimination by Gabor functions. Biol. Cybern., 55(2-3):71–82, 1986.

[68] A.C. Bovik, M. Clark, and W.S. Geisler. Multichannel texture analysis using localized spatial filters. IEEE Transactions on Pattern Analysis and Machine Intelligence, 12(1):55 –73, 1990.

[69] P. Kruizinga, N. Petkov, and S. E. Grigorescu. Comparison of texture features based on gabor filters. IEEE Transactions on Image Processing, 11:1160–1167, 2002.

[70] B.S. Manjunath and W.Y. Ma. Texture features for browsing and retrieval of image data. IEEE Transactions on Pattern Analysis and Machine Intelligence, 18(8):837 –842, 1996.

[71] Sun Zehang, G. Bebis, and R. Miller. On-road vehicle detection using Gabor filters and support vector machines. 14th International Conference on Digital Signal Processing, 2002. (DSP 2002), volume 2, pages 1019–1022, 2002.

[72] Cheng Hong, Zheng Nanning, and Sun Chong. Boosted Gabor Features Applied to Vehicle Detection. 18th International Conference on Pattern Recognition, 2006. (ICPR 2006), volume 1, pages 662–666, 2006.

[73] Zhang Yan, S. J. Kiselewich, and W. A. Bauson. Legendre and gabor moments for vehicle recognition in forward collision warning. IEEE Intelligent Transportation Systems Conference, 2006. (ITSC '06), pages 1185–1190, 2006.

[74] D. Alonso, L. Salgado, and M. Nieto. Robust Vehicle Detection Through Multidimensional Classification for on Board Video Based Systems. IEEE International Conference on Image Processing, 2007. (ICIP 2007), volume 4, pages IV – 321–IV – 324, 2007.

[75] U. Handmann and T. Kalinke. Fusion of texture and contour based methods for object recognition. IEEE Conference on Intelligent Transportation System, 1997. (ITSC '97), pages 876–881, 1997.

[76] Robert M. Haralick, K. Shanmugam, and Its'Hak Dinstein. Textural Features for Image Classification. IEEE Transactions on Systems, Man and Cybernetics, 3(6):610 –621, 1973.

[77] Milos Stojmenovic. Real Time Machine Learning Based Car Detection in Images With Fast Training. Machine. Vision and Applications, 17:163–172, 2006.

[78] B. G. Batchelor and P. F. Whelan. Intelligent Vision Systems for Industry. Springer Verlag, 1997.

[79] T. Danisman, I.M. Bilasco, C. Djeraba, and N. Ihaddadene. Drowsy driver detection system using eye blink patterns. International Conference on Machine and Web Intelligence, 2010 (ICMWI), pages 230 –233, 2010.

[80] David Thomas John O'Mara. Automated Facial Metrology. PhD thesis, Department of Computer Science and Software Engineering, University of Western Australia, 2002.

[81] Q.B. Sun, W.M. Huang, and J.K. Wu. Face detection based on color and local symmetry information. Proceedings of the Third IEEE International Conference on Automatic Face and Gesture Recognition, 1998, pages 130 –135, 1998.

[82] Daniel Reisfeld, Haim Wolfson, and Yehezkel Yeshuru. Context Free Attentional Operators: the Generalized Symmetry Transform. International Journal of Computer Vision, 14:119–130, 1995.

[83] G. Marola. On the detection of the axes of symmetry of symmetric and almost symmetric planar images. IEEE Transactions on Pattern Analysis and Machine Intelligence, 11(1): 104–108, 1989.

[84] H. Zabrodsky, S. Peleg, and D. Avnir. A measure of symmetry based on shape similarity. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1992. (CVPR '92), pages 703 –706, 1992.

[85] Hermann Weyl. Symmetry. Princeton University Press, Princeton, New Jersey, 1952.

[86] D. Shen, H.H.S. Ip, and Eam Khwang Teoh. A novel theorem on symmetries of 2D images. volume 3, pages 1002 –1005 vol.3, 2000.

[87] Nahum Kiryati and Yossi Gofman. Detecting symmetry in grey level images: The global optimization approach. In Proceedings of the 13th International Conference on Pattern Recognition, volume I, pages 951–956, 1996.

[88] Sun Changming. Symmetry detection using gradient information. Pattern Recognition Letters, 16(9):987, 1995.

[89] D. Westhoff, J. Zhang, and K. Huebner. Robust illumination-invariant features by quantitative bilateral symmetry detection. IEEE International Conference on Information Acquisition, 2005, page 6 pp., 2005.

[90] Kai Huebner. A Symmetry Operator and its. Application to the RoboCup. RoboCup Symposium 2003, Padua, 2003.

[91] Minsu Cho and Kyoung Mu Lee. Bilateral Symmetry Detection and Segmentation via Symmetry-Growing. Proc. British Machine Vision Conference(BMVC), 2009.

[92] Vito Di Gesu and Cesare Valenti. Symmetry Operators in Computer Vision. Vistas Astron, 1995.

[93] Xiao Zhitao and Wu Jun. Analysis on Image Symmetry Detection Algorithms. Fourth International Conference on Fuzzy Systems and Knowledge Discovery, 2007. (FSKD 2007), volume 4, pages 745–750, 2007.

[94] Parky Minwoo, Leey Seungkyu, Cheny Po-Chun, S. Kashyap, A. A. Butty, and Liuy Yanxi. Performance evaluation of state-of-the-art discrete symmetry detection algorithms. IEEE Conference on Computer Vision and Pattern Recognition, 2008. (CVPR 2008), pages 1–8, 2008.

[95] Yan Zhang, Arnab S. Dhua, Stephen J. Kiselewich, and William A. Bauson. *Embedded Computer Vision: Challenges of Embedded Computer Vision in Automotive Safety Systems*, pages 257–279. Springer Science and Business Media, 2009.

[96] Peter Kovesi. Symmetry and Asymmetry from Local Phase. Tenth Australian Joint Converence on Artificial Intelligence, pages 2–4, 1997.

[97] Zhitao Xiao, Zhengxin Hou, Changyun Miao, and Jianming Wang. Using phase information for symmetry detection. Pattern Recognition Letters, 26(13):1985 – 1994, 2005.

[98] D.G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. Int. J. Comput. Vision, 60(2):91–110, 2004.

[99] Chang Hao-Yuan, Fu Chih-Ming, and Huang Chung-Lin. Real-time vision-based preceding vehicle tracking and recognition. Proceedings of the IEEE Intelligent Vehicles Symposium, 2005, pages 514–519, 2005.

[100] AARoads - the online highway guide. Accessed on 22nd June, 2011. URL `http://www.aaroads.com`.

[101] Rafael C. Gonzalez and Richard E. Woods. Digital Image Processing. Prentice Hall Inc., Boston, MA, USA, 2002.

[102] B. Zafarifar, H. Weda, and P. H. N. de With. Horizon detection based on sky-color and edge features. Society of Photo-Optical Instrumentation Engineers SPIE Conference Series, volume 6822, 2008.

[103] Jinyou Zhang and H.-H. Nagel. Texture-based segmentation of road images. Proceedings of the Intelligent Vehicles '94 Symposium, pages 260 – 265, 1994.

[104] H. Steinhaus. Sur la division des corp materiels en parties (in French). Bulletin of the Polish Academy of Sciences and Mathematics, 4:801–804, 1956.

[105] A. Boeing and T. Braunl. ImprovCV: Open component based automotive vision. IEEE Intelligent Vehicles Symposium, 2008, pages 297–302, 2008.

[106] S. Hawe. A Component-Based Image Processing Framework for Automotive Vision. Master's thesis, University Of Western Australia, 2008.

[107] Intel. OpenCV (Open Source Computer Vision) Library Wiki. URL: http://opencv.willowgarage.com/wiki/. Accessed on 1st June 2010.

[108] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. Proceedings of the 1981 DARPA Imaging Understanding Workshop, pages 121–130, 1981.

[109] Jean yves Bouguet. Pyramidal implementation of the Lucas Kanade feature tracker. Intel Corporation, Microprocessor Research Labs, 2000.

[110] Kumar Nath Rajiv and Kumar Deb Swapan. On road vehicle/object detection and tracking using template. Indian Journal of Computer Science and Engineering, 1(2):98–107, 2010.

[111] Thiang, Teguh Guntoro Andre, and Lim Resmana. Type of Vehicle Recognition Using Template Matching Method. Proc. of the International Conf. on Electrical, Electronics, Communication and Information (CECI), 2001.

[112] Shigeo Abe. Support Vector Machines for Pattern Classification. Springer-Verlag London Limited, 2005.

[113] K.-R. Muller, S. Mika, G. Ratsch, K. Tsuda, and B. Scholkopf. An introduction to kernel-based learning algorithms. IEEE Transactions on Neural Networks, 12(2):181 –201, 2001.

[114] Christopher J.C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. Data Mining and Knowledge Discover, 2:121–167, 1998.

[115] C.-W. Hsu, C.-C. Chang, and C.-J. Lin. A practical guide to support vector classification. Technical report, Department of Computer Science, National Taiwan University., 2003.

[116] M. Hornik, K.and Stinchcombe and H. White. Multilayer feedforward networks are universal approximators. Neural Networks. Neural Networks, 2(5):359–366, 1989.

[117] Stuart J. Russell and Peter Norvig. Artificial Intelligence A Modern Approach. Prentice Hall Inc., 1995.

[118] Martin Riedmiller and Heinrich Braun. A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. IEEE International Conference on Neural Networks, pages 586–591, 1993.

[119] Y. LeCun, L. Bottou, G. Orr, and K. Muller. Efficient BackProp. Neural Networks: Tricks of the trade, 1998.

[120] Suli Zhang and Xin Pan. A novel text classification based on Mahalanobis distance. 3rd International Conference on Computer Research and Development (ICCRD 2011), volume 3, pages 156 –158, 2011.

[121] Guang Chen, Hong-Gang Zhang, and Jun Guo. Efficient Computation of Mahalanobis Distance in Financial Hand-Written Chinese Character Recognition. International Conference on Machine Learning and Cybernetics, 2007, volume 4, pages 2198 –2201, 2007.

[122] S. Ramaswamy and K. Rose. Fast adaptive Mahalanobis distance-based search and retrieval in image databases. 15th IEEE International Conference on Image Processing, 2008. (ICIP 2008), pages 181 –184, 2008.

[123] T. Kamei. Face retrieval by an adaptive Mahalanobis distance using a confidence factor. Proceedings of the International Conference on Image Processing. 2002, volume 1, pages I–153 – I–156 vol.1, 2002.

[124] Huijie Ji, Meihua Xu, and Feng Ran. Auto classification of skin symptom based on Mahalanobis distance. 3rd International Conference on Advanced Computer Theory and Engineering, 2010 (ICACTE), volume 6, pages V6–299 –V6–302, 2010.

[125] F. Babiloni, L. Bianchi, F. Semeraro, J. del R Millan, J. Mourino, A. Cattini, S. Salinari, M.G. Marciani, and F. Cincotti. Mahalanobis distance-based classifiers are able to recognize EEG patterns by using few EEG electrodes. Proceedings of the 23rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society, volume 1, pages 651 – 654 vol.1, 2001.

[126] K. Younis, M. Karim, R. Hardie, J. Loomis, S. Rogers, and M. DeSimio. Cluster merging based on weighted mahalanobis distance with application in digital mammograph. Proceedings of the IEEE National Aerospace and Electronics Conference, 1998. (NAECON), pages 525 –530, 1998.

[127] M. Bertozzi, A. Broggi, M. Del Rose, M. Felisa, A. Rakotomamonjy, and F. Suard. A Pedestrian Detector Using Histograms of Oriented Gradients and a Support Vector Machine Classifier. IEEE Intelligent Transportation Systems Conference, (ITSC 2007), pages 143 –148, 2007.

[128] Hui-Xing Jia and Yu-Jin Zhang. Fast Human Detection by Boosting Histograms of Oriented Gradients. Fourth International Conference on Image and Graphics, 2007. (ICIG 2007), pages 683 –688, 2007.

[129] K. Lillywhite, Dah-Jye Lee, and Dong Zhang. Real-time human detection using histograms of oriented gradients on a GPU. Workshop on Applications of Computer Vision, 2009 (WACV), pages 1 –6, 2009.

[130] Ning He, Jiaheng Cao, and Lin Song. Scale Space Histogram of Oriented Gradients for Human Detection. International Symposium on Information Science and Engieering, 2008. (ISISE '08), volume 2, pages 167 –170, 2008.

[131] Ge Junfeng, Luo Yupin, and Tei Gyomei. Real-Time Pedestrian Detection and Tracking at Nighttime for Driver-Assistance Systems. IEEE Transactions on Intelligent Transportation Systems, 10(2):283–298, 2009.

[132] Qiang Zhu, Mei-Chen Yeh, Kwang-Ting Cheng, and S. Avidan. Fast Human Detection Using a Cascade of Histograms of Oriented Gradients. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2006, volume 2, pages 1491 – 1498, 2006.

[133] Xifeng Ding, Hui Xu, Peng Cui, Lifeng Sun, and Shiqiang Yang. A cascade SVM approach for head-shoulder detection using histograms of oriented gradients. IEEE International Symposium on Circuits and Systems, 2009. (ISCAS 2009), pages 1791 –1794, 2009.

[134] J.F. Khan, R.R. Adhami, and S.M.A. Bhuiyan. Color image segmentation utilizing a customized Gabor filter. IEEE SoutheastCon, 2008, pages 539 –544, 2008.

[135] Du Yuren and Feng Yuan. Vehicle detection from video sequence based on gabor filter. 9th International Conference on Electronic Measurement & Instruments, 2009. (ICEMI '09), pages 2–375–2–379, 2009.

[136] Han Feng, Shan Ying, Cekander Ryan, S. Sawhney Harpreet, and Kumar Rakesh. A Two-Stage Approach to People and Vehicle Detection With HOG-Based SVM. 2006.

[137] Rudra N. Hota, Kishore Jonna, and P. Radha Krishna. On-road vehicle detection by cascaded classifiers. Proceedings of the Third Annual ACM Bangalore Conference, pages 27:1–27:5, 2010.

[138] Zhenjun Han, Qixiang Ye, and Jianbin Jiao. Online feature evaluation for object tracking using Kalman Filter. 19th International Conference on Pattern Recognition, 2008. (ICPR 2008), pages 1 –4, 2008.

[139] J. G. Daugman. Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. Journal of Optical Society of America A, 2:1160–1169, 1985.

[140] Yiming Ji, Kai H. Chang, and Chi-Cheng Hung. Efficient edge detection and object segmentation using Gabor filters. Proceedings of the 42nd annual Southeast regional conference, pages 454–459, 2004.

[141] Ramzi Abiantun and Marios Savvides. Boosted multi image features for improved face detection. Applied Image Pattern Recognition Workshop,, 0:1–8, 2008.

[142] Chih-Jen Lee, Tai-Ning Yang, Chun-Jung Chen, A.Y. Chang, and Sheng-Hsuan Hsu. Fingerprint identification using local Gabor filters. Sixth International Conference on Networked Computing and Advanced Information Management (NCM 2010), pages 626 – 631, 2010.

[143] Linlin Shen and Li Bai. Information Theory for Gabor Feature Selection for Face Recognition. EURASIP Journal on Applied Signal Processing, 2006:1–11, 2006.

[144] Ji Peijin, Jin Lianwen, and Li Xutao. Vision-based Vehicle Type Classification Using Partial Gabor Filter Bank. IEEE International Conference on Automation and Logistics, 2007, pages 1037–1040, 2007.

[145] Chengjun Liu. Gabor-based Kernel PCA with Fractional Power Polynomial Models for Face Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 26: 572–581, 2004.

[146] Mark S. Nixon and Alberto S. Aguado. Feature Extraction and Image Processing. Academic Press, 2008.

[147] I.T Jolliffe. Principal Component Analysis. Springer-Verlag New York, Inc., 2002.

[148] Aapo Hyvrinen. Survey on Independent Component Analysis. Neural Computing Surveys, 2:94–128, 1999.

[149] Kanti V. Mardia, J. T. Kent, and J. M. Bibby. Multivariate Analysis (Probability and Mathematical Statistics). Academic Press, 1980.

[150] Imola Fodor. A Survey of Dimension Reduction Techniques. Technical report, LLNL technicalreport, 2002.

[151] W. J. Krzanowski. Selection of Variables to Preserve Multivariate Data Structure, Using Principal Components. Journal of the Royal Statistical Society. Series C (Applied Statistics), 36(1):pp. 22–33, 1987.

[152] Yi-Wei Chen and Chih-Jen Lin. *Feature extraction, foundations and applications*. Springer, 2006.

[153] Xu Yang, Liu Jia, Hu Qingmao, Chen Zhijun, Du Xiaohua, and Heng Pheng Ann. F-score feature selection method may improve texture-based liver segmentation strategies. IEEE International Symposium on IT in Medicine and Education (ITME 2008), pages 697 –702, 2008.

[154] C. Guerrero-Mosquera, M. Verleysen, and A.N. Vazquez. EEG feature selection using mutual information and support vector machine: A comparative analysis. Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), pages 4946 –4949, 2010.

[155] S. Gunes, K. Polat, and S. Yosunkaya. Multi-class f-score feature selection approach to classification of obstructive sleep apnea syndrome. Expert Systems with Applications, 37 (2):998 – 1004, 2010.

[156] C.C. Chang and C.J. Lin. LIBSVM: a library for support vector machines. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm, 2001.

[157] Dan Simon. Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches. Wiley-Interscience, 2006.

[158] K. Markus. Using the Kalman Filter to track Human Interactive Motion Modelling and Initialization of the Kalman Filter for Translational Motion. Technical report, University of Dortmund, Germany, 1997.

[159] G Welch and G Bishop. An introduction to the Kalman Filter. SIGGRAPH, 2001.

Appendix A

Format of Feature Vector Files for the Experiments in Chapter 4

Below is the format of the feature vector file used in the experiment for Chapter 4. The format is based on [156] to facilitate the training or classification using the LibSVM library.

```
<class label> 1:<feature 1 value> 2:<feature 2 value> ... n:<feature n value>
```

An example of the feature vector file is given below. Each row stores the values for a different sample. In this example there are 15 samples with six features. The first 8 rows with class label 1 is the positive samples and the remaining with class label −1 are the negative samples.

```
 1 1:-0.724551 2:-0.417989 3:-0.180556 4:-0.385417 5:-0.530864 6:-0.534592
 1 1:-0.616766 2:-0.597884 3:-0.138889 4:-0.333333 5:-0.580247 6:-0.543473
 1 1:-0.772455 2:-0.365079 3:-0.222222 4:-0.364583 5:-0.555556 6:-0.456528
 1 1:-0.664671 2:-0.121693 3:-0.805556 4:-0.270833 5:-0.798778 6: 0.076087
 1 1:-0.616766 2:-0.058201 3:-0.791667 4:-0.385417 5:-0.790123 6: 0.119565
 1 1:-0.640719 2:-0.111111 3:-0.819444 4:-0.291667 5:-0.775778 6: 0.097826
 1 1:-0.640719 2:-0.058201 3:-0.763889 4:-0.385417 5:-0.814815 6: 0.086956
 1 1:-0.568862 2:-0.132275 3:-0.875012 4:-0.281255 5:-0.925926 6: 0.086956
-1 1:-0.592814 2:-0.047619 3:-0.958333 4:-0.291667 5:-0.654321 6: 0.043478
-1 1:-0.760479 2:-0.936508 3:-0.694433 4: 0.447917 5:-0.814815 6:-0.967391
-1 1:-0.772455 2:-0.862434 3:-0.534623 4: 0.322917 5:-0.851852 6:-0.793478
-1 1:-0.784431 2:-0.703704 3:-0.753346 4:-0.645833 5:-0.864198 6:-0.728261
-1 1:-0.508982 2:-0.174603 3:-0.680556 4:-0.604167 5:-0.419753 6:-0.108696
-1 1:-0.784431 2: 0.280423 3:-0.888889 4:-0.565656 5:-0.876543 6: 0.586957
-1 1:-0.712575 2: 0.142857 3:-0.930556 4:-0.385417 5:-0.790123 6:-0.423913
```