Embedded computing techniques for vision-based lane change decision aid systems

Satzoda Ravi Kumar

2013

Satzoda Ravi Kumar. (2013). Embedded computing techniques for vision-based lane change decision aid systems. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/54953>

<https://doi.org/10.32657/10356/54953>

**EMBEDDED COMPUTING TECHNIQUES FOR
VISION-BASED LANE CHANGE DECISION AID SYSTEMS**

SATZODA RAVI KUMAR
SCHOOL OF COMPUTER ENGINEERING
2013

EMBEDDED COMPUTING TECHNIQUES FOR VISION-BASED LANE CHANGE DECISION AID SYSTEMS

SATZODA RAVI KUMAR

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Doctor of Philosophy

2013

*Dedicated to my parents, wife, teachers, and
above all my Guru
Paramahansa Nithyananda*

Acknowledgements

There are many people who have enabled me directly or indirectly to complete this research successfully. First and foremost, I would like to thank Professor Thambipillai Srikanthan whose continuous guidance has helped me in all my research experience, including the research work presented in this thesis. Without this support at both professional and personal levels, I would not have been able to bring this research to this level of completion. I would like to express my sincere gratitude to him once again.

Next, I would like to convey my heartfelt gratitude to my wife and my best friend, Dr. Suchitra Sathyanarayana. She has been an inspiration and guiding force in my research career and her inputs and critical analysis of my work has been instrumental in shaping my way of thinking and expressing myself in my research and personal front.

My research started with my undergraduate Final Year Project with Assoc. Prof. Chip-Hong Chang from School of EEE, with whom I have a long standing research relationship. I would like to thank him for his continued support and encouragement. I would also like to thank various members of my research center - Center for High Performance Embedded Systems (CHiPES), where all this research work was developed. In particular, I would like to thank Ms. Boh-Nah Kiat Joo, Ms. Yap Marilyn and Mr. Jeremiah for supporting me with all necessary logistics while I was in CHiPES.

I would like to acknowledge the support that I have received from my sister Supriya Sathyanarayana, who was also instrumental in developing some ideas presented in this thesis as a student intern. In addition, I would like to thank my friends and colleagues who have supported me in various ways during the course of this research. I thank my friends Mr. G. P. Muttukrishnan and Mrs. Chitralekha who helped me collect drive test data using his car.

I thank my parents and parents-in-law for all the support over the last few years, and the trust they had in me to complete this research successfully. I express my deep gratitude to them for all the love they have been showering on me all my life.

My thesis, my research and my life would be incomplete without the presence of my Guru Paramahansa Nithyananda. I surrender everything that I have achieved during the course of this research at His holy feet. His presence and His blessings have been the guiding force in keeping myself grounded in basic human values, without being carried away by my ego.

Contents

Acknowledgements	iii
List of Figures	ix
List of Tables	xv
Abstract	xvii
1 Introduction	1
1.1 Motivation	1
1.2 Research Objectives	2
1.3 Research Contributions	2
1.3.1 Research Publications from This Work	3
1.4 Organization of the Thesis	5
2 Literature Survey	7
2.1 Advanced Driver Assistance Systems for Lane Change Assistance	7
2.1.1 Need for LCDAS	10
2.2 Functional Blocks in LCDAS	10
2.3 Lane Detection	11
2.3.1 Sensors for Lane Detection	11
2.3.2 Vision-based Lane Detection Techniques	12
2.3.3 Multi-lane Detection	16
2.3.4 Hough Transform for Lane Detection	17
2.4 Vehicle Detection	18
2.4.1 Vehicle Detection using Active Sensors	18
2.4.2 Vehicle Detection using Vision Sensors	20
2.4.2.1 Hypothesis Generation	21
2.4.2.2 Hypothesis Verification (HV)	24
2.5 Estimating Vehicle Proximity	26
2.6 Blind-spot Monitoring	27
2.7 Driver Intention Detection	28
2.8 Situation Assessment for Lane Change Decision Making	29
2.9 Existing Lane Change Decision Aid Systems	31

2.9.1	Functional Requirements of LCDAS: ISO 17387	31
2.9.2	Commercial LCDAS in Market	32
2.9.3	On-going Research Efforts in LCDAS	34
2.10	Summary	35
3	Detection of Host & Neighbor Lanes in Near-view	39
3.1	Introduction	39
3.2	Iterative Technique for Detecting Lanes in Near-view	41
3.2.1	Gradient Magnitude Histograms (GMH) for Sensing Edges of Vary- ing Prominence	42
3.2.1.1	Block-based GMH Generation	46
3.2.2	Detecting Host & Multiple Lanes using GMH	47
3.2.2.1	GMH-GAH-HT Engine to Extract Lane Edges	48
3.2.2.2	GMH-GAH-HT Engine for Robust Lane Detection	52
3.3	Dataset Generation	63
3.4	Results & Discussion	63
3.4.1	Evaluation Method	64
3.4.2	Accuracy Results	64
3.4.3	Computation Complexity Analysis	73
3.5	Computational Architecture for Multiple-lane Detection	78
3.6	Summary & Conclusions	79
4	Detection of Host & Neighbor Lanes in Far-view	81
4.1	Introduction	81
4.2	Extracting Host & Neighbor Lanes in Far-view	82
4.3	Extending Block-based Technique for Detecting Curved Lanes	86
4.4	Results & Evaluation	90
4.4.1	Straight lane detection in far-view	90
4.4.2	Curved lanes in far-view	93
4.5	Computational Challenges	96
4.6	Parallelizing Hough Transform Computation	97
4.6.1	Parallel Additive Hough Transform (AHT)	98
4.6.2	Block-based AHT Computation Architecture	101
4.6.3	Performance & Computation Cost Evaluation of AHT	104
4.7	Hierarchical AHT (HAHT) for Efficient Global Hough Space Generation	113
4.7.1	Extending the additive property to multiple levels	114
4.7.2	Deriving the Hierarchical AHT	116
4.7.3	HAHT for Detection of Straight Lanes	118
4.7.4	Performance & Computation Cost Evaluation of HAHT	119
4.8	Computation Cost Analysis for Lane Detection using Proposed AHT and HAHT	121
4.9	Summary	123
5	Estimating Proximity of Target Vehicles from Host Vehicle	125
5.1	Introduction	125
5.2	Deriving RoIs for Vehicle Detection	127
5.3	Vehicle Detection using Multiple Cues	130
5.3.1	Detecting Under-vehicle Shadow	130

5.3.2	Confirming Vehicle Signature using Horizontal & Slant Edge Symmetries	140
5.3.3	Overall Algorithm for Vehicle Detection in Lanes	145
5.4	Motorcycle Detection	145
5.5	Estimating Relative Proximity using Monocular Camera	151
5.5.1	Proposed Model for Relative Proximity Estimation	154
5.5.1.1	Experimental Verification of the Model	156
5.5.2	Estimating Proximity of Vehicles in Neighboring Lanes	157
5.6	Results & Discussion	158
5.6.1	Accuracy Evaluation	158
5.6.2	Computational Complexity Analysis	161
5.7	Computational Architecture for Vehicle Detection	164
5.8	Summary	164
6	Techniques for Lane Change Decision Aid	167
6.1	Introduction	167
6.2	Sensor Configuration for Surround Scene Sensing	168
6.3	Defining Vehicle State Parameters	169
6.4	State-dependent Risk Factor Computation	170
6.5	Lane Change Decision Aid Process	179
6.6	State Machine for Blind Spot Monitoring	183
6.7	Computation Framework for Proposed Lane Change Decision Aid System	187
6.8	Summary	188
7	Conclusions & Future Work	191
7.1	Conclusions	191
7.2	Future Work	194

List of Figures

2.1	Core functional blocks in a lane change decision aid system.	11
3.1	(a) Image patch I showing lanes of varying prominence (b) Gradient magnitudes of I showing lanes of varying gradient magnitudes (c) Edge map E generated by setting $T_1 = 300$, only host lane edges are detected, (d) E using $T_2 = 200$, both host lanes and neighbor lanes are detected. . . .	40
3.2	(a) Image patch I showing lanes and other features with gradients of varying prominence (b) GMH m for the image patch (c) Zoomed in m showing clusters at higher gradients (d) Edge map from the first group G ₁ (e) Edge map from the first group G ₂	44
3.3	GMH with peaks and group boundaries indicated	45
3.4	Edges extracted from the different edge groups (a) input image block, (b) edge map from MATLAB Sobel edge map generation, (c) edges from G ₁ and G ₂ , (d) edges from G ₁ , G ₂ , G ₃ (e) edges from G ₁ , G ₂ , G ₃ and G ₄	46
3.5	Block diagram showing the functional flow of the proposed iterative method for lane detection	51
3.6	Illustration showing the division of input image into blocks.	53
3.7	Block diagram showing proposed host lane detection method in near-view using GMH-GAH-HT engine	54
3.8	Vanishing point from host lanes can be used to predict possible locations of neighbor lanes.	56
3.9	Block diagram showing proposed neighbor lane detection method in near-view using GMH-GAH-HT engine	57
3.10	Host and neighbor lane detection with prediction modules	58
3.11	Carriageway boundaries and respective edge maps (a)(e) Double lane markings, (b)(f) Road divider, (c)(g) Broad continuous lane marking and (d)(h) Raised kerb	60
3.12	(a) Test car, (b) & (c) Camera setup inside the car behind the rear view mirror.	63
3.13	Map of Singapore showing the different routes which were captured, either completely or partially, for dataset generation	64
3.14	Sequence showing detection of lane markings using both GMH-GAH-HT engine and temporal history based prediction.	67
3.15	Test cases where there are no visible lane markings	67
3.16	Faint host lane markings detected by the proposed GMH-GAH-HT engine	68

3.17	Lane markings of varying prominence detected by the proposed GMH-GAH-HT engine	69
3.18	Host and neighbor lane markings detected by the proposed GMH-GAH-HT engine under varying weather and lighting conditions: Presence of shadows in (a)& (b), rainy weather conditions in (c) & (d), and night scene in (e) & (f)	70
3.19	Mis-detections due to presence of other road markings	71
3.20	Detection of lanes beyond neighbor lanes	72
3.21	Collection of ground truth data for computing lane position deviation	72
3.22	Plots showing lane position deviation from ground truth data for Seq #1 in top plot and Seq #3 in the bottom plot.	73
3.23	Computation architecture for detecting host and neighbor lanes in near-view	79
3.24	Processing engine (PE) in computation architecture for detecting host and neighbor lanes in near-view	79
4.1	Prominence of lane markings' reduces from near-view to far-view. Size of lane marking features also decreases.	82
4.2	Defining far-view region using lanes from near-view.	83
4.3	Detecting host & neighbor lanes in far-view by generating global Hough spaces from block level Hough spaces	84
4.4	Detecting host and neighbor lanes in far-view by generating global Hough spaces from block level Hough spaces	85
4.5	Straight lanes and their locations in blocks	86
4.6	Straight lanes and their locations in blocks	88
4.7	Curved lane in far-view	89
4.8	Linear approximation of curved lane in far-view	90
4.9	Illustration showing linear approximation of curved lane in far-view	91
4.10	Curved road markings in far-view not picked up by edge detection.	93
4.11	Illustration to show the effect of block size on linear approximation of the lane curvature. (a) Input image divided into blocks of size 90×128 , (b) Blocks showing edge map of the curved right lane, (c) Smaller blocks dividing the curvature further, (e) Linear approximation of the curve using bigger blocks, (f) Linear approximation of the curve using smaller blocks.	96
4.12	Additive property of the Hough transform; l_1 and l_2 are two parallel lines passing through A and B respectively.	99
4.13	Edge map divided into 16 blocks using a 4×4 grid and showing spatially equivalent points in two different blocks.	99
4.14	Edge maps of two test images in (a) and (b) followed by Hough spaces generated by conventional HT in (c) and (d) and the proposed AHT in (e) and (f).	101
4.15	An 8×8 block to illustrate Theorem 1. The local HTs of pixel coordinates marked with $\times$ only need to be stored.	102
4.16	Proposed AHT-based parallel implementation.	103
4.17	Comparison of total computation time (T) in cycles vs. image size $m \times m$	106

4.18	Total computation time (T) in cycles in log scale vs. grid resolution k for the proposed AHT-based implementation. For smaller m like 128, step-like decrease is seen because of the ceiling function introduced to cater for cases where k is not a factor of m	107
4.19	Area of the proposed implementation plotted against grid resolution k . . .	109
4.20	LUT memory area and computational array area contributing to the total area cost curve for 128×128 sized edge map plotted against grid resolution k to show the push-pull effect on area cost. All area costs are indicated in the number of gates as shown in Table 4.5.	110
4.21	Comparison of AT measure of the proposed implementation against different implementations for varying m and k	111
4.22	VLSI metric AT in log scale vs. image size $m \times m$. AT measure of the proposed implementation is computed at optimal grid resolution k_{opt} . . .	112
4.23	Comparison of total computation time in cycles (log scale) of the proposed implementation against existing designs for different video processing applications.	114
4.24	Comparison of VLSI metric AT in log scale vs. image size $m \times n$ for different video processing applications.	114
4.25	Extending the additive property of HT across multiple levels	115
4.26	Division of edge map into blocks and macroblocks and their origins	116
4.27	Detection of long lines using HAHT	118
4.28	Edge maps (a) Lane1, (b) Lane2, (c) Lane3 and (g) Lane4 and corresponding detected lanes in (d), (e), (f) and (h) using the proposed HAHT	120
5.1	Envisaged steps of the proposed vehicle proximity estimation module for lane change assistance	126
5.2	Illustrating mask creation	128
5.3	Illustrating mask creation for lanes represented by line segments. The axes indicates the indices for the blocks.	128
5.4	Illustrating mask creation for multiple line segments	129
5.5	Examples of under-vehicle shadows under different conditions (a) bright day light (b) presence of shadows of trees completely (c) change is overall illumination due to clouds (d)-(f) brighter illumination with shadows showing different variations of under-vehicle shadow intensities	131
5.6	Umbra and penumbra shadows seen on a road scene. The solid box shows umbra shadow cast by the vehicle and the dashed box shows a penumbra shadow by the foliage of trees.	132
5.7	Regions A_1 and A_2 selected on the image $\mathbf{I}$ for the experiment.	133
5.8	Scatter plot between $\bar{\mathbf{I}}_{A_1}$ and $\bar{\mathbf{I}}_{A_2}$	134
5.9	Scatter plot between $\bar{\mathbf{I}}_{A_1}$ and $\mathbf{I}_\Delta$	135
5.10	$\mathbf{I}_\Delta$ versus $\bar{\mathbf{I}}_{A_1}$	136
5.11	The image block (c) and RoI block (b) are shown for an input image with mask (a). The division of the image block into bands is shown in (d) and (e). The bands are further divided into sub-blocks in (f) and (g).	137
5.12	The image block $\mathbf{I}_{R_6}$ showing blocks $\mathbf{I}_{R_{6,8}}$ and $\mathbf{I}_{R_{6,20}}$ with road intensities.	139
5.13	Example showing under-vehicle shadow detection.	140
5.14	The image block $\mathbf{I}_{R_6}$ showing blocks $\mathbf{I}_{R_{6,8}}$ and $\mathbf{I}_{R_{6,20}}$ with road intensities.	141

5.15	Illustration on slant edge symmetry. From top: the first row shows the gradient magnitudes of the left and right halves of the vehicle image block (shown in the lowest row); the second row shows the GAHs for the left and right halves of the vehicle image block, with slant edge peaks circled.	144
5.16	Flowchart for the proposed vehicle detection algorithm.	144
5.17	Variations in motorcycle position & orientation.	146
5.18	(a) Scatter plots of $\bar{\mathbf{I}}_{A_1}$ versus $\bar{\mathbf{I}}_{A_2}$ and (b) $\bar{\mathbf{I}}_{A_1}$ versus $\bar{\mathbf{I}}_{\Delta}$	147
5.19	Illustrating the blocks around the motorcycle.	148
5.20	Scatter plot of $\bar{\mathbf{I}}_{\Delta l}$ versus $\bar{\mathbf{I}}_{\Delta r}$ for vehicle and non-vehicle training data. The red asterisks indicate non-vehicle samples and blue circles indicate vehicle samples.	149
5.21	Scatter plot of $\bar{\mathbf{I}}_{\Delta l}$ versus $\bar{\mathbf{I}}_{\Delta r}$ for vehicle and non-vehicle training data with the linear decision functions f_{MC1} , f_{MC2} and f_{MC3}	150
5.22	Illustration showing relative positions of the edges remains fixed as the motorcycle moves across frames. Top row: motorcycle and edge map detected at time t . Bottom row: at time $t + 4$ frame.	151
5.23	Flowchart for the motorcycle detection algorithm.	152
5.24	Lab experimental setup	153
5.25	Real world model of the lanes with the camera model	154
5.26	Test images for distance estimation	156
5.27	Top view of a vehicle in neighboring lane and its relationship with host lane	157
5.28	Some sample vehicle detection results in varying road scenarios in host lane: vehicles in far-view in (a) & (b), vehicles on bright roads in (c) & (d), vehicles in the presence of shadows of trees in (e), vehicles in the presence of shadows and road markings in (f), and vehicles in the presence of other road markings in (g) & (h)	159
5.29	Possible computational framework for vehicle detection using parallel processing engines for different lane regions	165
6.1	Sensor configuration in the proposed LCA system showing front and rear cameras, C_F and C_R respectively. The front and rear RoIs are indicated as shown. The blind spots where the cameras cannot sense are given by the striped shaded areas.	168
6.2	Effect of relative distance on the risk factor	171
6.3	Effect of relative velocity on the risk factor (a) $v_1^{(t)} < 0$ (b) $v_2^{(t)} > 0$	172
6.4	Effect of relative velocity on the risk factor with $ v_1^{(t)} < v_2^{(t)} $	173
6.5	Gaussian curve representing the proposed risk function $f(\cdot)$.	174
6.6	Gaussian risk functions showing increased risk for the same distance.	174
6.7	Minimum stopping distance versus host-vehicle velocity	175
6.8	Gaussian risk functions as d_s changes.	177
6.9	Flowchart for computing the risk factor at t using state parameters.	178
6.10	Illustration of risk factor on a sample image sequence.	179
6.11	Flowchart for determining the risk factor for a lane change scenario.	180
6.12	Illustration of road scene for Case 1.	181
6.13	Plot of $d_{F_0}^{(t)}$, $v_{F_0}^{(t)}$ and $\bar{\xi}^{(t)}$ against time for an image sequence	182

6.14	Illustration of the road scene when host-vehicle EV intends to change lane to the right lane. Three target vehicles TV_1 , TV_2 and TV_3 are seen in the road scene.	183
6.15	Plots of $d^{(t)}$, $v^{(t)}$, $xi^{(t)}$ for TV_1 , TV_2 and TV_3 in case 2. $*$, Δ , $\circ$ are the plots for TV_1 , TV_2 and TV_3 respectively. The top plot gives the relative proximities of the target vehicles. The middle plot gives the risk posed by each TV . The overall risk factor $\bar{\xi}^{(t)}$ is given in the bottom-most plot.	184
6.16	Proposed state machine for blind spot monitoring using state parameters	184
6.17	Plots of $d^{(t)}$, $xi^{(t)}$ and $\gamma^{(t)}$ versus time frame.	186
6.18	Computation Architecture for Proposed Lane Change Decision Aid System	188

List of Tables

2.1	Classification of ADAS based on road safety impact and traffic efficiency impact [1]	9
2.2	Lane change assistance systems by vehicle manufacturers	33
3.1	Description of datasets that are used for evaluation	62
3.2	Detection rate using proposed algorithm	65
3.3	Comparison of computation complexity with Hough based lane detection methods	77
3.4	Comparison of computation complexity between proposed method and [2]	78
4.1	$\rho - \theta$ values of lanes in near and far-view	92
4.2	Curved lane detection using proposed technique	94
4.3	Total Hough Transform Computation Times	105
4.4	Critical Paths and Delays	107
4.5	Number of Computation Elements and Area Cost	108
4.6	Optimal Grid Resolutions for $m \times m$ Sized Edge Maps	110
4.7	Optimal Block Size b_{opt} for Image Sizes in Video Applications	113
4.8	Comparison of HT Computations	120
4.9	Comparison of computation complexity of proposed lane detection method using AHT	122
4.10	Comparison of computation complexity of proposed lane detection method using AHT and HAHT for detecting lanes in both near and far views	123
5.1	Experimental results of the distance measurement	157
5.2	Comparison of computation complexity	163
5.3	Comparison of computation complexity with appearance based vehicle classifiers	164
6.1	Blind spot state machine example	186

Abstract

Incorrect assessment of the positions and speeds of nearby vehicles will compromise road safety due to unsafe lane change decisions. Vision-based lane change decision aid systems (LCDAS) are being increasingly explored to facilitate the automatic assessment of the scene around the host vehicle. In this thesis, computationally efficient techniques that can operate under complex road scene environment are proposed for a vision-based LCDAS.

Detection of multiple lanes is challenging particularly due to varying prominence of the associated edge features depending on their perspective with respect to the camera. In this thesis, a novel method is proposed for the automatic detection of host and neighbor lanes in the near view. The proposed method involves the systematic investigation of gradient magnitude histograms (GMH), gradient angle histograms (GAH) and Hough Transform (HT) iteratively to cater for varying prominence of lane markings. Evaluation of the proposed method (referred to as GMH-GAH-HT method) using a dataset consisting of more than 6000 images in various complex conditions shows that it is capable of high detection rates of 97% and 95% for host and neighbor lanes respectively.

It was observed that the lanes in far-view are fainter and smaller in size compared to those in near view. The positions of host and neighbor lanes in the near-view are relied upon to deploy the GMH-GAH-HT method to systematically ascertain the far-view lanes. Selective processing of faint edges was introduced to enhance the robustness of the proposed technique to detect the host and neighbor lanes in the far view. A block-level HT computation process called Additive HT (AHT) is also proposed to exploit the inherent parallelisms of HT, resulting in order of magnitude speed up. The computation complexity of combining the block-level Hough spaces is also significantly reduced by introducing a hierarchical derivative of the AHT called the HAHT. In addition, the proposed method for detecting multiple lanes in the far-view is capable of estimating curved lanes by breaking them into a number of smaller straight lines.

The detection of vehicles and their proximity in the RoI is tackled next. Two main characteristics, namely, under-vehicle shadow and multiple edge symmetry of vehicles were relied upon to first establish the presence of a vehicle. The detection of the under-vehicle shadow necessitated the automatic determination of the binarization threshold for varying lighting and road conditions. The proposed linear-regression based technique was evaluated on an exhaustive dataset to confirm that it can adapt to varying illumination conditions. Selective deployment of the GMH-GAH-HT made it possible to extract multiple edge-symmetry cues so as to further ascertain the presence of vehicles. The

proposed method for vehicle detection is shown to yield a high vehicle detection rate of 95% on a test dataset with images taken under varying road, illumination and weather conditions.

The lane width in the immediate vicinity of the target vehicle was then employed to estimate the proximity of the vehicle from the host vehicle. Unlike conventional methods that rely on stereo vision and 3-D models for estimating proximity of the detected vehicles, the proposed method has been shown to work with 2-D images resulting in a notable reduction in computational complexity. Verification against ground truth confirms that the proposed method is capable of estimating the relative distance of a target vehicle from the ego vehicle with an accuracy of 1m in the near view and 4m in the far view.

Unlike 4-wheel vehicles like cars and lorries, detection of motorcycles cannot rely on prominent under-vehicle shadows, clearly defined edges and symmetry signatures. This motivated the development of a novel method to detect the tyre region of the motorcycles. The proposed method is adaptive to local illumination conditions by examining the intensities immediately adjacent to lane markings. Systematic exploration of the surrounding areas around the tyre region was also carried out to further strengthen the identification process. An early evaluation of the proposed technique shows promising results in complex scenarios.

Existing LCDAS mainly focus on the blind spot detection aspect of the lane change decision aid process, as comprehensive assessment of the 360° scene around the vehicle is still a rather complex and computationally demanding process. In this work, a two-camera system consisting of front and rear facing monocular cameras has been employed to establish a near 360 degree field of view. The proximity of vehicles surrounding the host vehicle and their speeds were incorporated into a Gaussian risk function for estimating the risks posed by different vehicles in the front and rear views. A state machine was also introduced to monitor the blind spot region by combining the risk information of vehicles in front and rear RoIs. Finally, the proposed techniques lend well for compute-efficient realizations and simulations on real video sequences show that the integrated framework can be deployed to deterministically evaluate the risks associated with lane change maneuvers.

Glossary

ADAS	Advanced driver assistance systems
AHT	Additive Hough Transform
far-view	View of road scene in the image that is farther away from host-vehicle
front view	Road scene as seen from the front camera
GAH	Gradient Angle Histogram
GMH	Gradient Magnitude Histogram
HAHT	Hierarchical Additive Hough Transform
host lane	Lane in which the host vehicle is moving
host-vehicle	Vehicle in which the on-board cameras are setup
HT	Hough Transform
LCA	Lane change assistance
LCAS	Lane Change Assistance System
LCDAS	Lane Change Decision Aid System
near-view	View of road scene in the image that is closest to the host-vehicle (lower most 1/8-th of the image)
neighbor/neighboring lane	Lanes that are adjacent to the host lane, either on the right or left
rear view	Road scene as seen from the rear camera
region of interest	lane region which is monitored continuously
relative proximity	Relative distance between target vehicle and host vehicle
relative velocity	Relative velocity between target and host vehicle
risk factor	Risk posed by target vehicle to the host vehicle
target vehicle	Vehicle in the lanes that are nearest to the host vehicle
Ego-vehicle	Host vehicle with LCDAS
LDW	Lane Departure Warning
RANSAC	RANdom SAmple Consensus
SVM	Support Vector Machine
HOG	Histogram of Oriented Gradients

C H A P T E R 1

Introduction

1.1 Motivation

Lane change is a high risk maneuver and is one of the main causes of accidents on highways in most countries. The driver needs to continuously monitor different lanes, both in the front and rear of the vehicle, to detect any approaching vehicles. Any miscalculation and mis-judgment of the speeds and distances of the approaching vehicles can lead to incorrect decisions to change lane, resulting in accidents. Therefore, an advanced driver assistance system (ADAS), which aids the driver in monitoring the road scene around the vehicle and making correct decisions before changing lanes, can improve road safety. Such ADAS systems are called Lane Change Decision Aid Systems (LCDAS) or Lane Change Assistance Systems (LCAS). Recent studies show that an effective LCDAS is of high impact on road safety. Though some commercial solutions are available in the market, they are either meant for high-end luxury cars or have limited functionality or both. Therefore, there is a need to develop cost-effective solutions that can be deployed for mass production. One of the ways to meet this objective has been the increasing use of vision sensors or cameras for various ADAS solutions. Most current ADAS (including the limited LCDAS) employ active sensors like radar. But with advances in VLSI technologies and miniaturization of CMOS transistors, CMOS cameras are not only becoming increasingly cheaper but pervasive too. This pervasiveness and availability of cheap cameras make them an ideal choice for applications like LCDAS wherein the knowledge of surround scene is necessary. However, the use of cameras for detecting vehicles and locating their positions in the road scene has its own challenges because of the varied nature of the road scene. These challenges in developing robust and

computationally efficient image processing and computer vision solutions, that can be deployed for lane change assistance systems, have been the motivation for the proposed research in this thesis.

1.2 Research Objectives

This research aims at developing embedded computing techniques for vision-based lane change decision aid systems. It is envisaged to devise image processing solutions that aid the driver in monitoring the entire road scene around the vehicle, i.e. the front, rear and blind spots of the host vehicle, using monocular cameras. Using this information of the target vehicles around the host vehicle, it is aimed to develop techniques that help the driver to take an informed decision to change lanes. This research aims to design computationally efficient techniques for the core functional components of an LCDAS, which include region of interest extraction, vehicle detection, vehicle proximity estimation and situation/risk assessment. Architecture-aware algorithms will be proposed that can lend well for realization on embedded computing platforms.

1.3 Research Contributions

The major research contributions of this thesis are summarized below:

1. A detailed literature survey of existing techniques for the different functional blocks of lane change assistance systems is conducted. The applicability of the different sensor technologies in the different functional blocks of an LCDAS is investigated to establish the significance of vision-based sensors for LCDAS.
2. Computationally efficient techniques for detecting host and neighbor lanes in near-view of the host vehicle are proposed. Considering that both host lanes and neighboring/adjacent lanes form the regions of interest in an LCDAS, techniques are proposed to address the challenges in detecting these lanes.
3. A dataset of real-world road scene image sequences is created for the evaluation of the proposed techniques. The dataset is generated to capture different road scenarios under varying illumination conditions on Singapore highways and urban roads using a mountable high-definition (HD) resolution camera with a frame rate of 25 frames per second.

4. A thorough evaluation of the proposed near-view lane detection technique is conducted using the dataset of image sequences. The detection rates for detecting the host and neighbor lanes in different scenarios are analyzed.
5. Techniques are proposed and evaluated to detect lanes in far-view of the host vehicle. The proposed techniques use the lane information from near-view and detect the presence of lanes in far-view.
6. In order to address the computational challenges of the proposed far-view lane detection techniques, computation architectures are proposed that exploit inherent parallelisms of the proposed image processing techniques. A detailed performance evaluation of the proposed computing techniques is presented to show the drastic reduction in computational complexity that is achieved by the proposed techniques.
7. Novel techniques are proposed to detect vehicles and motorcycles in lanes. The proposed techniques are shown to detect vehicles in the presence of local variations in the road scene like shadows of trees, illumination changes etc.
8. Techniques to estimate the relative proximities of the vehicles from the host vehicle are proposed. The proposed techniques use the lane information as a cue to estimate the relative proximities using monocular cameras.
9. After detecting the vehicles and their relative proximities, a novel risk assessment framework is proposed that determines the risk involved in changing lanes. Risk assessment in single and multiple vehicle scenarios is proposed. A state machine is also proposed to detect vehicles in the blind spot areas.
10. Finally critical conclusions are drawn and the future research directions are proposed.

1.3.1 Research Publications from This Work

Following is the list of publications by the author of this report that were published in various international conferences and journals:

Journals

1. R. K. Satzoda, S. Suchitra, T. Srikanthan and S. Supriya, "Hierarchical Additive Hough Transform for Lane Detection", *IEEE Embedded Systems Letters*, vol. 2 (2), pp. 23-26, June 2010.

2. S. Suchitra, R. K. Satzoda and T. Srikanthan, "Exploiting Inherent Parallelisms for Accelerating Linear Hough Transform", *IEEE Transactions on Image Processing*, vol. 18, No. 10, pp. 2255-2264, October 2009.
3. R. K. Satzoda, S. Suchitra and T. Srikanthan, "Parallelizing the Hough Transform Computation", *IEEE Signal Processing Letters*, vol. 15, pp. 297-300, 2008.

Conferences

1. S. Suchitra, R. K. Satzoda and T. Srikanthan, "Identifying Lane Types: A Modular Approach", *IEEE 16th International IEEE Annual Conference on Intelligent Transportation Systems - ITSC'13*, Oct 2013, (Accepted).
2. S. S. Shantanu, R. K. Satzoda, S. Suchitra and T. Srikanthan, "Additive Hough Transform on Embedded Computing Platforms", *IEEE 56th International Midwest Symposium on Circuits and Systems*, Aug 2013, pp. 1196-1199.
3. S. Suchitra, R.K. Satzoda and T. Srikanthan, "Detection and Classification of Arrow Markings on Roads using Signed Edge Signatures", *IEEE Intelligent Vehicles Symposium - IV'2012*, pp. 796-801, June 2012.
4. R.K. Satzoda, S. Suchitra and T. Srikanthan, "Robust Extraction of Lane Markings Using Gradient Angle Histograms and Directional Signed Edges", *IEEE Intelligent Vehicles Symposium - IV'2012*, pp. 754-759, June 2012.
5. R. K. Satzoda, S. Suchitra, T. Srikanthan and J.Y. Chia, "Vision-based Vehicle Queue Detection at Traffic Junctions", *7th IEEE Conference on Industrial Electronics and Applications*, Singapore, pp. 85-90, July 2012.
6. S. Suchitra, R. K. Satzoda and T. Srikanthan, "Selective Extraction of Curvilinear Edges using Gradient Angle Histograms", *International Conference on Communications and Signal Processing*, Kochi, Kerala, pp. 358-362, Feb 2011.
7. S. Suchitra, R. K. Satzoda and T. Srikanthan, "Accelerating CORDIC for Hough Transform", *International Symposium on Integrated Circuits*, Singapore, pp. 167-170, Dec 2009.
8. R. K. Satzoda, S. Suchitra and T. Srikanthan, "Gradient Angle Histograms for Efficient Linear Hough Transform", *IEEE International Conference on Image Processing (ICIP)*, pp. 3273 - 3276, Cairo, Nov 2009.
9. S. Suchitra, R. K. Satzoda and T. Srikanthan, "Unified CORDIC-based Processor for Image Processing", *Proceedings of 15th International Conference on Digital Signal Processing*, pp. 343-346, July 2007.

In Preparation

1. R.K. Satzoda, S. Suchitra and T. Srikanthan , “Local-illumination Invariant Vehicle Detection Techniques”, *IEEE Intelligent Vehicles Symposium*, 2014.
2. R.K. Satzoda, S. Suchitra and T. Srikanthan , “Vision-based Lane Change Assistance System”, *IEEE Intelligent Vehicles Symposium*, 2014.
3. R.K. Satzoda, S. Suchitra and T. Srikanthan , “Multiple Lane Detection with Varying Edge Prominence”, *IEEE Intelligent Vehicles Symposium*, 2014.

1.4 Organization of the Thesis

The rest of thesis is organized in the following way:

- **Chapter 2:** A detailed literature survey is conducted in Chapter 2. Existing literature on the different functional aspects of lane change assistance systems are reviewed in detail. The computation algorithms that are employed in these functional blocks and the effect of sensing technologies on the efficiency of these functional blocks are investigated in this chapter. Existing LCDAS solutions are reviewed to understand the open challenges in this research area.
- **Chapter 3:** This chapter addresses identifying host and neighbor lanes in near-view of the host vehicle. The challenges involved in detecting these lanes are outlined and computationally efficient techniques are proposed to robustly detect the host lanes and neighbor lanes, in this chapter.
- **Chapter 4:** In this chapter, we propose techniques to detect host and neighbor lanes in far-view. Techniques are proposed to detect both straight lanes and curved lanes in far-view. Computationally efficient techniques to compute Hough transform using Additive Hough Transform (AHT) and Hierarchical AHT are proposed and evaluated in this chapter.
- **Chapter 5:** The main objects of interest in an LCDAS are the vehicles. In this chapter, adaptive compute efficient techniques are proposed to detect vehicles in lanes and also find their proximity to the host vehicle. Techniques are also proposed to detect motorcycles in lanes. The proposed techniques are thoroughly evaluated in varying road scenarios to check for their robustness.
- **Chapter 6:** Lane change assistance is incomplete without a decision making system that can aid the driver in making a decision to change lane. In this chapter,

a novel decision making system is described and evaluated. The risk in making a lane change is quantified using a mathematical model that depends on the states of the vehicles surrounding the host vehicle.

- **Chapter 7:** Finally, conclusions are drawn and future directions in this research are proposed in this chapter.

C H A P T E R 2

Literature Survey

In this chapter, we conduct a detailed survey of different functional and technological aspects of lane change assistance systems (LCAS) or lane change decision aid systems (LCDAS). We present a detailed review of core functional blocks of LCDAS like lane detection, vehicle detection, situation assessment etc. A survey of existing solutions, that provide either complete or partial functionality of LCDAS, is then presented. We conclude this chapter with a summary of the observations based on this study, leading to the chapters that follow. In most literature that is available, lane change assistance refers to lane change decision aid, i.e. a system that can aid the driver to take a decision on whether it is safe to change lane. Therefore, in the rest of the chapter and the thesis, lane change assistance is synonymously used as lane change decision aid. Similarly LCAS and LCDAS are interchangeably used.

2.1 Advanced Driver Assistance Systems for Lane Change Assistance

Over the past decade, there has been an increasing trend in incorporating Advanced Driver Assistance Systems (ADAS) in automobiles [3]. There have been various market research reports, which forecast the increasing need and investment into ADAS [3–9]. One of the major reasons for this increasing trend is also because of the increasing number of vehicles on roads, and therefore the need for driver and traffic safety [3]. Consequently, major automotive manufacturers like Mercedes, BMW, Volvo, Nissan, Continental etc. have been investing heavily into the research and development of active safety systems

or ADAS [7]. These systems include Adaptive Cruise Control (ACC) [10, 11], lane departure warning (LDW) [12], blind spot detection (BSD) [13, 14], night vision, park assist systems, collision avoidance, pedestrian detection, lane change assistance etc. [6, 9, 15]. There are various kinds of classification of these systems reported in literature [1, 10, 16] and one of the ways [16] is listed below:

1. Driver information systems: Route guidance systems, navigation systems, global positioning systems, traffic symbols detection systems can be classified as driver information systems as they gather and provide information to the driver, that help the driver in making decisions while driving.
2. Driver warning systems: These include systems like lane departure warning, lane change assistance, blind spot warning and collision warning. These systems warn the driver of any potential danger. Unlike information systems, these warning systems focus more on *safety* and also possibly provide corrective actions, that can help the driver in avoiding any accidents.
3. Intervening systems: These systems directly intervene in the actual movement of the vehicle by taking over vehicle control from the driver in case of emergency. Adaptive cruise control (ACC) is one such system which reduces or increases the speed of the vehicle based on its proximity to the vehicles in front and rear.
4. Integrated passive and active safety systems: These are mechanical safety systems that get activated either during or before a crash occurs. Automated emergency braking systems, seat belt pre-tensioners, automated air-bag release etc. are examples of such systems.
5. Fully automated systems: These form intelligent vehicle systems. They can be considered as one step beyond driver assistance. They operate without a human driver and take full control of the vehicle. Google self-driving car [17], which underwent test drives recently in 2012 is an example of these systems.

The acceptability of driver-less fully automated intelligent vehicles is still being questioned due to various factors like cost, legislation rules, dependability of systems, additional infrastructure etc. However, driver assistance systems are already available as commercial products. Until recently, these ADAS systems were either in research labs or in high-end luxury and executive vehicles only [3, 10]. However, this is changing now as most automotive manufacturers are now investing into affordable systems for high volume market segment [3]. [6, 18] present global market reviews of various ADAS systems and provide a list of commercially available ADAS systems and their manufacturers.

Some of the commercially available systems are lane departure warning systems, adaptive cruise control system, park assist systems, autonomous emergency braking systems etc. [19–21] that are manufactured by companies like Continental, BMW, Mobileye etc. This shift towards entry-level vehicles is primarily because of advances in cheap and reliable sensor technologies like vision sensors, and increasing R & D in low-cost and efficient processing technologies.

These ADAS applications also have different levels and types of impact on the driver and the other road users. Golias et al. in [1] presented an impact oriented classification of the driver assistance systems. They provide a detailed analysis of the kind of impact and the amount of impact each ADAS has on the road safety and traffic efficiency. Each ADAS system was marked as having high or low impact on road safety and traffic efficiency. Table 2.1 shows the impact-based classification that is presented in [1].

TABLE 2.1: Classification of ADAS based on road safety impact and traffic efficiency impact [1]

Traffic efficiency impact		Road Safety Impact	
		High	Low
	High	ACC, Lane change & merge, Collision avoidance, Vision enhancement	Automated transactions, Platooning Real-time traffic & traveler information
	Low	Automatic stop and go, Speed control Obstacle and pedestrian detection Intersection collision warning Integrated navigation, electronic mirror Driver identification, Hands-free & remote control, Driver vigilance monitoring, Driver health monitoring, Road and lane departure	Navigation routing, Parking & reversing aid, Tachograph, Alerting systems, vehicle diagnostics

It can be seen that four applications fall under high impact category in terms of both road safety and traffic efficiency, and lane change figures in that list. Other surveys on ADAS have also predicted lane change assistance to be an important and integral component of driver safety systems in future cars. Frost & Sullivan, one of the leading market research companies, in their 2008 market research [9] on ADAS predicts lane change assistance to be incorporated by major automotive companies by the year 2018. Though other high impact ADAS applications like ACC, collision avoidance, lane departure warning [22–24], pedestrian detection [25–27] etc. have been comprehensively researched upon, LCAS is still a relatively new research area [28]. In recent years, various aspects of lane change assistance are being researched upon by a few select groups [2, 28–32]. A survey of the different aspects of these studies, their salient features and challenges will be described next.

2.1.1 Need for LCDAS

Improper and unsafe lane changes are one of the top reasons for road accidents in most countries [33]. This is because, before making a lane change maneuver, the driver performs various physical and mental actions to take the decision to change lane [31, 34]. According to Chovan et al [35], “If the lane change is legal, the driver determines if conditions are favorable for a lane change. The driver checks mirrors, checks the blind spot, scans ahead, looks to the far-adjacent lane, and assesses the roadway for any limitations to lane changing. If the driver decides the lane change can be successfully completed, he or she signals the intent to change lanes with turn signals. By applying a steering input followed by a counter phase steering movement, the driver then executes the lane change maneuver.” Therefore, the driver needs to be completely aware of the surrounding vehicles and continuously monitor them, especially those moving in the lane where he intends to move into. This includes, monitoring the side-view and rear-view mirrors and checking the blind spots by physically looking over his shoulder. Also, the driver mentally estimates the risk involved in changing lane at that time instance by estimating the speeds and distances of the approaching vehicles [34]. This requires a high degree of alertness from the driver. Therefore, a system that can monitor the vehicle surroundings and aid the driver in making decisions to change lanes, can help in improving the safety of the driver and the other road users [1]. This is also reflected in the global safety systems technology road-map from Frost & Sullivan [9], which puts lane change assistance systems as an integral part of vehicles in the next decade.

2.2 Functional Blocks in LCDAS

A survey of the existing literature on LCDAS shows that depending on the functionality of LCDAS, different techniques and functional cores are proposed [30, 36]. According to [30], these core functional blocks include lane detection & tracking, vehicle detection & tracking, vehicle proximity and speed estimation, and situation assessment, as shown in Fig. 2.1. In addition to these modules, [2, 30] also talk about detecting driver intentions to change lanes. According to [30], driver intention detection is an additional component but an LCDAS must perform the above functional steps. Also, according to the ISO standards on LCDAS [36], driver is assumed to have made an intention to change lanes and an LCDAS is expected to check for any potential dangers in changing lanes.

These functional blocks are used in different combinations to realize different LCDAS [30, 37]. This primarily depends on the functionality of the LCDAS and the different sensors used in it. For example a radar based blind-spot monitoring system does not

FIGURE 2.1: Core functional blocks in a lane change decision aid system.

require blocks like lane detection and front/rear vehicle monitoring. However, a vision-based blind-spot monitoring in [38] requires lane detection block. Also, there are different kinds of techniques, that have been proposed for each of the functional blocks, many of which are generic algorithms that can be deployed in any ADAS including LCDAS. For example, techniques for lane detection have been proposed in various papers [39, 40] etc. for different ADAS applications like LDW, collision avoidance etc. Some of these techniques, when directly applied for lane change assistance may not yield the robust results for LCDAS; some may not even be directly applicable for LCDAS.

In the forthcoming sections, we review the different existing techniques for each of the functional blocks, their advantages, disadvantages and applicability in the context of lane change assistance.

2.3 Lane Detection

Lanes give the primary regions of interest for this application, primarily because approaching vehicles are assumed to be on the lanes. Therefore, this is one of the primary functional blocks in an LCDAS. Since lane detection is a common operation in many ADAS applications like lane departure warning, collision avoidance etc., there is a lot of literature on lane detection, which can be used for any of the ADAS applications. LCDAS has some additional requirements too, which will be reviewed in this section.

2.3.1 Sensors for Lane Detection

A survey of the literature shows that lane detection is best performed using vision sensors or cameras [41, 42] as compared to other active sensing technologies like radars, lasers and lidars. According to [42], though radars are good for measuring distances of objects like vehicles, they cannot be used to determine the lane in which the vehicles are detected. According to [2], though active sensors perform well in certain situations like rural areas, snow and rainy situation, vision sensors perform well in a wide variety of situations. Active sensors can resolve road boundaries in rural areas but fail on

multilane roads, where vision sensors are handy. Also, trajectory forecasting which is needed for applications like lane change assistance [2], cannot be done with active sensors alone. Another advantage of vision sensors, according to [2], is that vision sensors can be used to determine accurate information without the need of additional infrastructure like GPS, whereas radar systems would require GPS information for accurate sensing of the scene around the vehicle. In some works [41], the use of radars is demonstrated for lane detection, though it is concluded that the use of cameras is better and more straight forward to detect lanes. For example, in [41], the feasibility of using radars to estimate the course of the road is studied. It should be noted that the lanes are not detected but the course of road is estimated using *physical* structures like road borders. In [43], a ladar (laser radar) based lane detection is proposed but this also determines the boundaries of the lanes and not the lane markings. In addition to these systems, [44, 45] propose multi-sensor fusion methods for lane detection, wherein radars and vision sensors are used to detect the lanes. In these multi-sensor systems, the main task of lane detection is still accomplished primarily using the visual cues from cameras. The radar sensor information is used as supporting information, primarily to detect the road boundaries. Corporate organizations like Delphi and Continental have also indicated the use of vision-based sensors as the primary sensing technology for lane detection and tracking in [46] and [47]. According to the market research survey by Frost & Sullivan in [9], CMOS based vision sensors are vital components in ADAS applications like lane departure warning, lane change assistance etc., where lanes need to be detected. Based on this survey, we now review vision-based lane detection techniques.

2.3.2 Vision-based Lane Detection Techniques

[2, 39, 40, 48–70] are few of the works reported in literature. However, there are still open challenges in detecting lanes and lane markings. This can be inferred from the fact that there are over 40 IEEE journals and 200+ conference papers published on this topic in the last three years time. This is because of the varied environment of the road scene that poses a huge challenge in robustly detecting the lane markings.

In [48], the lane detection methods are broadly classified into two classes. The first class relates the detection of the lane region using the changing intensity distribution along the region of a lane, whereas the second class considers directly the spatial detection of lane characteristics. In the first class of methods, segmentation of the lane region is proposed using homogeneity in features like color [71, 72] and texture [73]. In the second class of lane detection methods, the lane is extracted by detecting the lane markings, which bind the lanes. In order to do this, different lane characteristics are used to first

robustly detect the lane markings and then define the lanes using the lane markings [2, 48].

The first class of lane region detection methods include techniques proposed in [71–73]. In [71], lane detection is considered as a two-class classification problem where the R, G, B values of lane pixel is used as a feature to classify lane from non-lane pixels. However, color is not an effective feature for defining the lane pixels because of the variations in the road scene due to various factors like cloud cover, tar color changes, atmospheric conditions etc. [48]. Color based methods would involve additional computations to normalize the color information to adapt the color information to these varying conditions [72]. Other techniques [73] employ texture as a feature to train classifiers and detect lane regions. However, these techniques are primarily image segmentation methods and could be an overkill in terms of computations, when structured roads with lane markings are to be analyzed.

The second class of methods described in [48] cater to marked urban roads/highways with painted lane markings. In most structured roads, either highways or urban roads, lane markings are strong indicators on the road surface which can be used to detect the lanes. This class is further divided into two kinds of approaches, based on the method of describing the lane characteristics. These approaches are either feature-driven approaches or model-driven approaches. In feature-driven approaches, lane features are extracted, localized and combined using meaningful characteristics. Model-driven approaches rely on the features of lanes (usually detected from feature-driven approaches) and employ deformable templates and pre-defined models on the lane features to define the lanes. According to [2], both these approaches are used together in most works to robustly detect lanes in images. Model driven approaches require the features detected by feature driven methods to conclusively fit a pre-defined lane model [2]. Some of the recent techniques [2, 39, 40, 49–70], which fall in this category, will be reviewed now to understand their applicability in an LCDAS.

Since feature driven approaches detect features like edge, color of lane markings etc. and aggregate them to detect lane-like features, various kinds of image processing techniques are used to detect the lane features. Some of these include edge detection [74], Hough transform [75], morphological operations [74], different filters like Gabor filters [62], steerable filters [2] etc. In [40, 50], Cheng et al. use color information of the lanes and their motion to extract lanes. It was shown that lane detection takes at least 0.7 seconds which does not suit real-time applications like lane change assistance. Similarly, intensity along with the pixel orientation is used in [51] to detect the lane markings. An intensity based technique is employed in [37] where two white lines are detected to

confirm lanes. However, these methods are prone to mis-detections owing to variations in the intensities in the road scene.

Various works have used lane edges and their grouping to detect the lanes [52, 63–66, 76]. For example, in [52], Wu et al propose a lane detection method using left and right edge detection masks, which help in identifying possible candidates that satisfy straight lanes. These candidates are confirmed as lane markings using a fan-scanning detection, in which a series of conditions need to be satisfied. This method is dependent on the correct identification of candidates using the proposed masks. If the candidates are incorrectly identified, the actual lane markings are missed out during the fan-scanning step of the algorithm.

Similarly, Hough transform is used in different ways in [63–66, 76] to group edges into lanes. In [63], Saudi et al detect lanes using Randomized Hough Transform, which is a variant of the linear Hough Transform that can be used to detect lines in an image [74, 75]. The computation complexity of Hough transform is usually high but this has not been addressed in [63]. In [66], a statistical Hough transform is presented to detect lane markings on images that need to be warped using inverse perspective mapping (IPM). This Hough transform method is applied directly on the image intensities and not on the edge pixels. Though this reduces the dependency on robust edge detection, the computational complexity increases as stated in [66]. According to [66], this method is computationally effective if it is used along with a lane tracking algorithm. Hardware centric optimized lane detection methods using Hough transform are presented in [64, 65]. A hardware implementation of Hough transform based lane detection is presented in [64]. In [65], a small set of line candidates is considered on which Hough transform is computed, thereby reducing the memory requirement to store the accumulation space of the Hough transform. This is done by detecting ridge features in the image which are pointing towards a previously detected vanishing point. Therefore, if the lane markings are faint or if the vanishing point is not correctly detected in the previous frame, there could be errors in the lane detection process.

In some other methods, the road scene is divided into two parts - near view and far view. Lanes in near view regions are assumed to be straight, whereas the lane markings can be either straight or curving in far view [52, 76]. In such methods, a combination of feature based approaches and model based approaches is used to detect the lane. For example, Gabor filters are used in [62] to detect the road markings in near view of the camera and road models are then applied to detect the lanes in the far view of the image. A steerable filter based lane detection method is proposed in [2] where three separable Gaussian based steerable filters are used to detect the lanes. A maximizing function is proposed using the steerable filters which helps to select the lane features.

This method is applied directly on the image intensities, and involves setting specific sized kernels that are maximized in specific angular directions. As stated in [2], this method was applied to get lane positions with known approximate angular positions of the lanes. Computationally, this method involves multiple kernel based operations and post processing steps including thresholding which could be image dependent. Another near lane and far lane detection model is presented in [76] with cameras fixed on rear-view mirrors for lane change assistance. Edge detection and average image intensities are used to detect lane markings in [76].

There is another set of feature-based lane detection methods that use machine learning techniques and inverse perspective mapping to detect lane features. A robust lane detection and tracking method for challenging road scenarios is proposed in [39]. A machine learning approach on image segments obtained from the inverse perspective mapping (IPM)[54] of the input image was employed to detect the lanes. An cascaded artificial neural network (ANN) was used to identify possible lane markings in the image followed by RANSAC for outlier removal. In addition to these, a line grouping approach is used to narrow down on lane markings. Considering the number of operations that are performed to detect lane markings, this method can be computationally complex. According to [39], the cascaded network to detect lane markings itself takes nearly 25ms in complex cases. The accuracy was reported to be around 80%. Borkar et al. in their recent work in [53] present another IPM and RANSAC based lane detection method. This method is primarily proposed for night images and involves an IPM followed by operations like adaptive thresholding for binarization, Hough transform computation and template matching to identify lane markings. These are then sent to RANSAC for outlier removal. This method operates at 0.8 seconds per frame on MATLAB simulations and an accuracy of 87% to 98% is achieved. The effectiveness of this method relies heavily on the detection of the lane markings. If they are not detected properly in the first step, the other steps in the method will fail to extract any lane markings. In [58](2012), Gopalan et al. propose another learning-based approach to detect lane markings, which does not require predefined road model. They employ an outlier boosting classifier algorithm to achieve a performance of 15 frames per second on 4GHz processor. This method involves various Haar-like classifiers [77] that are applied on road scenes to detect lane markings, resulting in high computational complexity. A RANSAC based method is implemented in [59](2011), in which image binarization is performed and the the binarized image is subjected to RANSAC, which fits a road model on the binarized pixels.

In addition to the above, there have been various other attempts to detect lanes effectively using model-based approaches [48]. Employing deformable templates is a commonly used method to detect and extract lanes [78]. Mathematical models are defined to describe different curvatures of lanes and when a lane is detected, the shape of the

detected lane is made to fit/match with the deformable model. These templates include equations of lines (for straight roads) or second-order quadratic equations or parabolas for curved roads. [2] lists the different models that are usually used to fit the lane marking signatures to determine lanes in an image. A comprehensive summary of lane detection methods that were published before 2004 are listed in [2]. The lane features and the models used for each of the lane detection methods are summarized in [2]. Stereo vision based techniques are also proposed for lane detection [54, 60]. In such methods the stereo vision concepts are used to remove outliers.

2.3.3 Multi-lane Detection

A review of the existing lane detection methods also shows that a large majority of the work is done for single lane scenario, which is ideal for applications like lane departure warning and lane keeping. However, in an LCDAS, detecting the host lane is not sufficient. The adjacent lanes are also needed in an LCDAS. A few works like [2, 52, 58, 67–70] address multiple lane detection. For instance, Hough transform and IPM are used to detect multiple lanes in [67] but this work does not address the robustness issue in detecting the lanes from Hough transform computation. A accumulator voting scheme called the parametric transform for multi-lane detection is proposed in [68]. However the technique is more suitable for autonomous vehicles as the method also integrates control systems to determine the parameters for multi-lane detection. [69] also employs Hough transform and IPM to detect multiple lanes and rely on strong edge candidates of lanes for accuracy. In real road scenes, where the conditions are variable, the accuracy of this method can be greatly affected, which is also the conclusion in [69]. Moreover, this method takes 48ms on an Intel Core Duo processor to process each frame. A more recent work on multi-lane detection is presented in [70], wherein a multi-object Bayes filtering method is used to detect multiple lanes. A series of kernel based operations are first used to detect lane markings, which are then sent to a multi-object Bayes filter and an adapted k-means algorithm, which remove the outliers. This method shows an effective detection rate of 89.7% and fails in cases where the lane markings are poorly visible. It fails differentiating guard rails, bike paths etc. from actual lane markings.

In addition to detecting lanes, tracking of lanes is also performed to increase the robustness [2]. In this step, tracking algorithms are used to predict the lane position using the history information of previously detected lane markings [2, 30]. This prediction value is used in two ways. Firstly, the predicted value is used to predict the possible position of lanes in the current frame. This reduces the computation cost. Next, the predicted value is used to verify the measured lane position in the current frame. This improves the robustness. Two commonly used filtering algorithms are Kalman filtering

and particle filtering. Kalman filtering or its variants are used in [2, 30] for tracking lanes. The complexity of Kalman tracking in [2, 30] depends on the overall functionality of the system and the different control parameters of the system. For example in [2, 30], the vehicle controls like steering angle and the yaw rate, the lane curvature and acceleration are used as control parameters for prediction using Kalman filters. These control parameters are also simple constants like those used in [67], in which case, the computation complexity is reduced. Similarly, particle filters are used in [66] for lane tracking.

2.3.4 Hough Transform for Lane Detection

The Hough transform (HT) is commonly used in computer vision applications to detect both regular and irregular shapes [74]. As shown in the previous sub-sections, linear Hough transform is commonly used for lane detection. The linear HT transforms an edge point at image coordinate (x, y) in the edge map into a sinusoidal curve in the Hough space indexed by $\rho - \theta$ using the following equation [75]:

$$\rho = x \cos\theta + y \sin\theta, \quad (2.1)$$

where ρ is the length of the perpendicular of a line passing through (x, y) and subtending an angle θ with the x -axis.

The HT computation using (2.1) involves trigonometric functions and these computations pose a bottleneck for high-speed implementation of the HT. A number of hardware architectures for the computation of HT have been proposed in [79–93]. In [79–82], CORDIC, a hardware efficient trigonometric engine [94], is used to compute the HT as its outputs can be directly mapped to the HT equation in (2.1). As the edge map is read serially, the HT is computed for every edge pixel using iterative, pipelined and high-radix CORDIC engines in [80], [82], and [81] respectively.

Another approach to compute the HT has been to replace multiplications with serial addition of a constant as shown in [83, 84, 92], which reduces the hardware complexity. The serial addition architecture in [83] adds $\cos\theta$ and $\sin\theta$ as x and y coordinates increase. In [84], distributed arithmetic (DA) [95] is used to add precomputed $\cos\theta$, $\sin\theta$ and $\cos\theta + \sin\theta$ serially as the bits of x and y coordinates are read for every edge pixel. In [92], a constant offset is added to the HT for $\theta = 0$ to determine the HT for subsequent angles in the Hough space.

[89, 91] describe systolic array architectures for HT computation. While [89] employs a scan line processor based systolic implementation using two-dimensional routing networks, [91] describes a systolic array architecture that uses the slope-intercept method [96] to compute the HT. Another method to reduce computations is using the gradient based O' Gorman's HT [97], wherein the HT is computed only for the gradient angle at that point and not all the angles in the Hough space. [93] implements a hybrid architecture for gradient based and conventional HT computation.

Most existing HT computation methods are not parallelized to compute the HT of several pixels simultaneously because they are constrained by serial reading of the edge map itself. The parallel implementations either parallelize the angle space but not the pixel coordinates, or employ extensive mesh arrays and pyramidal processors. Parallelism has also been explored for HT computation in [81, 83, 85–88, 98]. The architectures in [81, 83, 98] divide the Hough space into different angle spaces and compute the HT of an edge pixel coordinate for all these angle spaces in parallel. This parallelism however still requires the edge map to be read serially in a raster scan fashion. [88] illustrates a parallel multiresolution HT computation method, which has pyramidal parallel processors that compute the HT for different resolutions of the input image. [85–87] illustrate reconfigurable mesh array based parallel engines which usually have high hardware complexity.

2.4 Vehicle Detection

While lanes form the regions of interest, vehicles are the objects of interest in a lane change assistance system. Detecting vehicles in the road scene and estimating their distances from the host vehicle is usually the next critical step in an LCDAS. In order to do this, different techniques are employed based on the kind of requirements and also on the type of sensors used. Unlike lane detection operation, which usually can be performed on images using vision-based sensors only, vehicle detection can be done using various kinds of sensors like radars [99–101], vision sensors [78, 102–120] and lidars [121–126]. Some works like [121, 123, 126–128] present multi-modal approaches to detect vehicles using multiple sensor technologies like vision and radar or vision and lidars [127–131].

2.4.1 Vehicle Detection using Active Sensors

Unlike lane detection, active sensors like radars are extensively used for vehicle detection. Active sensors detect the vehicles and give physical quantities like distance and

speed directly, without using any additional computation resources [78]. This is because electromagnetic waves like radio waves are used to detect the presence of physical obstructions like vehicles. In fact, the most common sensor technology used in ADAS applications like lane keeping, collision avoidance, adaptive cruise control is using active sensors [132] such as radar-based systems in [99, 100, 128–130, 133], laser-based systems in [121–123, 134, 135], and acoustic-based systems in [136]. Automotive companies offering lane change assistance solutions use radars as the primary sensing technology (see Section 2.9.2). The ease of detecting vehicles using active sensors is however overshadowed by various factors like cost of sensors, ease of installation, interference issues etc. These factors are hence being questioned by automotive manufacturers for deploying active sensors in low-cost ADAS applications.

In [78], Sun et al compare the different kinds of sensors for the purpose of vehicle detection. They show that though active sensors are widely used for vehicle detection (e.g. [99–101] etc.), there are also physical and computational challenges in using such sensors. Firstly, most active sensors are expensive. Next, the silicon area cost of active sensors is major concern, which also adds to the cost of the sensor [137]. Active sensors in general have low spatial resolution and slow scanning speed. Active sensors like radars also suffer from interference issues, especially in road scenes where there are multiple vehicles [2, 42].

Though active sensors detect the presence of an obstruction easily in varied weather conditions [2, 78], the research community is questioning the data specificity issue that arises in using active sensors. This is explained in [101], where vehicles are being tracked using radars. Gunnarsson et al. in [101] quote data association as a significant difficulty in radar-based systems, i.e. the task of handling uncertainty regarding the origin of the detections is a major concern. This leads to association hypotheses which need to be generated to identify the correct sensor output. In an application like lane change assistance, with multiple vehicles in the same scene, the number of association hypotheses can be significantly huge. Therefore, additional computation models need to be formulated to extract the correct sensor outputs and remove the duplicate and irrelevant data. This issue is also discussed in [99] wherein techniques are proposed to determine the centroid of the vehicle because of multiple radar detections of the same vehicle.

The effect of range and field of view of radars on vehicle detection is discussed in [38, 46]. There are two kinds of radars, long range and short range. While short range radars have wider field of view (FOV), their longitudinal distance is limited and the case is vice versa with long range radars. In LCA system proposed in [138], both kinds of radars are employed to cover as much FOV as possible. However, using multiple radars not only increases the cost of the system, it also introduces the data association and clustering

problem as discussed above [101], which would require additional computational models to resolve the correct radar data.

Considering all these factors and the recent developments in the field of vision sensors, it is widely accepted that the future lies in vision-based integrated ADAS [9, 19, 38, 46, 137]. Various organizations are considering the feasibility of vision sensors for different driver assistance applications. For example, one of the objective of Information and Communication Technologies (IST) 6th Framework Program by the European Union is to demonstrate the feasibility of alternative sensor technologies like cameras for collision avoidance systems [137]. Though vehicles employing active sensors like radar are currently commercially available [36], there has been an increasing trend towards having passive sensors like optical sensors or cameras, as a complimentary or sometimes an alternative solution to active sensors. Multi-modal sensing of vehicles using multiple sensor technologies like radars and vision sensors is also being explored [127–131]. This is because cameras give additional capability of vision to these systems, which brings us to the next section, where we review the emerging use of passive sensors like cameras for vehicle detection.

2.4.2 Vehicle Detection using Vision Sensors

In the previous subsection, we showed the applicability of active sensors for vehicle detection. We discussed their advantages and disadvantages. According to market surveys and literature survey on vehicle detection, passive sensors like vision sensors are increasingly being used in ADAS applications for vehicle detection [2, 9, 30, 36, 47, 78, 101, 137, 139]. One of the main reasons why passive sensors are more attractive than active sensors is the low cost of these sensors like cameras [38, 78, 137]. Recent advances in semiconductor technologies and decreasing feature size of CMOS transistors has made vision-based sensors or cameras a cost efficient sensor technology option for low cost and efficient embedded systems. Reduced costs and miniaturization of cameras is also enabling 360° surround vehicle sensing using multiple cameras [30].

Unlike active sensors, additional computations are required to extract vehicle signatures when vision sensors are used to detect vehicles [78]. However, the data collected from cameras does not lead to data association and interference issues faced by active sensors [101]. Images from cameras give a complete picture of the scene as seen by the driver. Therefore the objects in the scene can be more deterministically found in images as compared to radar based sensing where data specificity is a common issue [101]. Moreover, cameras can provide data like the number of target vehicles and the type of vehicles.

A lot of research [78, 102–120] has been done because vehicles are the main targets in all automatic driver assistance systems. Vehicle detection also forms the key step in lane change assistance. In LCA systems, vehicle detection is done by on-board cameras. This section surveys vehicle detection techniques employed by on-board cameras, i.e. other methods like detecting vehicles from aerial images or from traffic signals and car parking lots are not covered in this section.

A detailed survey of on-road vehicle detection is presented in Sun et al’s survey paper on vehicle detection [78]. Majority of the vehicle detection techniques follow two main steps. The first step is called hypothesis generation (HG) and the second is called hypothesis verification (HV). As indicated by the names, the hypothesis generation step locates the possible vehicles in an image. These locations are then sent for verification in the second step. The various vehicle detection methods will be reviewed based on this classification.

2.4.2.1 Hypothesis Generation

In order to locate possible vehicle signatures during hypothesis generation phase, three kinds of methods are used. The first kind are called knowledge based methods, wherein the knowledge of the vehicle features is used to hypothesize the vehicle signature. These known parameters are symmetry [104, 105, 140, 141] of vehicles, color [106, 107], shadow [108, 142–146], corners and edges [109–113, 141], texture [114, 115] and headlights [147, 148]. Symmetry based methods like [108, 140, 142] depend on the symmetric views of the vehicles. In [140], HOG-like features [149] are used to detect the symmetry of the vehicles. Techniques in [104, 105] use the symmetry measure arising from the intensity of vehicles. According to [78], the presence of homogeneous areas is critical in computing the symmetry measure from intensity. Therefore, these methods are sensitive to noise. In methods like [105, 140], the symmetry measure is further used as a descriptor in a learning algorithm to verify the vehicle signature and reduce false negatives. However, in these methods, the whole image is scanned using a bounding box to locate the vehicle descriptors. A more recent symmetry based method is described in [141] wherein edge symmetry, instead of intensity symmetry, is used to detect the vehicle signatures.

Color is another cue that is used for detecting vehicles [106, 107, 150]. These methods rely on color information to differentiate between vehicle and the road color. Some of these methods like [150] detect headlights using color information to detect vehicles. Such techniques are computationally simple and effective in detecting vehicles in the night. However, detecting headlights during daytime is a more challenging task using color information. Techniques proposed in [106, 107] transform the color spaces into

higher dimensional spaces and classify whether a pixel is a vehicle pixel or a road color pixel. This information is also fed into a learning network for further confirmation.

Shadows of vehicles provide useful clues for their detection [108, 111, 143–146, 151]. It was found that the under-vehicle shadow is distinctly darker than the other road areas surrounding the vehicle. This property is used in [108, 111, 143–145, 151] as a cue to locate a possible vehicle in the road scene. However, the main challenge in these techniques is identifying the shadow because the intensity of the shadow is dynamic. The intensity of the shadow depends on the illumination of the image, which in turn depends on weather conditions. It also depends on the local surroundings of the vehicle. There is no systematic way to choose appropriate threshold [78] and therefore, the thresholds cannot be fixed. In [111, 151], an adaptive technique is presented to set the binarization threshold for shadow detection using the distribution of road pixel intensity values. Free road surface, i.e. road surface without any vehicles was first segmented using techniques like edge detection and entropy, and then the distribution of the road pixel intensities is modeled as a Gaussian curve. The binarization threshold was set based on the mean and standard deviation of the distribution of the road pixel intensities. Though this method effectively hypothesizes the vehicles in the road scene, the challenge is to detect the free road surface. If the road surface is detected incorrectly, the threshold may be set incorrectly leading to false detections. In addition, the local variations of the image scene are not considered in this method. In [143], vehicle shadows are used to detect vehicles where the shadows are detected using edge detection. Since the shadow is darker than the road surface, it will create an edge. However, the threshold for edge detection is set based on pre-determined value in [143], which is not adaptive to the varying illumination conditions of the road scene. Similarly, a vertical bottom-up vertical scanning method for under-vehicle shadow is described in [144]. This method also depends on fixed thresholds to determine the intensity change of under-vehicle shadows. Another such pre-determined threshold based under-vehicle shadow detection method is proposed in [145, 146, 152]. In [152], a preceding vehicle tracking and detection system, which includes road area finding, vehicle footprint extraction, and vehicle bounding box extraction, is proposed using vehicle shadow and symmetry. This method detects the footprint of the vehicle inside the road area to avoid the false detection of delineators.

Edge information is another commonly used feature to detect vehicles. Vehicles invariably have strong edges and techniques proposed in [109, 110, 113, 141] take advantage of these edges to locate vehicles in an image. The strength of edges, presence of vertical and horizontal edges in specific positions, projection of edge pixels, symmetry of edges, oriented edges etc. are some of the ways in which the edges of vehicles are used to locate the vehicles. For example, vertical profile of edge maps is created to locate the left and right bounds of vehicles in [109]. Similarly the orientation of edges is used in

[110] to deterministically locate vehicles. However, the main challenge with edge based methods is that the noise in the image scene can also create strong edges. For example shadows create strong edges and therefore vehicle edges must be sieved out of the noisy edges created by the shadows. Edge projection based methods like [109] are affected by such noise leading to false positives. In [113], Betke et al propose an edge based vehicle detection method in which vertical and horizontal profiles of vehicle edges are used to generate vertical and horizontal vectors. These vectors are used along with the aspect ratio of the vehicles to detect vehicles. This method however is dependent on too many pre-determined values. [141] combines symmetry and vehicle edges as a way to detect vehicles.

Other knowledge based methods include [114, 115] which employ texture patterns as priori knowledge to detect vehicles. Entropy measures, co-occurrence matrices etc. are used in this regard to quantify the texture of vehicles. Gabor filters are used in [153–155] to detect vehicles using the directional properties of Gabor filters. In these methods, Gabor filters are applied on input images to generate feature vectors, which are then classified using classifiers like SVM etc. Another knowledge based method is described in [156] which used edge histograms. Vehicle headlight detection [147, 148] is also classified as knowledge based method for hypothesis generation because headlights are a known information about vehicles. These methods are particularly useful during night time when headlights are prominently visible and other features like edges, texture etc. are not so evident.

The next kind of HG methods is called stereo-based methods [78]. In these methods, stereo vision is used to locate vehicles. Vehicles are distinctly above the road surface and hence they will have depth information as compared to other features on the road surface like shadows etc. Therefore, disparity maps and inverse perspective mapping of the images can help to locate objects that are not on the same height as the road surface. Some of these techniques include [103] which employ IPM to locate vehicles. However, these techniques involve the more complicated setup of stereo cameras as compared to monocular cameras.

The third kind of HG methods is called motion-based methods. In these methods [29], optical flow based techniques are used to locate moving objects on road surfaces. The optical flow of gray level intensities is monitored to located vehicles. Computationally these methods are more complex than the first two kinds of HG methods.

2.4.2.2 Hypothesis Verification (HV)

This is the verification step to confirm the presence of a vehicle in the locations hypothesized in the HG step. Therefore the input to this step is either the location from the HG step or a region of the image as identified by the HG step. In order to perform this verification, two kinds of methods are usually seen in literature [78]. They are template-based methods and appearance based methods. Template-based methods rely on pre-defined patterns or models for vehicles, whereas appearance-based methods often use machine learning algorithms to classify the vehicles from non-vehicles.

A survey of template-based HV methods shows that specific patterns of edges [116], edge distributions, symmetry of edges [141] etc. are used to verify the presence of a vehicle. In [116], edge patterns of the vehicle are used to perform HV. This method looks for specific edge transitions combined with on-board radar information to conclude the presence of vehicles. In [117], two separate generic 2D models for the appearance of obstacles (vehicles) are applied to detect vehicles in near-view and far-view. These models were based on edge symmetry and corner points. Although this method proved successful, the challenge is in the selection of the model when the distances are unknown, which are provided by other sensors like radars. In a more recent work in [141], edge symmetry is used in HV step to confirm the presence of the vehicle. This method relies on horizontal scanning of the edge map and a symmetry measure is computed using the edge pixels on every scan line. Local variations in the vehicle edges can directly affect the detection accuracy in this method. Some of these methods also use vehicle components to detect vehicles. For example, license plates and rear windows are detected and their presence is used to narrow down on the vehicle position in [112]. Another commonly used signature is the presence of a U shape to detect the vehicles [111]. Though the template based methods are computationally less complex than the next kind of HV methods (appearance based methods), the robustness of template-based methods depends on the features being correctly detected in specific patterns. In order to improve the accuracy of the vehicle detection process, appearance based HV methods are proposed. In [157], a vision-based preceding vehicle recognition method is proposed, which has the capability of recognizing a wide selection of vehicles. A multiclustered modified quadratic discriminant function (MC-MQDF) is proposed which is capable of estimating the complex distribution from different possible appearances for preceding vehicles. Though this method gives a good detection rate of over 95% and real-time computation times, these results are only for the hypothesis verification step. The efficiency of this method will be affected if the hypotheses are incorrectly generated.

There are a number of appearance based HV methods. Considering that vehicle detection is a two-class problem, appearance based methods learn the characteristics of a

vehicle from a set of samples that have vehicles and non-vehicles. Classifiers like support vector machines (SVMs), Bayesian networks, linear regression, neural networks etc. are used to verify vehicle signatures. SVMs are used as classifiers in many HV methods [154, 158, 159]. In [154], Gabor filters are used to describe the vehicle features, which are then sent to SVM to verify the presence of a vehicle. PCA-based methods are used in [109, 119] to detect global features of the vehicles, which are then sent to classifiers like neural networks [119] or local orientation coding (LOC). These global features are extracted from input datasets and the accuracy of these methods depends on the exhaustive nature of the datasets [78]. In [140], Bayesian decision networks are used for HV, wherein HOG-like features [149] are used to define the vehicle signatures. The challenges in these methods are defining the appropriate descriptors for the vehicles that give maximum inter-class separability, identifying the appropriate classifier and defining optimal decision boundaries so that there are minimum false positives and negatives. Accuracy of these methods also depends on the classifier training dataset, i.e. the more exhaustive the training dataset, the better is the classification accuracy. Also, in these methods, the computation costs can escalate because a sliding window is usually used to capture the vehicle signature in an image. Depending on the scales of the vehicles that are being detected, the number of such sliding windows can also increase. The sizes and the number of these sliding windows also play a role in the accuracy of these methods.

In summary it can be seen that vision sensors are either replacing radar sensors or being used as a complementary technology to radars in multi-modal systems to detect vehicles [78]. Knowledge based methods for hypothesis generation using features like edges, corners, shadows etc. are the most commonly used techniques. This is because of low computation complexity. However, robustness and adaptability to the varying road scene are two main challenges. The thresholding in most of these methods is pre-determined and not adaptive to varying road scenes leading to mis-detections. Therefore, a strong hypothesis verification (HV) step is often used on the candidates obtained from HG step. Appearance based HV methods that involve classifiers like SVM etc. give robust results in complex scenarios but their computational cost is considered to be high [78]. On the other hand, the second kind of HV methods, i.e. template based methods, involving edge symmetries, edge patterns, intensity patterns and symmetries are computationally more attractive than appearance based methods but robustness is still a challenge. Computational constraints on these techniques are critical for on-board vehicle detection [78], which is necessary in applications like LCDAS. Another important observation from this review is that majority of the work on camera based vehicle detection is limited to cars and 4 or more wheeler automobiles. There is limited research [160–164] on vision based motorcycle detection. In the context of lane change assistance, motorcycles are important objects of interest. Of the few references on

motorcycle detection, Wei et al in [160, 162] propose techniques that are suitable for motorcycle detection using on-board cameras, which is of interest in LCDAS. In [160, 162], Wei et al detect vehicles using SVM classifiers after a preprocessing stage that gives possible motorcycle candidates. A detection rate of nearly 90% is reported in [160, 162]. In [161], license plate are detected first and motorcycles are inferred from the properties of license plate, which is a convoluted way of detecting motorcycles. Also, this method is proposed for overhead traffic surveillance cameras. Similar motorcycle detection method is proposed in [164]. In [163], techniques are proposed using background subtraction and hence not suitable for on-board cameras where background keeps changing. Therefore, detecting motorcycles is another major challenge in vehicle detection operation of an LCDAS.

2.5 Estimating Vehicle Proximity

In an LCDAS, estimating the proximities and speeds of the approaching vehicles relative to the host vehicle is the next step after detecting the presence of vehicles in the regions of interest (lanes). In the previous section, we discussed how active sensors like radars can detect and estimate the distance of the vehicles at the same time due to the nature of the technology itself. Though active sensors give these quantities directly, it was explained earlier that there are challenges that these sensors face in *accurately* estimating the distances of vehicles. These challenges arise because of multiple data associations due to the inherent properties of the radars [101]. Another issue with distance estimation using active sensors is that they cannot determine the accurate distance of the vehicles on curved lanes [78]. It was also shown in Section 2.4 that passive sensors like cameras are increasingly being used as either an alternative sensing technology or complementing the active sensors in multi-modal fashion [2]. However, distance estimation using vision sensors has its own set of challenges and techniques are proposed in [165–167] etc. to estimate vehicle proximity using vision sensors.

Stereo-vision is one of the commonly used methods to determine the vehicle proximity. Given the camera calibration parameters, the disparity maps and inverse perspective mapping of images from stereo cameras can be directly used to infer the distance of the vehicle [78, 103]. However, stereo-vision based methods are computationally more expensive as compared to monocular cameras. Therefore, techniques have been proposed to estimate the vehicle distance using monocular cameras. But this needs more work than stereo-vision based distance estimation. Most of the monocular camera-based methods depend on a known property of the road scene or the camera calibration parameters to extract the depth information from a 2D image plane. [165–170] describe

different approaches to vehicle distance estimation using monocular cameras. A distance measure formula is also proposed in [152] which computes the distance to the preceding vehicle using a camera calibration and vanishing point detection process. The distance measure is dependent on the correct detection of the vanishing point. Another method using a ranging monocular camera model is proposed in [165] to estimate the distance of the vehicle. It depends on the initial calibration of the camera, its angular and spatial positions in the vehicle. Similarly, another camera calibration based method is presented in [167]. Tsai et al. in [166] propose a distance estimation of vehicles in night time, by determining the distance between the two headlights, which is assumed as a constant value. This method however depends on accurate detection of headlights. [168] describes a vehicle proximity estimation method based on the license plate of the vehicle. The size of the license plate is used as a constant and its size in the image plane is used to determine the proportionality constants for estimating the distance of the vehicle. This method relies completely on the accurate detection and chopping of the license plate signature. The height and width of a detected vehicle is used to determine the distance of a vehicle in [169]. The accuracy of this method depends directly on two variable factors, i.e. height and width of the vehicle, which are extracted from the live image scene. Therefore the scope for error is high.

Based on the review of these methods, it is observed that most of these techniques depend either on the height of the camera position or fixed objects like license plates, headlights etc. or on multiple variable factors like the height and width of the target vehicle. Also, in all these methods based on monocular cameras, the detection of the target vehicle and its features, i.e. size, height and width, is the most critical operation in determining the distance of the target vehicle accurately.

2.6 Blind-spot Monitoring

According to the ISO standards [36], one of the functional requirements of a lane change assistant is monitoring the blind spots of the host vehicle for any approaching vehicles. In fact, most works on lane change assistance until recently focused only on blind spot monitoring [36]. Some existing techniques, especially related to vehicle detection, are proposed primarily for blind spot monitoring. Some of these were discussed in the related sections previously. Here, we present some recent works that propose blind spot monitoring systems. [38, 171–175] are some techniques for monitoring blind spots using sensor data from vision sensors and radars. [36] also lists some commercial systems that are currently available in the market which perform blind spot monitoring. In most of the existing blind-spot monitoring methods, the sensors are installed under the side

mirrors and these sensors monitor the blind spots of the vehicle continuously. Some the recent blind spot monitoring methods are reviewed in this section.

[36] describes radar-based blind-spot monitoring systems that are commercially available in the market. Most of these systems use 24 GHz radars that are fixed on the side mirrors of the vehicles and pointed towards the blind-spot zones. These radars detect approaching vehicles in blind-spot areas and alert the driver during unsafe lane changes. However, both commercial entities [36] and academic groups [38] are exploring vision sensors as possible sensing technologies for blind spot monitoring. In [38], a vision-based blind-spot monitoring system is proposed that uses part-based features to detect and track vehicles moving the blind-spot areas of the vehicle. Different parts of the vehicle, when viewed in the side-view angle are detected and associated with each other to detect the presence of a vehicle in blind spots. Thereafter, these part-based features are continuously tracked. The proposed method shows a detection time of nearly 2 seconds per frame, which is not suitable for applications like LCDAS. Another vision-based blind spot monitoring system is proposed in [171] wherein a warping of the input image obtained from side-view cameras is performed to get a warped image of the front view of the vehicle in blind spot area. The count of the vehicle edges is then used as a descriptor for an SVM network to identify the presence of vehicles in the blind spot area. Similar approaches are used in other works where specific features of a vehicle in the blind spot area are used to detect the presence of a vehicle in blind spots and these features are tracked. In [172], different features are used for different timings of the day, i.e. vehicle shadows are used for day time and presence of headlights is used for night time vehicle detection in the blind spot areas. Techniques in [38, 171, 172] rely on the models of the vehicles and hence there could be mis-detections in these cases. A motion based method for object detection is proposed in [173] which track the vehicle motion and thereby differentiate the vehicle from other surroundings. In all these methods, there is an additional cost that is involved in installing the sensor systems pointing at the blind spot areas of the vehicle. The techniques involved in blind spot detection are similar to those used for vehicle detection and tracking. However, these systems do not give the complete functionality of an LCDAS.

2.7 Driver Intention Detection

Driver intention detection is another operation, that is drawing attention of some research groups. As part of driver intention analysis, techniques are proposed in [176–182] to detect the intention of driver to change lanes. This intention is detected by analyzing

the various movements of the driver and the vehicle control parameters. We briefly survey some of the techniques used to detect driver intention. Some of the existing methods such as [176] use the vehicle controls to determine the driver intention. Depending on the vehicle model parameters like steering angle, axle distance etc., a Bayesian network is proposed that learns the state of the vehicle and predicts if the vehicle is moving towards a lane change. Similarly, in [181], a Hidden Markov Model (HMM) is used to identify driver's intention to change lanes using vehicle control parameters like steering angle, steering angle velocity, car velocity, car acceleration, yaw rate and pedal positions. In [179], Doshi et al. propose a computer vision based technique to understand the driver's visual search patterns and the effect of external stimulus on these patterns. These are then extended to understand if driver intentions to change lanes can be detected. In [182], Morris et al. provide a detailed analysis and review of some recent techniques in driver intention detection for lane change assistance. They also propose a holistic intent detection framework which uses vehicle dynamics data along with data from multiple sensors like radars and cameras, to detect the intention of the driver. Detailed on-road tests were also conducted to determine how early the proposed system can actually detect the lane change intent of a driver.

This particular functional block is not a functional requirement in a lane change assistant but an early indication of the driver intention to lane change certainly adds to the safety quotient of a driver assistance system.

2.8 Situation Assessment for Lane Change Decision Making

Situation assessment in an LCDAS refers to the analysis of the surrounding objects (vehicles) of the host vehicle to determine whether a lane change is safe. The information about vehicle positions, speeds and lane positions are used to assess the situation. Research in deriving lane change maneuver decisions is rare [183]. There exist only a few works which give a quantifiable risk assessment for lane change assistance. [138, 184–189] propose some safety and alarm measures to aid in taking a decision to change lanes. In [188], the situation assessment is performed by finding the lead and lag distances between leading (vehicle in front of host vehicle) and lagging (vehicle in the rear of host vehicle) vehicles. These distances must satisfy a minimum reference distance criteria to conclude that it is safe to change lanes. This is a simplified model based on only the distances of the approaching vehicles. Similarly in [189], a safety criteria is proposed based the distances, acceleration and velocities of the host vehicle and the surrounding vehicles. This method was tested on a simulation data only assuming that the different velocities,

accelerations and distances are available. However, in a real scenario, this is not usually the case, wherein all the different states of all vehicles are accurately available. In [29], a lane change decision aid is proposed which continuously monitors the approaching overtaking vehicle. The optical flow based state vectors of the approaching vehicle are used to warn the host vehicle if a lane change is safe. This method does not consider the states of the vehicles in front of the host vehicle for lane change decision, which has been identified as a challenge in [29]. In [190], a general criticality criterion for risk assessment is presented and applied to lane change maneuvers. According to [190], “no interpretable and easily understandable combination of the criterion is known” for LCDAS. They propose a criteria based on the relative distance and time to crossing (TTC) but this is still a simplified model looking at the blind spots and approaching vehicles only. Schubert et al. have proposed a more sophisticated Bayesian network based situation assessment model in [30, 138, 183, 187]. Given the various vehicle parameters like distances and velocities, which they obtain from vision and radar sensors, a Bayesian probability tree is proposed to determine the probability of safe lane change. Lane positions, target vehicle positions, target vehicle speeds, the host vehicle state parameters are combined in different ways to design the Bayesian probability tree. Depending on Baye’s rule, the probability measures are applied for various conditions that must be satisfied to determine if the lane change is safe or unsafe. The Bayesian network provides a probability score for three states, i.e. *keepplane*, *lanechangeleft* and *lanechangeright*. Depending on the probability score to each of these states, a safety measure S is recommended. In addition to the Bayesian network, Schubert et al. also propose some metrics in [187] to measure the vehicle parameters for lane change assistance. These include deceleration to safety time (DST) and normalized lateral position (NLP).

Other lane change decision making methods are presented in [34, 191]. In [34], different lateral support systems with a special emphasis on sensor technologies, detection and safety assessment algorithms are reviewed. However, no practical implementations of systems for lane change assistance are reported. One of the few reported implementations of an LCA is described in [191] as part of the European Integrated Project PREVENT. There are other LCDAS decision methods [145] where the decision is made based on a fixed distance model. In other words a fixed distance on the image is marked and if a vehicle is detected in that marked area, a safety decision is triggered. It can be seen that these methods do not assess the risk based on the actual dynamic scene. For example, in [145], a pre-determined region in blind spot areas and rear of the vehicle is continuously monitored for any vehicle signature. If a vehicle is detected in a *dangerous* area, the driver is alarmed of an approaching vehicle.

Some works [192–194] propose the actual maneuver path that would be required for a safe lane change, given the situation assessment based on the various objects (vehicles)

in the scene. Some of these techniques are primarily meant for automated intelligent driverless vehicles and are not designed for driver assistance. Also, according to the functional requirements of the ISO standards on LCDAS (Section 2.1.1), the primary objective of an LCDAS is to aid the driver in letting him know about the surroundings. Therefore, the actual maneuvering of the vehicle is an optional functionality in LCDAS.

2.9 Existing Lane Change Decision Aid Systems

A detailed review of the existing work on the different components of an LCDAS was presented in the above sections. It is to be noted that the most of the above techniques are independent works, which could be used for different ADAS applications. For example, lane detection techniques are proposed in [39, 40] etc., which could be used for lane departure warning, lane keeping and some of them can be used completely for lane change assistance. Similarly, most of the vehicle detection techniques reviewed in [78] were proposed for vehicle detection that could be deployed in different applications. In other words, many of these works were not specifically meant for lane change assistance. Consequently, some of these techniques may not be completely applicable for lane change assistance. For example, as noted in Section 2.3, most lane detection methods were proposed for detecting host lanes only. Therefore, they may or may not be fully applicable to lane change scenario because adjacent lanes also need to be detected for an LCDAS. Also, certain optimizations could be possible in the algorithms when a specific application is targeted. For example, all vehicles need not be detected in an LCDAS; the detection of the nearest vehicles is sufficient. In this section, we survey literature on overall lane change assistance systems, i.e. systems that provide the either full or partial functionality of an LCDAS.

2.9.1 Functional Requirements of LCDAS: ISO 17387

In 2008, a separate ISO standard was instituted to define industry standards for lane change assistance systems [36]. This standard named, as ISO 17387, defines the functional requirements of lane change decision aid systems (LCDAS) or lane change assistance systems (LCAS). The following are the functional requirements of a lane change assistant:

1. The lane change assistant should support the driver during lane change maneuvers via detecting potential hazardous situations that result due to the driver failing to monitor the area around the vehicle adequately.

2. For this purpose, the assistance function should be capable of detecting road users approaching rapidly from behind as well as detecting other road users in the blind spot of the driver's own vehicle.
3. The assistance function should operate for both adjacent lanes equally, on the driver's and the front passenger's sides.
4. Ideally, the assistance function will be available under all road, weather, and traffic conditions with approximately the same level of quality.

Therefore, the primary objective of an LCDAS is that it should support the driver during lane change maneuvers by detecting potential hazardous situations, which the driver may have missed or failed to monitor. In the list above, item (1) can be seen as the primary objective of an LCDAS and items (2) to (4) support this primary objective. Based on the objectives, the functionality of an LCDAS has been defined in various ways. According to the ISO standards [36], the following different functional configurations are possible:

- *Blind spot warning*: continuously monitor the blind spot areas of the host vehicle, which are otherwise not visible through rear view mirrors in vehicles.
- *Close vehicle warning*: alert the driver of an approaching vehicles, especially from rear, which can be potentially dangerous if the driver intends to change lane.
- *Lane change warning*: monitor the blind spots and approaching vehicles, and assist the driver in safe lane change maneuvers.

The solutions proposed in literature cater to one or more of these different functional aspects of LCDAS [36]. Different sensor technologies are employed based on the functionality of the LCDAS. Similarly, different sub-systems with varying levels of complexity are proposed to cater for the different functional configurations of LCDAS.

2.9.2 Commercial LCDAS in Market

Before surveying the existing LCDAS further, we list some commercial solutions that provide various levels of lane change assistance functionality. Table 2.2 lists the different automotive manufacturers that have deployed LCDAS in some of their high-end vehicles [36]. The LCDAS functionality and type of sensor technology used in each of the products are also listed. It can be seen that majority of these systems rely on radar technology for sensing vehicles. 24 GHz radars are usually used in these systems to detect

approaching vehicles in different areas surrounding the host vehicle. These radars are positioned at different places on the vehicle. For example, the “Side Blind Zone Alert” from GM employs radars that are fixed just under the side mirrors, whereas Audi has 24GHz radars fixed under the rear bumper of the vehicle. The effectiveness of these systems is also dependent on the detection range of the radars and minimum activation speed when the system must alert the driver. Some vehicles like Audi have a detection range of 100m but some, like Jaguar, have limited distance range of just the immediate rear and side areas of the vehicle. Systems with lower activation speed like Ford can aid the driver in crowded areas as well and hence they are suitable for lane change assistance in urban areas. In contrast higher activation speed implies they are more effective on highways and major roads. More details on the exact technical specifications can be found in [36].

TABLE 2.2: Lane change assistance systems by vehicle manufacturers

Vehicle Manufacturer	Sensors	Functionality
Audi	Radar	<ul style="list-style-type: none"> • Blind spot warning • Lane change warning • Driver side monitoring
BMW	Radar	
Mazda	Radar	
VW	Radar	
Ford	Radar	<ul style="list-style-type: none"> • Blind spot warning • Driver side monitoring
GM	Radar	
Jaguar	Radar	
Mercedes Benz	Radar	
Nissan Infiniti	Radar	
Jaguar	Radar	
Volvo	Camera	
Peugot	Camera	<ul style="list-style-type: none"> • Blind spot warning

It is noteworthy that only two manufacturers (Peugot and Volvo) employ vision sensors for lane change assistance. While Peugeot only monitors the driver’s blind spot area, Volvo has two cameras fitted under the side exterior mirrors which constantly monitor the blind spots. Both these systems are not full-scale lane change assistance systems. However, the survey of current research shows vision sensors are gaining popularity owing to lower costs compared to radars and additional capabilities that they provide which radars cannot. Companies like Mobileye [195], Continental [19] etc. are exploring vision sensors in a multi-modal approach for commercial products. Since the type of sensors determine the overall LCA system and its functionality, we first review the existing sensor technologies that are commonly used in driver assistance systems.

2.9.3 On-going Research Efforts in LCDAS

As shown in the previous sections, most of the techniques in literature present algorithms and methods for one or more generic functional blocks of an LCDAS. These functional blocks could be used for other ADAS applications also. Complete or near complete LCDAS are proposed in a few papers only like [29, 30, 76, 145, 152, 186]. In this section, we briefly review some of these works. It is to be noted that each of these papers were reviewed for specific functional components also in the previous sections.

In [186], one of the earliest LCDAS was proposed for highway scenarios. This system used a fusion of radar and vision sensors to monitor the rear and sides of the vehicle. Vehicle detection and their proximities were detected using the fusion sensor technology. However, there is no specific mention of a risk assessment strategy for lane change. In [152], a real-time vision-based preceding vehicle detection and tracking system is presented. Algorithms for lane detection, followed by vehicle detection and tracking are presented. Distance estimation technique using the vanishing point of the image is also described. However, [152] does not describe the effect for vehicles in front of the host vehicle for lane change decision aid. Also, there is no mention about risk assessment in [152]. Another vision based lane change decision aid system is presented in [145], wherein a safe/danger situation assessment is described for rear approaching vehicles. The safety decision making system in [145] is dependent on vehicle detection in a predetermined lane area. In this work also, the effect of vehicle in front of the host vehicle is not included in the risk assessment for lane change. Similar techniques are proposed in a lane change warning system described in [76]. Alonso et al describe another LCDAS for rear approaching vehicles only in [29]. Vehicle detection is performed using motion-driven tracking followed by a risk assessment that is computed based on the state vectors of the approaching vehicles. The safety alarm is set to pre-determined value in [29]. In [30], Schubert et al propose one of the latest works that can be considered as a complete LCDAS. In [30], forward and rear looking cameras and radars both, are used to detect lanes, approaching vehicle (from front and rear) and then a Bayesian network is used to assess the situation around the host vehicle to give a safety decision for lane change. The techniques proposed by Schubert et al were generic methods, which were later customized for an LCDAS. This work is one of the latest in terms of tackling LCDAS as one single system. However, efforts are still continuing to make the system as computationally lean as possible by reducing the number of sensors.

2.10 Summary

In this chapter, a comprehensive review of recent studies, techniques and technologies related to lane change assistance systems was presented. The survey showed most LCDAS have common functional blocks like lane detection, vehicle detection, vehicle proximity estimation and situation assessment [30]. A detailed survey of each of the functional blocks and their constituent techniques was conducted. In this section we summarize some key observations that were made from the literature survey.

A survey of the different functional blocks showed that vision sensors or cameras are being increasingly used in ADAS applications. The reducing cost of cameras with advances in VLSI technology, resulting in cheap yet sophisticated and high resolution cameras, is leading to this trend [78]. In an application like LCDAS where multiple vehicles need to be detected from multiple RoIs (lanes), vision sensors provide additional information that other sensor technologies cannot provide [2].

Next, a detailed review of existing lane detection techniques shows that vision sensors are apt for detecting lanes on both structured and unstructured roads [2, 78]. There are some common challenges that are being addressed in most of these techniques resulting in varying levels of robustness and computational complexity. Firstly, extracting lane marking features in complex road scenarios with shadows, illumination changes etc. is still an open challenge [2, 66]. According to [2], robust extraction of the lane features is the first step in detecting lanes using any of the methods discussed in Section 2.3. If lane features like lane edges or color or texture etc. are not extracted robustly, the rest of the algorithm will, in high likelihood, fail to detect the lanes. Also, there is a trade off between robustness and computational complexity in existing techniques. It can be seen that some of these methods are either highly robust but computationally complex like [39, 53] etc., or computationally simple but not robust like [63–65] etc. For example the outlier boosting method in [58] shows highly robust detection results but it operates at only 15 frames per second. There are other methods [63, 64], which use simple edge features to extract lanes but their robustness can be challenged by the variable road scene. Another key observation, especially in the context of an LCDAS, is that most methods detect host lane only, whereas an LCDAS requires both host and neighbor lanes to be detected.

In the area of vehicle detection, though radar based vehicle detection and distance estimation techniques are popular in literature, visual cues are being increasingly used to accurately detect vehicles in a road scene, either independently or in fusion with radar sensors [78]. Visual information provides additional cues like exact positions of the vehicles on curved roads, types of vehicles etc. which radar-only vehicle detection

systems do not provide [2]. However, vehicle detection using vision sensors is a challenging research area, primarily during day time [78]. This is because of the variations in the road scene and varying vehicle features that are possible. Vehicle are usually first located using basic features like edges, shadows, color etc. [78]. Such knowledge-based hypothesis generation methods are more popular than stereo or motion based methods to localize vehicle candidates. Lower computational complexity is the main reason for this choice [78]. However, robustness and adaptability to the varying road scene are two main challenges. Most of the techniques available in literature involve setting appropriate thresholds, which are usually pre-determined heuristic values. This could lead to mis-detections primarily because of the varying road scene. Not only do the illumination and intensity variations change for the entire image globally, but there are variations locally around the vehicle also. For example, a cloudy day makes the whole image look darker. This variation is seen on the entire image. In addition to this variation, if there are shadows of trees or tar color changes on the road surface, they add to the local variations in the image. Therefore an algorithm needs to be highly dynamic and robust to variations in the image scene. More specifically, thresholds need to be highly dynamic and image dependent to robustly detect the vehicles signatures. Therefore, a strong hypothesis verification step is often used on the candidates obtained from HG step. Appearance based HV methods that involve classifiers like SVM etc. give robust results in complex scenarios but their computational cost is considered to be high [78]. On the other hand, the second kind of HV methods, i.e. template based methods, involving edge symmetries, edge patterns, intensity patterns and symmetries are computationally more attractive than appearance based methods but robustness is still a challenge.

In addition to robustness, vehicle detection using on-board cameras has computational challenges also. Computationally, the on-board vehicle detection method for lane change assistance (and other applications like lane departure warning) have real-time processing constraints [78]. These computational constraints being satisfied at low-cost is another research challenge in the field of vision-based driver assistance system. In addition to vehicle detection, vision-based vehicle proximity estimation also poses challenges in terms of robustness. Most vision-based vehicle proximity estimation techniques rely on accurate vehicle detection [78, 165–170]. Some of them like [165–170] depend on features of vehicles like distance between headlights etc. Therefore, computationally efficient and accurate vehicle detection is necessary for vehicle proximity estimation also.

We also surveyed techniques on situation assessment for lane change decision aid. It was seen that there is limited work on an objective formulation of risk assessment in a lane change process. Most existing techniques [184–186, 188, 189] are not adaptive to the changing surround scene and the risk is made a function of fixed parameters like

pre-defined danger regions. Few techniques like [30] give a detailed analysis of the risk assessment. However, these techniques in [30] are dependent on inputs from multiple sensors like long and short range radars, and cameras. Also, the review showed that there is a lot of literature on blind spot monitoring and preceding vehicle monitoring. The effect of vehicles in front of the host vehicle on lane change decision for assessing the safety of lane change is not explored in detail. A surround view of the vehicle using only vision sensors for lane change assistance is also not completely explored in existing literature. With increasing pervasiveness of camera sensors, an LCDAS that is completely dependent on vision sensors is an attractive alternative in terms of robustness, computational complexity and more importantly cost. Based on this detailed review of the constituent functional blocks and overall lane change decision aid systems, adaptive and computationally efficient techniques will be proposed in the forthcoming chapters to address some of the key challenges in developing vision-based lane change decision aid systems.

CHAPTER 3

Detection of Host & Neighbor Lanes in Near-view

3.1 Introduction

In a lane change assistance system, lanes form the regions of interest (RoI) that need to be constantly monitored. Unlike applications like lane departure warning which rely on host vehicle lanes only, lane change assistance also requires neighboring lanes in addition to the host vehicle lane. Therefore, lane change assistance requires detection of both host and neighbor lanes. As shown in Section 2.3, most existing lane detection methods robustly detect host lanes. However, they usually fail to detect neighboring lanes due to the challenges posed by features that form the neighboring lanes. In order to understand this, we applied one of the recent host lane detection methods, which we proposed in [196]. This method uses a customized edge detection process followed by Hough transform (HT) to detect host lane markings in near-view of the host vehicle. The proposed method takes advantage of the properties of host lanes and employs gradient angle histograms [197] to detect straight line edges in an input edge map and generates what is termed as Straight Line Edge Map (SLEM). This SLEM is then split into directional edge maps using the gradient directions. These directional edge maps are then analyzed using HT to confirm the presence of host lane markings. More details about this method are presented in [196].

Even though the neighbor lanes also show similar properties as those exhibited by host lanes listed above, this method was unable to robustly detect neighboring lane markings. This is because the prominence of the neighboring lanes are usually less prominent than the host lanes. Also, the prominence of the neighboring lanes varies as the distance of

the lanes increases from the camera in the host vehicle. By prominence, we mean the edge strength, which is more distinct in the case of host lanes as compared to neighboring lanes. Therefore, there is no one specific threshold that can be set to extract the edges of all the lane markings, i.e. host and neighboring lane markings. Host lane markings can be extracted with a higher threshold but we need to decrease the threshold, sometimes more than once, to extract the edges of the neighboring lanes. So if one threshold is set, only one set of edges will be extracted. This is illustrated in Fig. 3.1(a), which shows lanes of varying prominence. The gradient map of the image block is shown as a 3D mesh in Fig. 3.1(b). It can be seen that the gradient magnitudes of the host lane markings are higher as compared to the neighbor lane markings. Setting one threshold T_1 as shown in Fig. 3.1(b) will give only the host lanes (Fig. 3.1(c)) and eliminate the neighbor lanes. Multiple thresholds need to be set as shown in Fig. 3.1(b) to detect both the lane edges. Considering that edges are commonly used in most lane detection methods, similar challenges will be faced by other lane detection methods also.

FIGURE 3.1: (a) Image patch **I** showing lanes of varying prominence (b) Gradient magnitudes of **I** showing lanes of varying gradient magnitudes (c) Edge map **E** generated by setting $T_1 = 300$, only host lane edges are detected, (d) **E** using $T_2 = 200$, both host lanes and neighbor lanes are detected.

We also studied the limited existing techniques on multiple lane detection methods [2, 52, 58, 67–70], which were reviewed in Section 2.3. It was shown that lane feature detection step in these methods also will face the same challenges. Therefore, thresholding the gradient magnitudes adaptively based on the spatial position of the lanes is critical to robustly detect the neighbor lanes. This applies to any edge based lane detection method, and not just to the method described earlier in this section.

Another major issue arises in the process of tackling the above-mentioned challenge. Since the neighbor lane markings are usually fainter, the thresholds must be lowered in dealing with lower intensities and gradients. However, in doing so, unwanted non-lane pixels from the background and other faint intensity variations like shadows are also released. Therefore, noisy pixels or non-lane pixels are also released, decreasing the robustness in extracting the lane edges from the pool of lane and non-lane pixels. Also,

additional computations are required to extract lanes from this pool resulting in higher computational complexity.

In this chapter, a novel technique for detecting host and neighbor lanes in near-view is presented. Near-view covers the region closest to the camera in front of the vehicle (more details about near-view will be discussed in the next section). An iterative method is proposed that addresses following challenges:

- It detects the host lanes and neighboring lanes of varying edge prominence. The proposed techniques are developed to detect the immediate adjacent neighbor lanes. The same techniques can be extended to lanes beyond the adjacent neighbor lanes also (this will be discussed later in the chapter). For the rest of the chapter the terms “neighbor lanes” or “neighboring lanes” refer to the neighbor lanes adjacent to the host lanes.
- It detects lane edges with low prominence but eliminates noisy non-lane edges.
- It determines the position of host lanes and neighbor lanes.

The proposed multiple-lane detection method will be evaluated for robustness in varying complex road scenes. A computation architecture will also be presented and the computation cost of the proposed technique will be evaluated.

Nomenclature: Before we proceed, we briefly list the different terms and their meaning, that are introduced in this chapter and used in the rest of the thesis. Firstly, the vehicle in which the driver is being assisted is called the *host vehicle*. *Host lanes* refer to the left and right host lanes that form the host lane in which the host vehicle is moving. *Neighbor lanes* refer to the adjacent lanes (if present) to the host lanes. A neighbor lane can be left or right neighbor lane, depending on its position with respect to the host lane. The left and right host lanes are represented by L_{HL} and L_{HR} respectively. The neighbor lanes are represented with respect to L_{HL} and L_{HR} , in the following way. The left neighbor lane is given by L_{HL-1} . Similarly the right neighbor lane is given by L_{HR+1} .

3.2 Iterative Technique for Detecting Lanes in Near-view

In this section, an iterative technique is proposed that detects and extracts host lanes and neighbor lanes. This multiple-lane detection problem is first solved in the near-view of the camera, which is defined as the region of road scene that is closest to the host

vehicle. The distinctiveness of near-view in a road image is that lanes appear as straight lines in this area [198]. This property of lanes simplifies the lane detection process to finding straight lanes as an initial step [198]. These straight lanes found in the near-view are then used to detect the lanes in *far-view*, which is defined as the road surface seen away from the host vehicle. The lanes in far-view could be either straight or curved but they are always found to be straight in the near-view. For the rest of the thesis, the lower 1/8-th of the road surface as captured by the camera in the image plane is taken as near-view. This proportion is dependent on the camera position and orientation and can be precalibrated accordingly.

As discussed in the previous section, near lanes and neighbor lanes have different prominence. Therefore, different thresholds are required to detect these lane edges. Depending on the road scene, it may be required to dig deeper into the gradient magnitudes (which are thresholded to get the edges) to sense and extract lane edges of different prominence. In order to do this, we first describe a technique to group together edges of varying prominence. After grouping the edge pixels of similar prominence, we then propose a technique that releases each group of pixels in an informed and iterative manner, to check if that group of edge pixels has lane edges. The process of grouping edge pixels, releasing them and analyzing them for lane marking detection is described in the forthcoming sections.

3.2.1 Gradient Magnitude Histograms (GMH) for Sensing Edges of Varying Prominence

In this section, a novel technique is proposed to sense edges of varying prominence as the first step to detecting host and neighbor lanes. In order to do this, *Gradient Magnitude Histograms* or GMH are proposed to enable setting of different thresholds adaptively and generate groups of edge pixels that have similar prominence. The process of generating GMH and setting these different thresholds are described in this section.

Given an image $\mathbf{I}$, the gradient magnitudes and angles at every pixel position are computed using Sobel kernels $\mathbf{S}_x$ and $\mathbf{S}_y$, in x and y directions respectively, as follows:

$$\mathbf{G}_x = \mathbf{I} \otimes \mathbf{S}_x \quad \mathbf{G}_y = \mathbf{I} \otimes \mathbf{S}_y \quad (3.1)$$

where $\otimes$ represents convolution operation, $\mathbf{G}_x$ and $\mathbf{G}_y$ are the gradient magnitudes in x and y directions. The Sobel kernels $\mathbf{S}_x$ and $\mathbf{S}_y$ are given by:

$$\mathbf{S}_x = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad \mathbf{S}_y = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad (3.2)$$

The gradients $\mathbf{G}_x$ and $\mathbf{G}_y$ are then used to compute gradient magnitude $\mathbf{G}$ and gradient angle Θ as shown below:

$$\begin{aligned} G(x, y) &= \sqrt{G_x^2(x, y) + G_y^2(x, y)} \\ \Theta(x, y) &= \tan^{-1} \frac{\mathbf{G}_y(x, y)}{\mathbf{G}_x(x, y)} \end{aligned} \quad (3.3)$$

where (x, y) is the index of a pixel on the image plane. The gradient magnitude matrix $\mathbf{G}$ is thresholded to give the edge map $\mathbf{E}$, i.e.

$$\mathbf{E} = \mathbf{G} > T \quad (3.4)$$

where T is a threshold. Sobel gradient computation and thresholding forms the basis of more advanced edge detection methods like Canny edge detection [199].

The main challenge in detecting multiple lanes is thresholding of gradients in $\mathbf{G}$ such that lane edges of varying prominence are extracted. This implies that one single value of the threshold T is not sufficient, i.e. there must be multiple thresholds given by $T = \{t_1, t_2, \dots, t_\alpha\}$. In order to get these thresholds automatically depending on the image content, gradient magnitude histogram (GMH) is computed.

A gradient magnitude histogram (GMH) is defined as follows:

Definition: Gradient Magnitude Distribution $\mathbf{m}$ of an $M \times N$ image $\mathbf{I}$ is generated using the gradient magnitudes in $\mathbf{G}$ by binning them into k bins, each of width w_m . Therefore, $\mathbf{m}$ is given by:

$$\mathbf{m}^T = \{m_i\} \quad \text{where } 1 \leq i \leq k \quad (3.5)$$

and

$$k = \left\lceil \frac{\max(\mathbf{G}) - \min(\mathbf{G})}{w_m} \right\rceil \quad \text{and} \quad \sum_{i=1}^k m_i = MN. \quad (3.6)$$

The GMH vector $\mathbf{m}$ shows an interesting property: edge pixels of an object having similar gradient magnitudes usually group around a local peak in $\mathbf{m}$. This is illustrated in Fig. 3.2 for an image block in Fig. 3.2(a). The corresponding GMH $\mathbf{m}$ is shown in Fig. 3.2(b) for bin width $w_m = 10$.

FIGURE 3.2: (a) Image patch $\mathbf{I}$ showing lanes and other features with gradients of varying prominence (b) GMH $\mathbf{m}$ for the image patch (c) Zoomed in $\mathbf{m}$ showing clusters at higher gradients (d) Edge map from the first group $\mathbf{G}_1$ (e) Edge map from the first group $\mathbf{G}_2$.

It can be seen from the GMH in Fig. 3.2(b) that the lower bins will have large counts. This is consistent with the fact that most natural scenes has similar intensities, i.e. majority of image block has small variations in gradient magnitudes. Edges, which form a small percentage of the image, have higher intensity changes, i.e. higher gradient values. This property of edge pixels is seen in the GMH in Fig. 3.2(b) where the distribution tapers down drastically as the gradient value increases.

Fig. 3.2(c) shows a zoomed in view of the higher bins in the GMH by setting the high valued bins 1 to 3 to zero, for ease of viewing the higher low valued bins. It can be seen that gradients in the higher bins show a grouping effect. If we divide $\mathbf{m}$ into three parts $\mathbf{m}_1$, $\mathbf{m}_2$ and $\mathbf{m}_3$ as shown in Fig. 3.2, we get three groups of gradients $\mathbf{G}_1$, $\mathbf{G}_2$ and $\mathbf{G}_3$. The gradient values in $\mathbf{G}_1$ correspond to the pixels with highest intensity change (gradient) in the image block. Fig. 3.2(d) gives the edge map with edge pixels having gradients in $\mathbf{G}_1$. It can be seen that these edge pixels correspond to the lane marking with high intensity change in Fig. 3.2(a). Similarly the edge pixels released in $\mathbf{G}_2$ form the edges with lower intensity change than those in $\mathbf{G}_1$ as shown in Fig. 3.2(e).

The group boundaries in $\mathbf{m}$ when multiplied by the histogram bin width give the thresholds $T = \{t_1, t_2, \dots, t_\alpha\}$, which can be used to get edge pixels in each cluster. In order to get these groups in $\mathbf{m}$, a peaks set P is computed in the following way.

Given $\mathbf{m}$ has k bins, peaks set P is given by

$$P = \{p_j | p_j = m_i \text{ if } m_{i-1} < m_i > m_{i+1}\} \quad (3.7)$$

where $1 \leq i \leq k$ and $1 \leq j \leq \alpha_p$, α_p being the number peaks in $\mathbf{m}$. A peak of peaks set P' is defined on P by applying the same rule as above, i.e.

$$P' = \{p'_{j'} | p'_{j'} = p_j \text{ if } p_{j-1} < p_j > p_{j+1}\} \quad (3.8)$$

where $1 \leq j' \leq \alpha_{p'}$, $\alpha_{p'}$ being the number of peak of peaks in $\mathbf{m}$. The peak indices set J has the indices where the peak of peaks occur in $\mathbf{m}$, i.e.

$$J = \{j_1, j_2, \dots, j_{\alpha_{p'}}\} \quad (3.9)$$

Each j_i in the above set J is the index with respect to the indices of $\mathbf{m}$, i.e. the GMH.

After finding P' and J , the thresholds that create the groups $\mathbf{G}_i$ are found by determining the indices of the minimas between every adjacent pair of peaks of peaks in P' . So given two adjacent peaks of peaks $p'_{j'}$ and $p'_{j'+1}$, their corresponding indices from J are $j_{j'}$ and $j_{j'+1}$. The threshold t_i that separates the two groups whose peaks are located at $j_{j'}$ and $j_{j'+1}$ is given by

$$t_i = \text{index}_{\mathbf{m}}(\min(\mathbf{m}(j_{j'} : j_{j'+1}))) \times w_{\mathbf{m}} \quad (3.10)$$

where $\text{index}_{\mathbf{v}}(.)$ gives the index of the value obtained in the function with respect to the vector $\mathbf{v}$; $\min(.)$ gives the minimum value of the input elements. For a given J , the threshold matrix is computed in this way to give $T = \{t_1, t_2, t_{\alpha_{p'}-1}\}$. Each pair of (t_i, t_{i+1}) in T gives the group boundaries for a group of gradient magnitudes $\mathbf{G}_i$.

FIGURE 3.3: GMH with peaks and group boundaries indicated

This process of computing group boundaries is illustrated using a GMH in Fig. 3.3. Applying the above equations, we get $J = \{1, 40, 48, 61, 73\}$ where the peaks of peaks $p'_{j'}$ are found. These are indicated with dashed arrows in Fig. 3.3. The group boundary thresholds are given by the set $T = \{280, 430, 580, 620\}$. These thresholds are shown using solid arrows in Fig. 3.3. It can be seen from Fig. 3.3 that the proposed method divides $\mathbf{m}$ into a total of 5 groups $\mathbf{G}_1$ to $\mathbf{G}_5$. The edge maps that result from the first four groups are given as $\mathbf{G}_1$ to $\mathbf{G}_4$ in Fig. 3.4(b)-(e). It can be seen that the edge maps capture the lanes and other features with prominent edges. It can be seen in Fig. 3.4(e) that other non-lane edges are also extracted in the edge map. These edges need to be eliminated and the lane edges only must be retained. In the sections that follow, these groups in GMH are used in an informed manner to robustly extract the lane edges of varying prominence only and eliminate non-lane edges.

FIGURE 3.4: Edges extracted from the different edge groups (a) input image block, (b) edge map from MATLAB Sobel edge map generation, (c) edges from $\mathbf{G}_1$ and $\mathbf{G}_2$, (d) edges from $\mathbf{G}_1$, $\mathbf{G}_2$, $\mathbf{G}_3$ (e) edges from $\mathbf{G}_1$, $\mathbf{G}_2$, $\mathbf{G}_3$ and $\mathbf{G}_4$

3.2.1.1 Block-based GMH Generation

The distribution in the GMH was found to be dependent on the image size. If the GMH is generated for the entire $M \times N$ sized image, it was found that grouping of gradients suppresses finer features in the image. The idea that a group in a GMH corresponds to an object in the image is more controlled if image blocks are taken instead of the entire image itself. In order to have a controlled release of edge pixels using GMH, it is proposed to divide the image into blocks, each of size $M_b \times N_b$. In such a block-based scenario, the GMH was found to be more effective in generating edge maps with controlled release of pixels by making use of the algorithm described in the previous section.

This method of dividing the image into blocks and local processing of these blocks is a commonly used technique to improve robustness in image processing. This helps to get local features more robustly as compared to applying the same algorithm over the entire image globally. Techniques like contrast stretching, histogram equalization etc. also employ such local block based processing to improve robustness.

Employing this block-based method in the proposed GMH method helps in releasing edge pixels in a more controlled manner. The groups corresponding to individual objects of interest, i.e. lanes are more evident when GMH is applied in a block-based fashion. In addition, analyzing the GMH of the entire image is computationally more complex as there are too many local peaks that are formed in the GMH of the full image. Also, the block-based method will help in processing the image in a parallel fashion. More details about the computational complexity and parallel processing of the image will be discussed in the subsequent sections.

The block size $M_b \times N_b$ was heuristically found for the image data set that was used for testing, such that GMHs give reasonably clear groups for analyzing. It was found that for an image size of 720×1280 , the acceptable block size was found to be 90×128 . This is found to be approximately $1/8^{th}$ to $1/10^{th}$ of the image size. This is a one-time calibration for a particular position and image resolution of the camera. This ratio can change based on the camera setup and resolution. This ratio was determined based on the number of edge pixels captured in each block. It was found that if the block sizes are smaller, then the number of edge pixels captured in the GMH-GAH-HT engine are not sufficiently separable from noisy edge pixels that are formed by gradient changes due to non-lane features on and around the road surface. This is because a smaller block size will have lesser number of edge pixels, which form local peaks in GMH (and GAH) and result in peaks, even if they do not correspond to valid lane edges. When the block size is increased to the above mentioned block sizes, the peaks in GMH and GAH were more representative of the valid linear edges. It was seen that these peaks in GMH and GAH had lesser contribution from noisy edge pixels. Incidentally, the ratio of $1/8$ also resulted in acceptable block sizes for image sizes of 320×480 and 480×640 . For the rest of thesis, all image sizes are set to HD resolution, i.e. 720×1280 and the corresponding block sizes (wherever blocks are used) is set to 90×128 .

3.2.2 Detecting Host & Multiple Lanes using GMH

We now propose a technique that uses the edge pixel groups created in the GMH to robustly extract the host and neighbor lane markings of varying edge prominence. Multiple thresholds are set in the GMH so that edge pixels in the block are collected into

different groups, each group having edge pixels of similar gradients. Now, each of these groups in the GMH needs to be analyzed if there are any lane markings in them. We first describe the technique to extract lane markings when a GMH of a block is given. We will show how groups in GMH are released one by one and analyzed for presence of lane markings. We will then use this module in different ways or configurations to detect host lanes and neighbor lanes in near-view.

3.2.2.1 GMH-GAH-HT Engine to Extract Lane Edges

This is an iterative technique that releases the edge pixels from GMH and confirms the presence or absence of lanes. We proposed a host lane detection method in [196] in which gradient angle histogram (GAH) and Hough Transform (HT) are used together to locate host lane markings. This method in [196] does not take into account lane edges with varying prominence. We now propose an improved technique using GAH and HT along with GMH in an iterative fashion so that we extract lane markings of varying edge prominence. This lane edge extraction technique is described as follows.

For the rest of the chapter and the thesis, we will consider the block-based GMH method. Therefore, an image $\mathbf{I}$ is divided into blocks of size $M_b \times N_b$ and each block is represented by $\mathbf{I}_{b(i,j)}$, where $b(i,j)$ is the block index with i and j being the indices along the vertical and horizontal directions respectively.

For a block $\mathbf{I}_{b(i,j)}$, the GMH $\mathbf{m}_{b(i,j)}$ is divided into non-overlapping groups $\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_\alpha$, using the grouping method proposed in the previous section, i.e.

$$\mathbf{m}_{b(i,j)} = \mathbf{m}_1 \cup \mathbf{m}_2 \cup \dots \mathbf{m}_\alpha \quad (3.11)$$

where $\mathbf{m}_1$ is the rightmost group corresponding to the edge pixels with the highest gradients. For the sake of brevity, we represent $\mathbf{m}_{b(i,j)}$ as $\mathbf{m}$ corresponding to an image block $\mathbf{I}_{b(i,j)}$.

In the proposed iterative method, in each iteration (indexed by i), each of these groups $\mathbf{m}_i$ is released one-by-one starting from $\mathbf{m}_1$, the rightmost group. This is because the edge pixels of lane markings usually have higher gradients as compared to other objects in the road scene. The edge map generated from the group $\mathbf{m}_i$ is denoted by $\mathbf{E}_i$. The number of pixels in $\mathbf{m}_i$ is given by n_i and this is also the total number edge pixels released in $\mathbf{E}_i$.

In order to find out if there are any lane marking edges in the edge map $\mathbf{E}_i$ that is released in the GMH group $\mathbf{m}_i$, $\mathbf{E}_i$ is examined for straight lines that form a specific pattern. For this purpose, a technique using Gradient Angle Histograms (proposed in

[197]) is devised that computes a gradient angle histogram in an iterative fashion. From [197], a gradient angle histogram (GAH) extracts only straight line edges from an edge map that may have any kinds of edges, i.e. straight line edges, curved edges, noisy edge pixels. A histogram is generated using the gradient angles of the *candidate edge pixels* in the input edge map. Edge pixels on a straight line edge will have similar gradient angles and this is shown as a peak in the GAH. By identifying the peak in the GAH and plotting the edge pixels that contribute to this peak, the straight line edges in the candidate edge pixels are extracted. Such edge maps containing straight line edges only are termed as Straight Line Edge Maps (SLEM) in [197]. More details about this technique can be obtained from [197].

In [197], the candidate input edge map, from which the SLEM is generated, is obtained from Sobel edge map generation. We now propose an iterative method using the GMH to release edge pixels in an informed manner to check the presence of straight lines using GAH and then check the presence of lanes. If a lane is found, the release of edge pixels from GMH is stopped, and no further processing is done. Therefore, in this way, a check on the computational cost is performed at every step in the proposed iterative method.

Given a GMH $\mathbf{m}$, starting with the first group $\mathbf{m}_1$, the number of pixels in the i -th group $\mathbf{m}_i$, i.e. n_i , is checked if it is greater than a minimum value n_{min} . $n_i \leq n_{min}$ indicates a small group, which may not give meaningful or sufficient information in the next steps. If $n_i \leq n_{min}$, then $\mathbf{m}_i$ is merged with the next group $\mathbf{m}_{i+1}$, and $\mathbf{m}_{i+1}$ is again checked for the minimum edge pixel count before sending the edge map $\mathbf{E}_{i+1}$ for the next steps.

When a group $\mathbf{m}_i$ satisfies the above condition, the corresponding edge map $\mathbf{E}_i$ is generated. $\mathbf{E}_i$ is used to create two GAHs, which are called as positive GAH and negative GAH, denoted by ∇_i^+ and ∇_i^- respectively. ∇_i^+ corresponds to the histogram made from the edge pixels in $\mathbf{E}_i$, which have positive gradient magnitudes in x direction, i.e. $\mathbf{G}_{ix} > 0$. Similarly ∇_i^- is made from the edge pixels with negative x -gradients, i.e. $\mathbf{G}_{ix} < 0$. These GAHs are used to generate accumulated GAHs, denoted by $\bar{\nabla}^+$ and $\bar{\nabla}^-$ for the positive and negative GAHs respectively. $\bar{\nabla}^+$ and $\bar{\nabla}^-$ are generated by summing the ∇_i^+ and ∇_i^- with the GAHs from the groups released from GMH till the i -th iteration, i.e.

$$\begin{aligned}\bar{\nabla}^+ &= \sum_{j=1}^i \nabla_j^+ \\ \bar{\nabla}^- &= \sum_{j=1}^i \nabla_j^-.\end{aligned}\tag{3.12}$$

The properties of lanes are now used to analyze these accumulated GAHs. Since lanes are straight lines in the near-view, the edge pixels of lanes in $\mathbf{E}_i$ should show peaks in

GAHs. The reason for creating separate GAHs from positive and negative gradients is that lane markings are composed for a pair of straight line edges, where one edge is made of positive gradients, i.e. dark to light intensity change, and the other edge is composed of negative gradients or light to dark intensity change. Therefore, if there is a lane marking in $\mathbf{E}_i$, there must be a peak in both GAHs, i.e. $\bar{\nabla}^+$ and $\bar{\nabla}^-$. Moreover, the peaks must occur at the same place or angle in both $\bar{\nabla}^+$ and $\bar{\nabla}^-$. If both these conditions are satisfied, there is high probability that the edge pixels giving these peaks could be that of the lanes. In order to confirm further that these edge pixels are from the lane markings, priori information about the lanes is used. For example, the lane orientations and positions can be used as priori information to check for peaks in those angle ranges in accumulated GAHs. More details about this priori information will be presented in subsequent sections.

At this point, if $\bar{\nabla}^+$ and $\bar{\nabla}^-$ do not show any peaks in the defined angle ranges, the next group from GMH is asked. If peaks are found in the accumulated GAHs, then edge maps $\mathbf{E}_i^+$ and $\mathbf{E}_i^-$ are generated, which have edge pixels that form the peaks in the accumulated GAHs. It is to be noted that $\mathbf{E}_i^+$ and $\mathbf{E}_i^-$ have edge pixels from the current group of edge pixels released. According to the properties of GAH [197], this should give straight line edges only in $\mathbf{E}_i^+$ and $\mathbf{E}_i^-$.

If there are straight line edges in $\mathbf{E}_i^+$ and $\mathbf{E}_i^-$, the next step is to locate the exact positions and orientations of the straight lines using the Hough Transform. Hough Transform is a technique that is used to transform edge pixels in the image plane to parametric space called Hough space. For every edge pixel at (x, y) , the following equation:

$$\rho = x \cos \theta + y \sin \theta \quad (3.13)$$

is applied to determine ρ , which is the perpendicular distance from the origin to the point edge pixel at (x, y) , such that the perpendicular subtends the angle θ with the horizontal. Hough space made by ρ s and θ s is generated such that for every edge pixel, ρ s are computed for a range of angles $-90 \leq \theta \leq 90$. For a given (ρ, θ) tuple, an accumulation counter at (ρ, θ) location is updated by 1. In this way, if a straight line is present in the edge map, all the collinear points on the straight edge will result in a peak at one particular (ρ, θ) . Therefore, a peak in the Hough space at (ρ, θ) indicates a straight line with a perpendicular distance for origin as ρ , that makes θ with the horizontal axis. More details about Hough Transform can be seen in [75].

For the edge pixels in $\mathbf{E}_i^+$ and $\mathbf{E}_i^-$, Hough transform (HT) is computed using the above equation. Two Hough spaces are generated $\mathbf{HS}^+$ and $\mathbf{HS}^-$. The parameterized tuple (ρ, θ) for every edge pixel is used to update these Hough spaces. It is to be noted that the Hough spaces have accumulation information from the previous iterations and the HTs

FIGURE 3.5: Block diagram showing the functional flow of the proposed iterative method for lane detection

of edge maps of current iteration $\mathbf{E}_i^+$ and $\mathbf{E}_i^-$ are updated to the same Hough spaces. Thereafter peak detection is performed on $\mathbf{HS}^+$ and $\mathbf{HS}^-$. If (ρ^+, θ^+) and (ρ^-, θ^-) are the locations with peak heights a'^+ and a'^- in $\mathbf{HS}^+$ and $\mathbf{HS}^-$ respectively, then the following differences are computed:

$$\begin{aligned}
 \delta_\rho &= \rho^+ - \rho^- \\
 \delta_\theta &= \theta^+ - \theta^- \\
 \delta_{a'} &= a'^+ - a'^-.
 \end{aligned} \tag{3.14}$$

These peaks correspond to lanes if and only if the following conditions are met:

$$\begin{aligned}\delta_{\rho min} &\leq \delta_{\rho} \leq \delta_{\rho max} \\ \delta_{\theta min} &\leq \delta_{\theta} \leq \delta_{\theta max} \\ \delta_{a' min} &\leq \delta_{a'} \leq \delta_{a' max}\end{aligned}\tag{3.15}$$

where the minimum and maximum values of the differences are set based on the priori information of the lane markings. If these conditions are not met, the GMH is asked to release the next group. Fig. 3.5 shows the flowchart of the proposed algorithm. The computation control flow is shown using solid arrows and the data transfer and variables used at each computation step are represented by dotted arrows in the figure. We call this entire engine as *GMH-GAH-HT* engine because a combination of GMH, GAH and HT is used to extract the lane markings.

3.2.2.2 GMH-GAH-HT Engine for Robust Lane Detection

We will now show how the GMH-GAH-HT engine is used to extract host lanes and neighbor lanes in near-view. The iterative method is applied on the near-view image blocks to detect host lanes first. These host lanes are then used to detect the neighbor lanes in the next step.

Host lane detection in near-view: In order to detect the host lanes, we make use of some priori information about the host lanes. Considering that the camera is mounted at the center of the vehicle, the left host lane is searched in blocks 3 and 4, i.e. $\mathbf{I}_{b(8,3)}$ and $\mathbf{I}_{b(8,4)}$ (the block setup is shown in Fig. 3.6). Similarly, for the right host lane, $\mathbf{I}_{b(8,5)}$ and $\mathbf{I}_{b(8,6)}$, are searched. It should also be noted that the processing for host lanes starts from inward blocks towards outward blocks, i.e. $\mathbf{I}_{b(8,4)}$ is processed before processing $\mathbf{I}_{b(8,3)}$ for left host lane detection. This priori information about host lanes helps to avoid unwanted computations in the rest of the image blocks. In addition to the block positions, we also set a possible angle range for the host lanes. Assuming that the vehicle is not changing lanes (which is the case in the application scenario), the host lanes will exhibit a specific angle range only. Therefore, the angle ranges, where host lanes are expected to show peaks in the accumulated GAHs in the GMH-GAH-HT method, can be set to the following: $\theta_{\nabla hl} \in [20, 60]$ and $\theta_{\nabla hr} \in [-20, -60]$. These angle ranges are found to be wide enough to cater for the change in orientations of the host lanes due to lateral drifts of the vehicle in the host lane.

FIGURE 3.6: Illustration showing the division of input image into blocks.

In order to detect the host lanes, the edge pixel groups in the GMH are accessed from the right, i.e. from the highest gradient values to the lower gradient values. The GMH-GAH-HT algorithm is used of every edge pixel group from GMH. If lane markings are found satisfying the δ_ρ and δ_θ conditions given in (3.15), then the release of edge pixels from GMH is stopped and the host left and right lanes (L_{HL} and L_{HR}) are said to have been extracted.

If edge pixels are not found till the least prominent edge pixel group in GMH, then the next block in the near-view is processed. If the host lane markings are not detected in the last possible image block in near-view (i.e. $I_b(8,3)$), then it is assumed that either there is no host lane marking in the near-view or it is too small or ambiguous to be detected by the GMH-GAH-HT engine. In such cases, we look for history based lane information extraction to predict the lane marking position in the current frame, which we will explain in the next sub-section. Fig. 3.7 shows the algorithm for detecting host lanes using the GMH-GAH-HT engine. It can be seen that if the algorithm is set to run without the termination condition, it will try to extract lane markings from all the groups that are identified in the GMH. In some cases, computing all the groups in GMH may be necessary. However, in some cases, computing all groups may lead to unwarranted computations and the algorithm may be required to stop early. By releasing too many groups, especially the lower gradient groups, there is a possibility that more noisy and non-lane edge pixels are released resulting in increased number of computations. Therefore, the following termination conditions are set to stop the iterative method in an informed fashion:

FIGURE 3.7: Block diagram showing proposed host lane detection method in near-view using GMH-GAH-HT engine

1. The priori information of the lane markings will be used as the first indication of the termination condition. For example, host lane markings are usually more prominent. Therefore, they are expected to be in the higher gradient groups in GMH. Hence, while traversing the GMH groups from right to left, if a lane marking pair is found, the GMH iterations are terminated in the block.
2. The next termination condition is with regards to the case when faint lane markings are being extracted. This is especially the case for neighboring lane markings. In such cases, the lower gradient groups must be investigated to extract lane markings. However, releasing all the pixels in the last group also releases all the pixels in the gradient map, which is undesirable. Therefore, the final group must be further broken down to a sub-group. This is done by setting the maximum number of edge pixels released. According to literature on edge detection, the maximum number of edge pixels in an $M \times N$ sized image is approximately $0.1MN$, i.e. 10%

of the image size. Therefore, if this maximum number of pixels released is crossed, the termination condition is reached.

Neighbor lanes' detection in near-view: Given the host lanes $L_{HL}(\rho_{hl}, \theta_{hl})$ and $L_{HR}(\rho_{hr}, \theta_{hr})$, we now proceed to find the neighbor lanes in near-view. In order to do this, we use the host lane information to predict the possible positions and orientations of the neighbor lanes. We describe the method here for the immediate neighbor lanes L_{HL-1} and L_{HR+1} , but this can be extended to any number for neighbor lanes. Also, for the sake of explanation, it is assumed that there are neighbor lanes L_{HL-1} and L_{HR+1} present on either side of the host lanes, i.e. there is no road boundary on either side of the host lanes (we will comment on the presence of the road boundary later in this section). The host lane positions are used to hypothesize the neighbor lane positions, i.e. the possible locations where neighbor lanes can be found are determined using host lane positions. This is done by computing the vanishing point using the host lane positions. Given (ρ_{hl}, θ_{hl}) and (ρ_{hr}, θ_{hr}) as the ρ s and θ s that define the left and right host lanes respectively, the equations of the left and right host lanes are given by:

$$\begin{aligned}\vec{h}_l : x \cos \theta_{hl} + y \sin \theta_{hl} &= \rho_{hl} \\ \vec{h}_r : x \cos \theta_{hr} + y \sin \theta_{hr} &= \rho_{hr}\end{aligned}\tag{3.16}$$

where $\vec{h}_l$ and $\vec{h}_r$ denote the lines that pass through the left and right host lanes. Solving the above two equations, the resulting intersection point is the vanishing point $V_{pt}(x_{pt}, y_{pt})$, i.e.,

$$V_{pt} = \begin{bmatrix} x_{pt} \\ y_{pt} \end{bmatrix} = \begin{bmatrix} \cos \theta_{hl} & \sin \theta_{hl} \\ \cos \theta_{hr} & \sin \theta_{hr} \end{bmatrix}^{-1} \begin{bmatrix} \rho_{hl} \\ \rho_{hr} \end{bmatrix}\tag{3.17}$$

This is shown in Fig. 3.8. The significance of the vanishing point is that the neighbor lanes in near-view must also pass through it. This is because, all lane markings in the near-view appear to straight lines and all parallel lines in a perspective view pass through the vanishing point. This implies that the possible lane candidates which will be extracted by the GMH-GAH-HT engine must pass through the vanishing point. Therefore, this information can be used to generate the priori information for the neighbor lanes. Assuming that lanes are parallel and are of known approximate widths, we can find a relationship between the perspective orientations of host lanes and neighbor lanes. Given that the angles of the host lane are θ_{hl} and θ_{hr} for left and right host lanes respectively, the approximate neighbor lane angles can be computed as:

$$\theta_{fl} = \theta_{hl} + \Delta\theta\tag{3.18}$$

$$\theta_{fr} = \theta_{hr} - \Delta\theta\tag{3.19}$$

where Δ_θ is an approximate fixed constant for a given lane width (this is assumed as constant for a given road infrastructure). With θ_{fl} and θ_{fr} , we determine the possible ρ values for neighbor lanes using the parametric line equation. Since the neighbor lanes must pass through V_{pt} , θ_{fl} , θ_{fr} and V_{pt} must satisfy the linear equation of the Hough transform (3.13) to give ρ_{fl} and ρ_{fr} , i.e.,

$$\rho_{fl} = x_{pt} \cos \theta_{hl} + y_{pt} \sin \theta_{hl} \quad (3.20)$$

$$\rho_{fr} = x_{pt} \cos \theta_{hr} + y_{pt} \sin \theta_{hr} \quad (3.21)$$

Therefore, the neighbor lane candidates are expected to occur at (ρ_{fl}, θ_{fl}) and (ρ_{fr}, θ_{fr}) . But it must be noted that these are predicted values. Therefore we use this information for verification of neighbor lane markings that will be extracted by the GMH-GAH-HT engine. This information is also used in the history based neighbor lane prediction in case the lane markings are not extracted by GMH-GAH-HT engine. This will be discussed later in this section when we propose techniques about history based prediction. We now propose the following algorithm for detecting the neighbor lanes in near-view.

FIGURE 3.8: Vanishing point from host lanes can be used to predict possible locations of neighbor lanes.

After detecting the host lanes, the priori information for neighbor lanes is computed as discussed above. In addition to the possible $\rho - \theta$ values for the neighbor lanes, the image blocks in near-view, in which neighbor lanes can be found are also set. These image blocks are $\mathbf{I}_{b(8,1)}$ to $\mathbf{I}_{b(8,2)}$ for left neighbor lane and $\mathbf{I}_{b(8,6)}$ to $\mathbf{I}_{b(8,8)}$ for right neighbor lane. The GAH angle ranges that will be used in GMH-GAH-HT engine are also set based on the predicted θ_{fl} and θ_{fr} from (3.18). This priori information is sent to the GMH-GAH-HT engine, wherein the edge pixel points in the GMH are analyzed for lane markings.

Unlike the host lane markings, the lane marking extraction for neighbor lanes involves multiple checks in the GMH-GAH-HT engine. This is vital for robust detection of neighbor lane markings because they can be such low prominence that noisy edge pixels may also be picked up in the GMH edge pixel group. Therefore, in order to eliminate the noisy edge pixels and extract the neighbor lane markings, the following checks are

performed on the lane candidates picked by GMH-GAH-HT engine. Firstly, at the GAH stage, peaks must be found in the angle ranges defined for neighbor lanes. As shown earlier, these angle ranges of the GAH are determined from the orientations of host lane markings. After ensuring that the detected edge pixels satisfy the first condition HT is employed on the straight line edges generated from the GAH resulting in possible lane candidates given by (ρ_{fl}, θ_{fl}) and (ρ_{fr}, θ_{fr}) . These candidates are checked again using the second condition to further ascertain that noisy edges are not qualified as lane markings. In the second condition, we use the predicted values of (ρ, θ) that were computed using the host lane parameters and vanishing point. If the error between the predicted values and the detected values is in acceptable range, we consider the detected lane markings are possible neighbor lane candidates. This process is repeated till we reach the end of the GMH groups. If a lane marking is not detected using the edge groups in the GMH of the current block, we repeat the same process with the next block. If a lane marking is not detected even after processing all the possible neighbor lane blocks, we use the information from history to predict the neighbor lanes in the current frame. Fig. 3.9 shows the neighbor lane detection algorithm.

FIGURE 3.9: Block diagram showing proposed neighbor lane detection method in near-view using GMH-GAH-HT engine

FIGURE 3.10: Host and neighbor lane detection with prediction modules

Temporal Continuity in Lane Positions In the near-view, where the proposed algorithms are applied to detect lane marking signatures (both host lane and neighbor lanes), there will be cases when lane markings are not detected. This happens because of the following reasons:

1. The lanes could be dashed lane markings. Therefore, in some frames, the lane marking is seen in the near view and in next few frames, the gap between the markings is seen in the near view. When the gap occurs in the near view, the proposed algorithm will not be able to pick up any lane marking.
2. The lane markings could be split in the blocks in the near view. There will be cases when the lane marking as a whole appears in the near view but it is split across two adjacent blocks. In this case, the signatures of lanes seen in the GAH and HT are not significant enough to pick them as lanes.
3. The lane markings are missing. This is the case when there are no lane markings in near-view either due to some construction or unmarked road stretches.

In these cases, the temporal history of lane markings is used to draw lanes in the current frame. Tracking control filters like Kalman filter and particle filter have been used for lane tracking in [2]. In addition to the visual cues about the lane positions as inputs to

these filters, other sensor information like yaw angle, steering angle etc. are also used to define the controls in these filters. What we show now is a simplified tracking method based on the locations of lanes only, that are detected in the images. However, filters like Kalman and particle can also be used on the detection results obtained from the proposed techniques to predict/track based on other inputs also.

The proposed temporal history based lane prediction is based on the location of lane markings in the current frame (if detected) and an average of the locations of the previous frames. The flowcharts in Fig. 3.10 show the temporal history prediction of host lanes and neighbor lanes in near-view. Both are similar in functioning. Referring to the host-lane module, the temporal history based prediction functions in the following way. In the current frame $\mathbf{I}_i$, if a lane marking is found at (ρ_i, θ_i) , we check this location with the average values of (ρ, θ) over the last N frames. The mean values are computed using the following equations:

$$\bar{\rho}_i = \frac{1}{N} \sum_{k=i-N}^{i-1} \rho_k \quad \bar{\theta}_i = \frac{1}{N} \sum_{k=i-N}^{i-1} \theta_k \quad (3.22)$$

If the error between the current (ρ_i, θ_i) and the average $(\bar{\rho}_i, \bar{\theta}_i)$ is less than a pre-defined error ϵ , then we consider the current (ρ_i, θ_i) as valid entry. If the error is greater, then current (ρ_i, θ_i) is updated using the location from the previous frame, i.e. $(\rho_{i-1}, \theta_{i-1})$.

Now, if in the current frame, the lane markings are not detected, we increment what we call a null-counter. This counter finds the number of consecutive ‘no-lane’ detection in the current frame. If this null-counter goes beyond a threshold, it is assumed that there is no lane and the system generates lane marking based on the vehicle position. The null-counter is reset whenever a lane is detected in the current frame. In the case when a lane is not detected in the current frame and the null-counter is not maximized, then we assign the lane positions of the previous frame to the current frame. The temporal tracking for neighbor lanes is similar to the host-lane temporal prediction and is shown in Fig. 3.10. The computed or predicted host lane positions are used to first predicted the possible neighbor lane positions as discussed previously. If the neighbor lanes are not found in the vicinity of the predicted values, then the temporal prediction unit starts function in the same way as the host lanes prediction (shown in Fig. 3.10). Therefore, the same temporal module can be used for neighbor lane prediction also. It will be shown in the results section that the proposed method predicts the location of lanes to an acceptable accuracy.

Detecting Carriageway Boundary for Near-lanes The carriageway boundaries identify the end of the carriageway and therefore help in determining the boundary

conditions of the RoIs. In other words, they give an indication when to stop looking for the neighbor lane markings. When a carriageway boundary is identified, it implies that lanes need not be searched in the region beyond the boundary. This will help in eliminating the mis-detection of vehicles from the opposite direction as vehicles in the direction of the ego-vehicle. In addition, detecting the carriageway boundary also helps in reducing the computational complexity by eliminating unwarranted processing of additional blocks in the image.

We will first describe how the proposed techniques can be extended to identify a carriageway boundary adjacent to the host lane, i.e. we are interested in detecting the presence of any boundary with respect to the host lane only. If we find a carriageway boundary next to the host lane, say left host lane, it implies that left lane change is not possible, because there is no left adjacent lane L_{HL-1} . Considering that the host lane and the next adjacent neighbor lanes are the primary RoIs for lane change assistance, detecting carriageway boundary between these two lanes is more critical than between two neighbor lanes.

The following kinds of carriageway boundaries of host lanes are investigated as part of this research:

1. Double lane markings (Fig. 3.11(a))
2. Broad continuous lane marking (Fig. 3.11(c))
3. Road divider (Fig. 3.11(d))
4. Raised kerbs or footpaths (Fig. 3.11(b))

FIGURE 3.11: Carriageway boundaries and respective edge maps (a)(e) Double lane markings, (b)(f) Road divider, (c)(g) Broad continuous lane marking and (d)(h) Raised kerb

These are the different kinds of carriageway boundaries that are usually found on major roads and highways. The following observations were made about the edges of these carriageway boundaries:

- It was found that these boundaries exhibit a group of linear edges that are located close to each other. Fig. 3.11(e)-(h) show the respective edge formations for the different carriageway boundaries.
- The width of the boundary is usually wider than the normal lane markings. Fig. 3.11(c) shows this case. The width of a normal lane marking would be nearly $3\times$ lesser than the width seen in this case.
- The different edges that form these boundaries usually have varying prominence as shown in all the four cases in Fig. 3.11

Based on these observations, the proposed iterative method is ideal to detect these boundaries. In order to detect this boundary we use the following parameters from the proposed GMH-GAH-HT engine to identify the presence of the carriageway boundary:

- The difference between $\rho+$ and $\rho-$ that form the lane pair gives the width of the lane marking. The carriageway boundary is detected if this width is found to be more than the normal lane width that we described earlier in this section.
- The number of pairs of positive and negative ρ s indicates the presence of carriageway boundary. If the number of pairs is more than one, then this is most likely a double lane marking, which is a carriageway boundary.
- The number of lines itself (not just pairs) increases in the presence of a carriageway boundary. So if the GMH-GAH-HT engine results in group of lines, either paired or not, this is another strong indicator of a carriageway boundary.

When either of the above conditions happen, the carriageway boundary flag is triggered, which indicates that neighbor lane detection need not be performed on that side of the host lane.

Now, coming to the detection of carriageway boundary between lanes beyond the neighbor lanes. During our experimentation, we found that the proposed signatures are also valid for the boundary between such lanes. However, due to the perspective effect of these lanes, the lane markings were also found to be mis-detected as carriageway boundary. This happens because these lanes appear to be closer to each other due to perspective in the image. As a result, they also appear as groups of lines. Therefore, the proposed way of detecting the boundary is more applicable in the case of host lanes, which is more critical in the context of lane change assistance.

TABLE 3.1: Description of datasets that are used for evaluation

Set No.	Number of frames	Timing	Lighting/Weather	Other features	Kinds of Roads
Set 1	143	Day	Partially cloudy	Tree shadows	Major roads
Set 2	288	Day	Low illumination, dull weather	Changing tar colors	Major roads
Set 3	265	Day /Night	Day & Artificial tunnel lighting		Tunnel
Set 4	301	Day	Bright	Dark road	Tunnel
Set 5	301	Day	Bright	Light road color, faded markings	Major roads
Set 6	299	Evening	Bright	Tar color changes	Major roads
Set 7	281	Midnight	Night street light illumination	Street light glares	Major roads
Set 8	361	Day	Overcast skies, light drizzle, low illumination	Drizzle/Rainy	Highways
Set 9	361	Late Evening	Normal illumination	Tar color changes	Highways
Set 10	278	Late Evening	Normal illumination	Presence of road markings	Highways
Set 11	301	Afternoon	Rainy/drizzle, low illumination, wipers on	Wipers on	Highways
Set 12	260	Afternoon	Rainy/drizzle, low illumination, wipers on	Wipers on	Highways
Set 13	601	Late Evening	Rainy, cloudy	Wipers on	Highways
Set 14	291	Day	Bright	Dark road	Highways
Set 15	299	After sunset	Low illumination	Presence of arrows	Highways
Set 16	357	Evening	Normal illumination	Tar color changes	Major roads
Set 17	301	Evening	Low illumination	Wet roads	Major roads
Set 18	479	Evening	Overcast skies, low illumination, drizzle	Drizzle/Rainy	Major roads
Set 19	451	Day	Bright	Thick shadows of trees and overhead bridge	Major roads
Set 20	228	Night	Very low illumination, street light illumination	Glare from street lights and other vehicles	Major roads

3.3 Dataset Generation

In order to evaluate the proposed techniques, an exhaustive dataset of videos was created. Two video cameras were attached to the front and rear windcreens of the test ego-vehicle such that one camera looks forward and the other looks at the road scene behind the car. The test car and the camera setup is shown in Fig. 3.12. The videos were captured at a frame rate of 25 frames per second and at a high definition (HD) resolution, i.e. a frame size of 720×1280 pixels. Nearly 25 kilometers of test drive was conducted in different lighting conditions including day time, night time and late evening, with vehicle speed ranging from 70kmph to 90 kmph. Fig. 3.13 shows the test routes to collect the datasets. The videos were captured under different weather conditions such as bright day light, cloudy day, drizzling day etc. Different kinds of roads were captured like highway roads, major roads and roads inside a tunnel. Smaller video sequences were extracted from these videos based on the different road conditions that they capture. Table 3.1 gives the list of the different video sequences, the number of frames in each sequence and the road conditions that are being captured in the sequence. The total number of images in all datasets put together is 6525 frames.

FIGURE 3.12: (a) Test car, (b) & (c) Camera setup inside the car behind the rear view mirror.

3.4 Results & Discussion

In this section, we evaluate the robustness of the proposed multiple lane detection algorithm. Series of image sequences from real world data are used to evaluate the effectiveness of the proposed algorithm. We also present a discussion on the computation complexity of the proposed algorithm.

FIGURE 3.13: Map of Singapore showing the different routes which were captured, either completely or partially, for dataset generation

3.4.1 Evaluation Method

In order to evaluate the proposed techniques throughout the thesis, the following evaluation method is used. These techniques are evaluated for two metrics: (1) Accuracy, and (2) Computational cost efficiency. In order to evaluate the accuracy of the proposed techniques, we have developed these methods using MATLAB programming platform. All the algorithms were handcoded in MATLAB, i.e. in-built MATLAB toolbox based functions like edge detection, Hough transform etc. were *not* used. Instead, we developed our techniques for edge detection (as described in this chapter using GMH-GAH-HT engine), Hough transform (described in Chapter 4) etc. using MATLAB programming language. The proposed techniques were evaluated for accuracy using these MATLAB functions and the datasets presented in Section 3.3. The accuracy evaluation was performed on a PC-based platform.

In addition to accuracy, we also evaluate the computational complexity of the proposed methods against existing methods. For each of the proposed techniques, we derive the number of computations and compare them with equivalent metrics of existing techniques. Considering that most existing techniques for LCDAS have not addressed the computational complexity aspect of the algorithms, we derive the number of computations for existing techniques also, with which the proposed methods will be compared.

3.4.2 Accuracy Results

In this sub-section, we evaluate the accuracy of the proposed technique for multiple-lane detection in near-view. We present the accuracy of the technique in two different ways.

Detection Rate: Firstly, we evaluate the detection rate of the proposed technique in the different image sequences. The detection rate R is defined as:

$$R = \frac{\text{Number of Total Correct Detections}}{\text{Total Number of Frames in Sequence}} \quad (3.23)$$

In order to find the total correct detection in a sequence of input images, the output lane positions in near-view obtained from the proposed algorithm were marked on the input images. This was done for all the lanes, i.e. host lanes and neighbor lanes. If the output lane markings are obtained between the two edges of the lane markings present on the image, then the output lane position is considered to be a correct detection. If the output lane marking falls outside the lane marking, then it is considered as a mis-detection or a detection with an error.

TABLE 3.2: Detection rate using proposed algorithm

Seq. No.	N_f	Neighbor Left	Host Left	Host Right	Neighbor Right
1	143	1.000	1.000	1.000	0.965
2	288	0.979	0.983	0.983	0.965
3	265	1.000	1.000	1.000	NA
4	301	1.000	1.000	1.000	NA
5	301	(103) 0.941	1.000	0.950	NA
6	299	0.987	1.000	0.973	NA
7	281	NA	0.947	0.940	NA
8	361	NA	0.972	1.000	0.958
9	361	0.942	0.994	0.992	0.903
10	227	0.956	0.996	0.991	0.947
11	281	0.961	0.982	0.982	0.957
12	260	0.000	0.973	0.965	0.950
13	601	0.972	0.987	0.988	0.968
14	291	0.966	0.976	0.979	0.948
15	299	0.980	0.990	0.983	0.967
16	357	0.969	0.983	0.983	0.972
17	155	0.000	0.974	0.981	0.955
18	479	0.979	0.979	0.985	NA
19	451	0.984	0.987	0.982	NA
20	228	0.947	0.969	0.961	NA

Table 3.2 lists the detection rates of the proposed algorithm for the left and right host lanes, and the neighbor lanes that are found immediately next to the host lanes. ‘NA’ under neighbor lane columns in some sequences indicates that the sequence does not have that particular neighbor lane, i.e. a road boundary is detected on that side of the host lane. In some sequences, under the neighbor lanes, we indicate a number in parenthesis in addition to the detection rate. This number indicates the total number of frames used for computing the detection rate of that particular neighbor lane in that sequence. In

some sequences (Seq. #5), the neighbor lane may be covered by an overtaking vehicle in the initial frames. Therefore, we discard those frames in the beginning and the detection rate is determined for the frames from where the neighbor lane is visible according to ground truth data. The number in parenthesis gives the total frames used for computing the detection rate for that particular far lane.

From Table 3.2, it can be seen that the detection rates of the host lanes are between 94% to 100%, whereas the detection rates of neighbor lanes is between 90% to 100%. The high detection rates, sometimes near 100%, are because of the iterative nature of the proposed GMH-GAH-HT engine followed by the temporal history based estimation of the lane markings. In the various image sequences considered, the lanes are found to be of varying prominence. The GMH-GAH-HT engine is able to detect such lane markings to a high degree of accuracy as seen in Table 3.2. For example, in Seq. #1, the both host and neighbor lane markings are found to be varying prominence across frames. They are faded in some parts and prominent in others but the GMH-GAH-HT engine is able to detect and extract both the host and neighbor lane markings. We can see a near 100% accuracy in this sequence for all lanes. If the GMH-GAH-HT engine misses detecting lane markings due to gaps in the lane markings (as discussed in Section 3.2.2.2), the temporal history based prediction algorithm is able to predict the correct lane positions in most cases. As shown in Section 3.2.2.2, unless the correct detection of the lane markings occurs periodically using the current frame information, the proposed history based prediction can introduce high errors. Therefore, the detection accuracy of the lane markings using the GMH-GAH-HT engine plays a primary role in the detection rates shown in Table 3.2. Fig. 3.14 shows how temporal history based prediction is able to predict the lane position (when the lane is not available) but is corrected from time to time by the detection of lane marking by GMH-GAH-HT engine, whenever there is lane available. In Fig. 3.14, the lane marking is available in the near-view in Frames F1 and F2, which are picked by GMH-GAH-HT engine. Then, in F3, F4 and F5, there is not enough lane information for GMH-GAH-HT engine to pick. Therefore, temporal tracking module starts predicting the lane marking. It is because of this temporal tracking, that sometimes the predicted lane position may be slightly deviated as seen in F5 (host left lane and far left lane). However, in F6, the lane marking is again available in near-view, which is detected by the GMH-GAH-HT engine. This results in the correct detection of lane marking, which will be used for temporal prediction in the next sequence of images when there are no lane markings in the near view.

From Table 3.2, it can be seen that the detection rates of neighbor lanes are usually lesser than those of host lanes. This is because of the low prominence of the neighbor lanes as compared to the host lanes, which increases the complexity of detecting the neighbor lanes. Even though the neighbor lanes are less prominent and in some cases

FIGURE 3.14: Sequence showing detection of lane markings using both GMH-GAH-HT engine and temporal history based prediction.

faded, the proposed algorithm shows an average detection rate of around 97% and 95% for left and right neighbor lanes respectively. In Table 3.2, the detection rates for left neighbor lanes in Sequences #12 and #17 are shown as 0 but that does not mean that the detection rate is 0. We indicated a 0 to show that though there is lane boundary in the form of a kerb, the lane marking is not visible even in the ground truth as shown in Fig. 3.15. The proposed algorithm is able to detect the road boundary but not the lane marking because there is no lane marking in the image.

FIGURE 3.15: Test cases where there are no visible lane markings

Few sequences, with some complex scenarios are worth mentioning to show the effectiveness of the proposed algorithm. The effectiveness of GMH-GAH-HT engine for detecting faint or faded markings is shown in all sequences, especially in detecting the neighbor lanes. In some sequences like Seq. #1, even the host lanes were found to be faint and broken. But the proposed host lane algorithm using the GMH-GAH-HT engine was able to extract the lane markings. Fig. 3.16 shows the detection results of the faded host lane markings in Seq. 1. It can be seen that the host lane markings in the input image frames (on the left) are faded to a high degree but they are detected accurately by the algorithm. The neighbor lanes in most sequences are fainter than the host lanes. The

FIGURE 3.16: Faint host lane markings detected by the proposed GMH-GAH-HT engine

GMH-GAH-HT algorithm is able to extract these faint neighbor lanes successfully. This is the reason for high detection rates of the neighbor lanes shown in Table 3.2. Fig. 3.17 shows two sample outputs showing host and neighbor lanes detected by the proposed algorithm. It can be seen from the input images that the lane markings are of varying prominence. The host lane markings and left neighbor lane are of higher prominence than the right neighbor lane. In such cases, varying thresholds are needed to extract the edges as shown by the sample results in Fig. 3.17. Fig. 3.18 shows the detection of lane markings in other complex scenarios. Fig. 3.18(a) & (b) show the detection results of lane markings in the presence of strong shadows. The GAH-HT combination in the GMH-GAH-HT engine ensures that non-lane markings like the shadows of trees are eliminated and lane markings are extracted. Fig. 3.18(c) & (d) show detection results on lanes in rainy weather conditions. It can be seen that the lanes are detected even in the presence of the wiper movement. This is because the edge pairing that is done using GAH+ and GAH- in the GMH-GAH-HT engine rejects the noisy edges generated by the wiper and selects only the lane markings. Fig. 3.18(e) & (f) show image frames

FIGURE 3.17: Lane markings of varying prominence detected by the proposed GMH-GAH-HT engine

from the two night sequences (Seq. #10 & #20). Though the lane markings are faded and faint, especially the left neighbor lane in Fig. 3.18(f), the proposed algorithm is able to detect the lane marking.

Mis-detections: There is a small percentage of mis-detections and these cases were analyzed further to understand the cause for not detecting them accurately. It is to be noted that the accuracy is defined by checking if the detected lane is falling on the actual lane marking in ground truth information. In some cases, a lane is detected but may have been a few pixels either to the right or left of the lane marking in the near-view. Therefore, in such cases, though the lane is detected, it is still considered as incorrect because of the localization constraint that is set, i.e. the detected lane marking must fall on the lane marking from ground truth.

One of the reasons for mis-detections due to the above reason is because of the absence of lane marking in the near-view and the temporal tracking algorithm is predicting the lane position based on history. In the proposed temporal history based prediction, it is assumed that the vehicle is not making any drastic movements in the time span between two frames, i.e. 1/30-th of a second. However, if the vehicle does make such maneuvers, the result from the temporal tracking algorithm is found to be a few pixels away from the ground truth lane marking, which results in a mis-detection. In Fig. 3.14, the left neighbor lane marking in F5 can be considered as mis-detection because it is a few-pixels away from the actual lane marking. However, as discussed earlier in this section, the presence of a marking in the near-view again in the subsequent frames is detected by the GMH-GAH-HT engine and the deviation (if any) in the temporal history based

FIGURE 3.18: Host and neighbor lane markings detected by the proposed GMH-GAH-HT engine under varying weather and lighting conditions: Presence of shadows in (a) & (b), rainy weather conditions in (c) & (d), and night scene in (e) & (f)

prediction is removed from the new detection onwards. Also, as shown in Section 3.2.2.2, the proposed prediction is based on image information only. Other vehicle parameters like yaw angle of the vehicle can be introduced and a more sophisticated tracking filter like Kalman filter can be used to eliminate such errors resulting from movements of the vehicles laterally in such small windows of time.

Another reason for mis-detections is the presence of other road markings. Fig. 3.19 shows two such cases where mis-detections occur due to presence of other markings. It is to be noted that such detections do not happen all the time. They occur only if a road marking shows exactly the same properties as that of the lane and also falls in the similar expected regions where lanes are found. In most cases, these mis-detections do not occur because such detections are rejected based on temporal history information. After rejecting these candidates, the GMH-GAH-HT engine looks for markings again that satisfy the properties of the lane markings. These detections persist as mis-detections if and only if they occur continuously in more than 10-15 frames continuously, which affects the temporal averaging (discussed in Section 3.2.2.2).

FIGURE 3.19: Mis-detections due to presence of other road markings

There can be other reasons for mis-detections, especially when the lane marking itself is not seen at all as shown in Fig. 3.15. This occurs usually when it is raining or the lane markings are faded beyond any trace on the road. Also, the proposed vision-based techniques failed to detect the lane markings in very heavy rain, wherein even the driver himself was unable to find where the lane markings are.

Detection of lanes beyond neighbor lanes: The proposed GMH-GAH-HT engine is most effective in detecting the neighbor lanes adjacent to the host lanes. In the context of an LCAS, as described in the ISO standards, the adjacent neighbor lanes form the critical RoIs that need to be continuously monitored. However, one can argue about the effect of vehicles on the lanes beyond the neighbor lanes, i.e. the outer lanes in the case of a carriageway with more than 3 lanes. Consider the scenario of a 4-lane highway, with the ego-vehicle in Lane 2. Therefore, it is sufficient to detect Lane 1 and 3 for the objectives listed in ISO standards for LCDAS. However, if we consider the vehicles on Lane 4 trying to change lane into Lane 3, they will pose a danger to ego-vehicle if it is also trying to change lane to Lane 3. In such scenarios, it is important to detect the farther lanes also.

The lane markings of the lanes beyond the neighbor lanes are fainter than the adjacent neighbor lanes. The proposed GMH-GAH-HT engine is able to detect such faint lane markings also. Fig. 3.20 shows some sample results where these lanes beyond the neighbor lane markings are detected. However, there are false positives also. In cases when there are arrow markings etc. in the outer lanes, it becomes difficult to differentiate the lane markings from the other road markings. This is because the perspective effect of the image reduces the gap between the arrow markings and valid lane markings. Also, due to the same reason, the arrow markings are distorted to look like parallel lines like the lane markings. Therefore, the proposed engine, though is successful in detecting the lane markings of the lanes beyond the neighbor lanes, it is less effective in differentiating their lane markings from other road markings and lane like features. However, for the adjacent neighbor lanes, the detection accuracy is as high as 97% as shown previously.

FIGURE 3.20: Detection of lanes beyond neighbor lanes

FIGURE 3.21: Collection of ground truth data for computing lane position deviation

Lane position deviation analysis: In addition to the above analysis on detection rates, we also determined the deviation of lane positions with respect to a known ground truth information. In this analysis, we manually selected points on the input images in a few sequences to mark the host and neighbor lanes in near-view. This is shown in Fig. 3.21. The ‘*’ shown on the lane markings are marked manually. These points are marked on the outer edge of the host lane markings. In the case of neighbor lane markings, the edges are closeby and not easily separable. Therefore, for neighbor lanes, these points are marked on the lane marking. These points are then used to find the equations of lines for the lanes in near-view. We then compute the intersection point of these lane markings with the base of the image $y = 720$, i.e. the nearest point on the road surface to the vehicle as seen in the image.

The proposed algorithm gives the lane positions in the form of (ρ, θ) , which represents the parameterized line equation, i.e. $\rho = x \cos \theta + y \sin \theta$. We consider the (ρ, θ) tuples for the left edge of every lane marking. This parameterized equation for each lane is used to find the x coordinate where the lane intersects at $y = 720$, in the case of the lane positions obtained from the proposed technique. These lane positions from ground truth and results from the proposed techniques, are plotted for two sequences as shown in Fig. 3.22. The top plot shows the lane positions for Seq. #1 and the bottom plot for Seq. #3. The line plots with ‘-*-’ indicate ground truth data and the ones with ‘--’ represent the experiment data.

The plots in Fig. 3.22 show that the host lane positions obtained from the experiment follow the ground truth data very accurately. The average deviation in the two sequences

FIGURE 3.22: Plots showing lane position deviation from ground truth data for Seq #1 in top plot and Seq #3 in the bottom plot.

for the host lane data was found to be less than 10 pixels. However, the average deviations in neighbor lane positions with respect to the ground truth data was found to be higher as seen in the plots. This higher deviation can be attributed to two factors. Firstly, there are errors in ground truth generation itself, especially for neighbor lanes. Since neighbor lanes are fainter and less prominent than the host lanes, the ground truth marking of the neighbor lanes is more error prone. As indicated earlier, the ground truth points are marked on the neighbor lanes but the (ρ, θ) for the lane marking corresponds to the left edge of the lane marking. Secondly, the slightly higher deviations also arise due to the temporal history based prediction. The plots in Fig. 3.22 show that the deviation goes to zero and increases/decreases in an alternating fashion over every few frames. This is because of the temporal prediction taking over when there are no lane markings detected in the near-view. The deviation is higher in the case of neighbor lanes as compared to host lanes. However, the deviations in the lane positions does not lead to too many false detections as shown in Table 3.2.

3.4.3 Computation Complexity Analysis

In this subsection, we will evaluate the computation complexity of the proposed GMH-GAH-HT engine in the context of lane detection in near-view and compare it against existing lane estimation techniques. In order to perform the comparison, we consider a

few recent lane detection implementations reported in literature. Firstly, we compare with Hough based implementations in [64], [65] and [69]. Next, the implementation reported in [2], which is one of the most cited lane estimation techniques in literature, is considered for comparison. In addition to the above, we also compare a color processing based lane detection method in [52].

Also, in this chapter, considering that we are interested in detecting the lanes in near-view, the same region of interest is considered for all the methods for comparison (although [64], [65], [69], [2] and [52] process the entire image for lane detection). Therefore, the last row of blocks comprising of 90×1280 pixels in the image is considered for lane detection in near-view. The computational complexity of the proposed method for lane detection in near-view is first determined and then compared with the same metric for the methods described in [64], [65], [69], [2] and [52].

Given that the proposed method is a block-based method, the number of computations in a single block are determined first. The total computation cost for detecting host and neighbor lanes in the entire near-view given is equivalent to approximately 10 times the computation cost of one block. This is because there are ten 90×128 sized blocks in the last row of blocks in the current image division.

In the proposed GMH-GAH-HT method, the computation complexity can be divided into two components. Firstly, we have the GMH and GAH operations which *select the edge pixels* that most probably form the lane edges. Secondly, the HT computation receives the edge pixels from GMH and GAH engine to *locate* the lane edges in the image. The computational complexity of the first step using GMH and GAH is dependent on the block size. The computation complexity of this step can be established in the big O notation as $O(M_b N_b)$ for an $M_b \times N_b$ sized block. This is because computing the GMH and GAH involves computation of gradient magnitudes and angles for every pixel in the $M_b \times N_b$ sized block. If Sobel edge kernels described in Section 3.2.1 are considered, the computation cost (in terms of additions) to compute the gradient magnitude for every pixel can be determined as 6 additions (say 16-bit additions). Gradient angle computation can be implemented as a look-up table and hence no additional computations are required. GMH computation involves histogram counter additions, which results in $M_b \times N_b$ additions. Similarly, if we consider the worst case scenario of going through all peaks in a GMH, the GAH computation cost is equal to $M_b \times N_b$ additions. Therefore, the total cost of computing the GMH and GAH is given by:

$$C_{ADD}^1 = 8M_b N_b \text{ additions} \quad (3.24)$$

It should be noted that the above formulation is for the worst case scenario in which all pixels in the block are processed for GAH computation (in the proposed GMH-GAH-HT engine, a percentage of the all the pixels in the block only are processed).

Let us now consider the second component, i.e. the Hough transform computation for locating the lane edges. According to several studies [80, 200], the computational cost of HT in terms of both timing and area affects most line detection algorithms. This is because a HT computation on every edge pixel involves N_θ number of two trigonometric operations, two multiplications, one addition and as many accumulations. Here N_θ is the number of angles for which HT is computed. Therefore, if $p\%$ of $M_b \times N_b$ sized image block is edge content, then the number of computations required to compute the HT of the image block can be summarized as follows:

$$C_{ADD}^2 = pM_bN_bN_\theta \quad (3.25)$$

$$C_{MUL}^2 = 2pM_bN_bN_\theta \quad (3.26)$$

$$C_{TRIG}^2 = 2pM_bN_bN_\theta \quad (3.27)$$

$$C_{ACC}^2 = pM_bN_bN_\theta \quad (3.28)$$

It can be seen from above equations that the computation cost of HT can be reduced by either reducing p or N_θ , i.e. either by reducing the number of edge pixels for which HT is computed or by reducing the number of angles in Hough space. The proposed GMH-GAH-HT engine ensures that the computational load on the HT is minimized by reducing both p and N_θ in an informed manner. In the proposed algorithm, edge pixels are filtered at every stage and edge pixels forming the lane edges only are selected for HT computation. Firstly, the use of GMH results in controlled release of edge pixels that form prominent edges. Therefore, the factor p in above equations is monitored continuously. If a lane marking is found in the very first iteration of the GMH-GAH-HT engine, then there is no need for the engine to release more edge pixels. Conventionally, the edge map is created once without having the knowledge of the kind of edges we are looking at. However, the GMH-GAH-HT engine allows an informed release of edge pixels. p is lesser for host lanes as compared to neighbor lanes. This is because of the higher prominence that the host lanes have as compared to neighbor lanes. The proposed engine is aware of this and hence releases only the required number of edge pixels. In addition to GMH, we have GAH in the proposed engine. This ensures that N_θ is controlled because HT is computed only for the angles that show a peak corresponding to a linear edge in the GAH.

In the proposed method, if we consider a worst case scenario of the lane marking being along the diagonal of an $M_b \times N_b$ sized block (i.e. the longest edge in the block), the number of edge pixels for which HT is computed in the GMH-GAH-HT engine is given

by:

$$p_{prop} = n_t \times \sqrt{M_b^2 + N_b^2} + \delta_p \quad (3.29)$$

where n_t is the thickness of the edge, i.e. the number of edge pixels that define the edge, and δ_p is the additional stray noisy pixels. In addition to reduction in number of edge pixels, the use of GAH further reduces the number of angles for which HT is computed in the proposed method. In conventional methods, an angle range of about 40° is considered so that the variations in the lane orientation can be covered adequately, in order to compute the Hough transform. In the proposed method, the GAH gives the possible angle where the lane edge could be located. Therefore, we consider an angle range of 15° only for which HT is computed.

In order to compare the computational cost efficiency of the proposed method, let us first consider the conventional Hough based lane estimation techniques in [64], [65] and [69]. In all these methods, different variants of Hough Transform are used to detect lanes. The following derivations for computation cost hold true for all three methods because HT is the first and the basic operation that is employed in [64], [65] and [69]. Additional steps are employed in [64, 65, 69] to further improve the accuracy of the lane detection process in these methods, which adds to the computation cost that is derived below.

Let us consider one block of image of size 90×128 , i.e. $M_b = 90$ and $N_b = 128$ for comparison. In the conventional Hough based lane detection methods in [64, 65, 69], an edge map is computed using Sobel kernels [74]. Therefore the computation cost of determining the edge map is equal to $6M_bN_b$ additions. About 10% of the edge map is usually considered to be edge content [80]. This gives $p = 0.1$. Therefore, Hough transform is applied for $0.1M_bN_b$ pixels for N_θ angles. Table 3.3 compares the number of operations for processing one 90×128 sized image block using conventional Hough based lane detection method and the proposed method. We consider $M_b = 90$ and $N_b = 128$, and $N_\theta = 40$ angular locations in Hough space for conventional methods. For the proposed method, we consider the worst case scenario of having a lane along the entire diagonal of the block and the width of each lane edge is 3 pixels (which is an exaggeration). Therefore, we get $n_t = 6$, three for each edge - left and right edge of the lane. It can be seen from Table 3.3 that a savings of nearly 70% is obtained for most operations in the proposed method. The total number of operations (in terms of additions) is computed by assuming the cost of 16-bit multiplication, trigonometric operation and comparison as being equivalent to sixteen 16-bit additions.

It should be noted that the above savings is obtained if we employ the conventional Hough transform that uses the parametric equation in (2.1). In the next chapter, a

TABLE 3.3: Comparison of computation complexity with Hough based lane detection methods

Operations	Conventional [64, 65, 69]	Proposed	Percentage savings
Additions	115200	106230	7.78
Multiplications	92160	28140	69.46
Trigonometric operations	92160	28140	69.46
Accumulations	46080	14070	69.46
Total computations (equivalent additions)	3110400	1020780	67.20

computationally efficient block-based parallel Hough transform engine is proposed that reduces the computation cost of the proposed lane detection further.

We will now compare the cost of the proposed method against [2]. This method uses steerable filters to detect directional edges. For every image pixel coordinate, it employs three $k \times k$ sized kernels ($k = 5$ is the minimum value). Each of these kernels is *not* a simple kernel like Sobel (which can be computed using 6 additions). Instead, each kernel operation involves k^2 multiplications followed by $k^2 - 1$ additions. In addition to the kernel operations, it also involves three additional multiplications, one division, one square root and one trigonometric operation to determine the directionality of each pixel coordinate. Therefore, for an $M_b \times N_b$ image block, the number of operations are equal to:

$$C_{ADD}^{SF} = (3k^2)M_bN_b \quad (3.30)$$

$$C_{MUL}^{SF} = (3k^2 + 3)M_bN_b \quad (3.31)$$

$$C_{TRIG}^{SF} = M_bN_b \quad (3.32)$$

$$C_{DIV}^{SF} = M_bN_b \quad (3.33)$$

The selected pixels that give a particular directionality using the steerable filters are then subjected to Hough transform. For the HT computation in [2], we consider a similar complexity for Hough transform computation as the proposed method (thereby giving computational benefit to [2]). Table 3.4 compares the computation complexity of the two methods. The cost of one division is considered to be the same as one multiplication for comparison purposes in Table 3.4. It can be seen that the proposed method is computationally more efficient than the lane detection method in [2], even if the worst case scenario is considered for the proposed method.

The method in [52] employs color image processing, which is computationally more complex than techniques that use gray level images. This is because each input RGB pixel is first converted to YIQ format using a 3×3 matrix multiplication operation. The computation cost of every 3×3 matrix multiplication operation is equivalent to 9

TABLE 3.4: Comparison of computation complexity between proposed method and [2]

Operations	[2]	Proposed	Percentage savings
Additions	864000	106230	87.70
Multiplications	910080	28140	96.90
Trigonometric operations	39600	28140	28.93
Accumulations	14070	14070	0
Total computations (equivalent additions)	16072950	1020780	93.64

multiplications and 6 additions. Therefore for 90×128 sized image block, the computation cost of RGB to YIQ color space conversion itself is 1728000 addition operations (assuming 1 multiplication is 16 additions). This cost is 1.4 times the cost of the entire proposed lane detection process.

3.5 Computational Architecture for Multiple-lane Detection

In this section we describe a computational architecture for the proposed technique. The GMH-GAH-HT engine forms the core computation block of the proposed algorithm. It was shown in Section 3.2 that the algorithm inherently is most robust when implemented in a block-based fashion rather than taking the entire image. Therefore, we propose a block-based computation architecture (shown in Fig. 3.23) for this engine, which lends well with the block-based nature of the GMH-GAH-HT engine. The proposed computation architecture comprises of a series of processing engines (PEs), which are controlled by a control unit called Near-view Control Unit (NVCU). Each PE takes an image block as input and detects the lane markings in the near-view. The NVCU configures which PEs should detect near lanes and which PEs should detect the neighbor lanes. Fig. 3.24 shows the functional blocks of each PE. The control unit in the PE sets the parameters for the GMH-GAH-HT engine to detect and extract lane markings. The parameters are related to either near lanes or neighbor lanes depending on the control signal sent by the NVCU. There is a temporal prediction unit which predicts the lane markings in case they are not detected in the block. It should be noted that the same GMH-GAH-HT engine and temporal prediction engine are used in all PEs. With the change in control signals from the control unit and NVCU, each PE can be configured to detect either the near lanes or neighbor lanes. Also, each PE can operate in parallel. In other words during near lane detection, all PEs corresponding to near lanes can operate in parallel. Based on the information generated by these PEs, the remaining PEs can be triggered to operate in parallel and detect the neighbor lanes.

FIGURE 3.23: Computation architecture for detecting host and neighbor lanes in near-view

FIGURE 3.24: Processing engine (PE) in computation architecture for detecting host and neighbor lanes in near-view

3.6 Summary & Conclusions

In this chapter, an iterative technique for multiple-lane detection was proposed to extract host lanes and neighbor lanes in near-view. A feedback-based engine using gradient magnitude histogram (GMH) along with Gradient Angle Histograms and Hough Transform was proposed to detect lanes of varying edge prominence. GMH was used to group edge pixels of varying prominence and the GMH-GAH-HT engine was proposed to dig deeper into low-gradient edge pixels in an informed fashion to release groups of edge pixels and extract lane markings. The detailed formulation of the proposed technique for host lanes and neighbor lanes was proposed. A simple history based prediction method was proposed which could be reused for predicting both host and neighbor lanes. The proposed techniques were comprehensively tested on varied image sequences from test videos, that were captured under different lighting and weather conditions. The detection rates of the proposed techniques were as high as 98% for neighbor lanes and nearly 100% for host lanes. Detection rates of upto 95% were seen in rainy weather conditions and night videos. A detailed analysis of detection rates in different complex scenarios shows that the proposed techniques are able to detect faint and faded lane markings

under varying global and local illumination conditions. An analysis of the computational complexity was also presented. It was shown the computation cost due to HT was reduced by nearly 50% because of the proposed use of GMH-GAH-HT engine for lane detection. In addition the GMH based edge release processing also reduces the computational complexity substantially as compared to existing techniques. A parallel computation architecture was also proposed, which can be configured to give different levels of parallelism depending on the resource constraints and user requirements. In the next chapter, we will delve into detecting lanes in far-view and propose parallel computation engines that will enable lane detection in far-view as well as near-view.

C H A P T E R 4

Detection of Host & Neighbor Lanes in Far-view

4.1 Introduction

In the previous chapter, techniques for detecting multiple lanes, i.e. host lanes and neighbor lanes, in near-view were proposed. In this chapter, techniques are proposed to detect these lanes in far-view. Far-view is defined as the region of road surface that is not immediately in front of the host vehicle. Extracting the host and neighbor lanes in far-view involves the following challenges. First, the lane markings in far-view are fainter than those found in near-view. The prominence of the intensity gradients of the lane markings varies as we go from near-view to far-view. Fig. 4.1 illustrates this phenomena. In the case of multiple lanes, the size of the lane markings also varies as the lanes are traversed from near-view to far-view. Also, the host and neighbor lanes are bound by dashed lane markings. In far-view, the dashed lane markings appear smaller than the other features in the road scene such as vehicles, buildings etc. Therefore, detecting these smaller and less prominent lane marking features in far-view in the presence of other features is the first challenge. Next, lanes in far-view can be either straight or curved. Unlike near-view, where the lanes are straight, the same lanes can appear as straight or curved depending on the road curvature. Therefore, detecting whether the lanes are straight or curving in far-view is the next challenge.

In this chapter, we first propose techniques to detect host and neighbor lanes in far-view in Section 4.2 and 4.3. The proposed techniques detect and locate both straight lanes and curved lanes in far-view. The effectiveness of the proposed techniques in detection of lanes in far-view is then discussed in Section 4.4. Thereafter, the computational

FIGURE 4.1: Prominence of lane markings' reduces from near-view to far-view. Size of lane marking features also decreases.

challenges of the proposed techniques are described in Section 4.5. In order to accomplish fast lane detection process, we propose parallel block-based computation engines in Section 4.6. The proposed techniques and architectures increase computation speed by orders of magnitude, without increasing the overall computation cost. Thereafter, in Section 4.7, techniques are proposed to efficiently combine block-based information to generate global information, which is used to detect the host and neighbor lanes in far-view. Finally the chapter is summarized in Section 4.9.

4.2 Extracting Host & Neighbor Lanes in Far-view

In this section, a technique is proposed to extract the host and neighbor lanes in far-view. Considering that the roads, especially highways and major roads, are usually straight, the proposed method first tries to find straight lanes in far-view using the cues from the lanes detected in near-view. If it fails to detect straight lanes, then we check if the lanes are curving in the far-view region. The proposed method involves in the following steps:

- Step 1: Setting far-view region and priori-information from near-view lanes
- Step 2: Extracting lane marking edges in far-view using block-based GMH techniques and priori information
- Step 3: Analyzing block-based information to extract positions of straight lanes in far-view
 - Step 3.1: Confirming presence of curved lanes in case straight lanes are not found

These steps will be described in greater detail and parallel techniques to compute these steps efficiently are proposed. In order to detect the different lanes in far-view, the lane information obtained from near-view is used to set the far-view region and priori information about lanes in far-view. Since the lanes in far-view are expected to be straight lanes, the vanishing point from the host lanes in near-view are used to set the vanishing point. The vanishing point defines the region in the image towards which all the lanes are directed. The area in the image below the vanishing point is expected to show the lanes. Therefore, the vanishing point can be used to define the near-view region. This is shown in Fig. 4.2. Let the shaded blocks represent the near-view region where lanes are detected (given by solid lines). Let the host lanes that are detected in near-view be given by $L_{hl}(\rho_{hl}, \theta_{hl})$, $L_{hr}(\rho_{hr}, \theta_{hr})$ and neighbor lanes by $L_{fl}(\rho_{fl}, \theta_{fl})$, $L_{fr}(\rho_{fr}, \theta_{fr})$. The (ρ, θ) indicate the position of the lanes as given by their Hough spaces. As discussed in Section 3.2.2.2 (equation (3.17)) in the previous chapter, the vanishing point $V_{pt}(x_{vpt}, y_{vpt})$ is computed using the host lanes $L_{hl}(\rho_{hl}, \theta_{hl})$, $L_{hr}(\rho_{hr}, \theta_{hr})$. It can be seen from Fig. 4.2 that the vanishing point can be used to define the blocks that form the far-view region. The solid white blocks in Fig. 4.2 form the far-view region in which the lanes will be searched. In addition to setting the far-view region, the lanes in near-view

FIGURE 4.2: Defining far-view region using lanes from near-view.

also give some priori information about the lanes in far-view. If the lanes are straight, then the far-view lanes will be as shown by the dashed lines in far-view blocks in Fig. 4.2. Therefore, the angular information about the lanes in far-view can be obtained from the angular information of lanes in near-view. This information will be used in the next steps to eliminate/reduce non-lane features.

The blocks in far-view are now processed to extract edges of the lane markings in far-view. Since these edges are expected to be of varying prominence in terms of gradient magnitudes, the proposed block-based GMH based technique is employed in the following way to extract these lane markings. For every block in far-view region, the GMH is

computed and the threshold boundaries are set to split the GMH into groups of edge pixels having similar gradient magnitudes.

Next, each group of pixels from GMH is released and the positive and negative GAHs are computed for the edge pixels (the computation process of GAH+ and GAH- is described in Section 3.2.2). In order to evaluate the GAHs, the priori information from Step 1 is used. This priori information includes the angle orientations of the lane markings in near-view. Let these priori information with angle orientations be given by the set $P_\theta = \{\theta_{hl}, \theta_{hr}, \theta_{fl}, \theta_{fr}\}$. If the lanes in far-view are straight lanes, they must also be having similar angle orientations. Therefore, GAH+ and GAH- are analyzed for these angles only. The edge pixels falling in these angle bins are retained and other edge pixels are rejected.

As every group of edge pixels from the GMH is released, the accumulated GAH+ and GAH- are computed by filling the bins corresponding to the angles in P_θ . This edge pixel release process is terminated using the termination conditions that were described in Section 3.2.2.2. It is expected that the lanes in the far-view region can be of low prominence in terms of gradient magnitudes. Therefore, the groups of edge pixels in GMH must be released till a minimum threshold in GMH where the termination condition is satisfied.

After reaching the termination condition, the bins in GAH+ and GAH- are checked to see if the counts in them are greater than a minimum threshold. If these bins cross the threshold in both GAH+ and GAH-, the positive and negative edge maps E^+ and E^- are generated. Fig. 4.3 summarizes the proposed technique to extract lane marking edges in far-view.

FIGURE 4.3: Detecting host & neighbor lanes in far-view by generating global Hough spaces from block level Hough spaces

After getting lane edge markings in each block $B(i, j)$, Hough transform is applied to E^+ and E^- in every block to generate the Hough spaces $HS_{(i,j)}^+$ and $HS_{(i,j)}^-$ corresponding to E^+ and E^- respectively. These block-level Hough spaces are combined together to generate global Hough spaces HS^+ and HS^- . This addition of Hough spaces is shown in Fig. 4.4. The peaks in HS^+ and HS^- will indicate if there are any straight lines. The difference between close peaks in HS^+ and HS^- indicate pairs of close parallel lines, which form the lane markings (this was discussed in Section 3.2.2.2).

FIGURE 4.4: Detecting host and neighbor lanes in far-view by generating global Hough spaces from block level Hough spaces

In order to find out if the blocks in far-view have straight lanes, the following analysis is done using the peaks obtained from HS^+ and HS^- . For the sake of explanation, the peaks in HS^+ are used to describe the method. The same is extended to HS^- also. As shown in Fig. 4.2, let the host lanes in near-view be represented by $L_{hl}(\rho_{hl}, \theta_{hl})$ and $L_{hr}(\rho_{hr}, \theta_{hr})$. Let the peak heights of these lines at (ρ_{hl}, θ_{hl}) and (ρ_{hr}, θ_{hr}) in the Hough spaces be given by P_{hl} and P_{hr} . In the global Hough spaces generated from the blocks in near-view and far-view, i.e. HS^+ and HS^- , let $(\rho_{hlf}, \theta_{hlf})$ and $(\rho_{hrf}, \theta_{hrf})$ be the peak locations in HS^+ with peak heights given by P_{hlf} and P_{hrf} . In order to check if there are straight lanes in far-view, the following are computed:

$$\begin{aligned} \delta\rho_{hl} &= |\rho_{hl} - \rho_{hlf}| & \delta\rho_{hr} &= |\rho_{hr} - \rho_{hrf}| \\ \delta\theta_{hl} &= |\theta_{hl} - \theta_{hlf}| & \delta\theta_{hr} &= |\theta_{hr} - \theta_{hrf}| \\ \delta P_{hl} &= |P_{hl} - P_{hlf}| & \delta P_{hr} &= |P_{hr} - P_{hrf}| \end{aligned} \quad (4.1)$$

The following conditions are checked for the differences found above for the peaks in HS^+ and HS^- .

$$\begin{aligned} 0 &\leq \delta\rho_{hl} \leq T_{\delta\rho_{hl}} & 0 &\leq \delta\rho_{hr} \leq T_{\delta\rho_{hr}} \\ 0 &\leq \delta\theta_{hl} \leq T_{\delta\theta_{hl}} & 0 &\leq \delta\theta_{hr} \leq T_{\delta\theta_{hr}} \\ T_{\delta P_{hl}} &\leq \delta P_{hl} & T_{\delta P_{hr}} &\leq \delta P_{hr} \end{aligned} \quad (4.2)$$

where T s in the above equations represent thresholds that are set based on the region size. If these conditions are met in both HS^+ and HS^- , then straight lanes are considered as present in far-view at $(\rho_{hlf}, \theta_{hlf})$ and $(\rho_{hrf}, \theta_{hrf})$. In Section 4.4.1, we illustrate how this method detects the far straight lanes for different road scenarios.

4.3 Extending Block-based Technique for Detecting Curved Lanes

If a straight lane is not detected using the technique described above, then the far-view region is checked for curved lanes. Unlike curve fitting models to detect the curved lanes, a technique based on linear approximation of the curve is proposed in this section. The technique is described as follows.

Firstly, it must be noted that all lanes curve in the same direction. If the direction of curvature of the host lane is detected, the directions of the neighbor lanes can be also be determined. Therefore, the curvature of the host lane is determined as the first step. The near-view information of the host lane and the additive property of the Hough transform are used to accomplish this. The proposed method is shown first for the left host lane. The same can be extended to the right host lane. Let $L_{hl}^+(\rho_{hl}^+, \theta_{hl}^+)$ and $L_{hl}^-(\rho_{hl}^-, \theta_{hl}^-)$ represent the positive and negative edge of the left host lane, which are detected in the near-view blocks. For the sake of explanation, the positive edge $L_{hl}^+(\rho_{hl}^+, \theta_{hl}^+)$ is used in the explanations below.

FIGURE 4.5: Straight lanes and their locations in blocks

The first step to find the curvature of the lane is to determine the blocks which the host lane would have covered if the lane were straight. Referring to Fig. 4.5, if the lane were straight lane, the left host lane given by L_{hl}^+ would have covered the blocks indicated as shown in the figure. The axes give the indices of the block. Therefore, if the

lane were straight, the left lane found in block $B(8, 2)$ in near-view region, would have traversed through blocks $B(7, 2)$, $B(7, 3)$ and $B(6, 3)$ in far-view region before hitting the vanishing point V_{pt} . In order to find out the curved trajectory, we first locate these blocks and check for the curvature of the lane block by block in the following fashion.

In order to locate these blocks, the Hough transforms of the block origins are used. If the image is divided into $M_b \times N_b$ blocks and if V_{pt} is found in block $B(m, n)$, then a $M_b - m + 1 \times N_b$ matrix $\mathbf{G}_{\theta^+}$ is defined in the following way:

$$\mathbf{G}_{\theta^+} = \begin{bmatrix} \rho_{(m-1,1)}^+ & \rho_{(m-1,2)}^+ & \cdots & \rho_{(m-1,N_b)}^+ \\ \rho_{(m,1)}^+ & \rho_{(m,2)}^+ & \cdots & \rho_{(m,N_b)}^+ \\ \vdots & & \ddots & \vdots \\ \rho_{(M_b,1)}^+ & \rho_{(M_b,2)}^+ & \cdots & \rho_{(M_b,N_b)}^+ \end{bmatrix} \quad (4.3)$$

where each element $\rho_{k,l}^+$ in $\mathbf{G}_{\theta^+}$ is the global Hough transform values of the origin of block $B(k, l)$ for the angle of the left host lane, i.e. θ_{hl}^+ . For the left host lane $L_{hl}^+(\rho_{hl}^+, \theta_{hl}^+)$ in Fig. 4.5, $\mathbf{G}_{\theta^+}$ is given by:

$$\mathbf{G}_{\theta^+} = \begin{bmatrix} \rho_{(5,1)}^+ & \rho_{(5,2)}^+ & \cdots & \rho_{(5,6)}^+ \\ \rho_{(6,1)}^+ & \rho_{(6,2)}^+ & \cdots & \rho_{(6,6)}^+ \\ \vdots & & \ddots & \vdots \\ \rho_{(8,1)}^+ & \rho_{(8,2)}^+ & \cdots & \rho_{(8,6)}^+ \end{bmatrix} \quad (4.4)$$

We now define a theorem that will be applied on $\mathbf{G}_{\theta^+}$ to locate the blocks through which L_{hl}^+ will pass.

Theorem 4.1. *Given a horizontal edge E defined by two points $E_1(x_1, y_1)$ and $E_2(x_1, y_2)$, such that $y_2 > y_1$, a line $l(\rho, \theta)$ intersects E at a point between E_1 and E_2 if and only if:*

$$\rho_{E_1} < \rho < \rho_{E_2} \quad (4.5)$$

where ρ_{E_1} and ρ_{E_2} are computed for the same angle θ representing l , in the following way:

$$\begin{aligned} \rho_{E_1} &= x_1 \cos \theta + y_1 \sin \theta \\ \rho_{E_2} &= x_1 \cos \theta + y_2 \sin \theta \end{aligned} \quad (4.6)$$

Proof. The (ρ, θ) pairs given by (ρ_{E_1}, θ) and (ρ_{E_2}, θ) represent two parallel lines l_{E_1} and l_{E_2} passing through E_1 and E_2 respectively as shown in Fig. 4.6. If ρ of the given line $l(\rho, \theta)$ is such that $\rho_{E_1} < \rho < \rho_{E_2}$ for the same θ , then l is parallel to l_{E_1} and l_{E_2} and is passing between E_1 and E_2 as shown in Fig. 4.6. Mathematically, the line $l(\rho, \theta)$ can

be represented in its parametric form as:

$$l : \rho = x \cos \theta + y \sin \theta \quad (4.7)$$

Substituting a point (x_1, y_3) in l above, if $\rho_{E_1} < \rho < \rho_{E_2}$, then $y_1 < y_3 < y_2$. $\square$

FIGURE 4.6: Straight lanes and their locations in blocks

Using this theorem, given a line and two points, it can be found if the line passing through a point located on the line joining the two given points.

This theorem is now used to find the blocks through which the left host lane L_{hl}^+ passes. We define two functions $f_x : \Re^{(M_b-m+1) \times N_b} \rightarrow \Re^{(M_b-m+1) \times N_b-1}$ and $f_y : \Re^{(M_b-m+1) \times N_b} \rightarrow \Re^{(M_b-m) \times N_b}$ which map $f_x : \mathbf{G}_{\theta+} \rightarrow \mathbf{G}\mathbf{x}_{\theta+}$ and $f_y : \mathbf{G}_{\theta+} \rightarrow \mathbf{G}\mathbf{y}_{\theta+}$ such that:

$$\begin{aligned} \mathbf{G}\mathbf{x}_{\theta+}(i, j) &= 1 \text{ if } \mathbf{G}_{\theta+}(i, j) < \rho_{hl}^+ < \mathbf{G}_{\theta+}(i, j+1) \\ \mathbf{G}\mathbf{y}_{\theta+}(i, j) &= 1 \text{ if } \mathbf{G}_{\theta+}(i, j) < \rho_{hl}^+ < \mathbf{G}_{\theta+}(i+1, j) \end{aligned} \quad (4.8)$$

According to Theorem 4.1, a 1 at $\mathbf{G}\mathbf{x}_{\theta+}(i, j)$ indicates L_{hl}^+ intersects the horizontal lower edge between the block origins of $B(i, j)$ and $B(i, j+1)$. Similarly a 1 at $\mathbf{G}\mathbf{y}_{\theta+}(i, j)$ indicates L_{hl}^+ intersects the vertical edge between the block origins of $B(i, j)$ and $B(i+1, j)$. For the illustration shown in Fig. 4.5, $\mathbf{G}\mathbf{x}_{\theta+}$ and $\mathbf{G}\mathbf{y}_{\theta+}$ will results in:

$$\mathbf{G}\mathbf{x}_{\theta+} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \quad \mathbf{G}\mathbf{y}_{\theta+} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \quad (4.9)$$

Mapping the rows and columns of $\mathbf{G}\mathbf{x}_{\theta+}$ and $\mathbf{G}\mathbf{x}_{\theta-}$ obtained above, to the axes of the blocks in Fig. 4.5, the path traversed by the left host lane L_{hl}^+ can be deduced as: $B(8, 2) \rightarrow$ cuts the lower edge of $B(7, 2) \rightarrow$ cuts left edge of $B(7, 3) \rightarrow$ cuts the lower edge of $B(6, 3) \rightarrow$ before meeting the vanishing point. In this manner, the exact trajectory of the line can be traced from block to block.

This is the path if the host lane in near-view follows a straight path in far-view. According to the method proposed in Section 4.2, if it is found that the lane is not straight in far-view, the lane will not follow the path as obtained from the above process. In order to check how the road is curving, we try to follow the straight path block-by-block.

Let us now consider the curved lane given by the solid lines as shown in Fig. 4.7. The method is described first for the left lane given by $L_{hl}^+(\rho_{hl}^+, \theta_{hl}^+)$. From the near-view information, the left lane is expected to follow the dotted line (as described above). Therefore, the blocks in far-view region it would traverse are $B(7,2) \rightarrow B(7,3) \rightarrow B(6,3)$. In order to check the curvature, we traverse from one block to the next and

FIGURE 4.7: Curved lane in far-view

check if the lane feature is found at the (ρ, θ) values that describe the lane marking. In the case shown in Fig. 4.7, block $B(7,2)$ is first checked for the presence of line at $(\rho_{hl}^+, \theta_{hl}^+)$. Since there is a line at the specific (ρ, θ) value, the next block $B(7,3)$ is checked. In $B(7,3)$, it can be seen that lane is curving to the left (the solid curve) and is not along the straight path (the dotted line). When Hough transform is applied to the edge map of $B(7,3)$, it will be seen that a peak is found at a different (ρ, θ) . Let this straight line be given by $l_{(7,3)}(\rho_{(7,3)}, \theta_{(7,3)})$. Now Theorem 4.1 is applied to the edges around $B(7,3)$ to check the path traversed by $l_{(7,3)}(\rho_{(7,3)}, \theta_{(7,3)})$. The next block indicated in this process is $B(6,3)$. The whole process is now repeated for $B(6,3)$ with the new (ρ, θ) value given by $l_{(7,3)}(\rho_{(7,3)}, \theta_{(7,3)})$. In other words, the presence of line $l_{(7,3)}$ is checked in $B(6,3)$. On repeating this process, it will be seen that a new line $l_{(6,3)}(\rho_{(6,3)}, \theta_{(6,3)})$ will be found in $B(6,3)$. This becomes the input for the next block that will be processed. This process is repeated till the designated far-view region is covered (more details about this will be discussed in the next sub-section). Fig. 4.8 shows the result of this linear approximation resulting the curvature of the road. If the

left lane curves to the left, the right lane also should curve to the left. This information is used to infer the curvature of the right lane.

FIGURE 4.8: Linear approximation of curved lane in far-view

An illustration of the proposed method on a real road image is shown in Fig. 4.9. The input image scene is shown in Fig 4.9(a). The lane information of the left host lane in near-view is marked in Fig. 4.9(b). The (ρ, θ) tuples for the lane edges are $L^+_{hl}(751, 55^\circ)$ and $L^+_{hl}(744, 58^\circ)$. After applying Theorem 4.1, the block on top left corner is suggested for processing. The edge map of this block is shown in Fig. 4.9(c). It can be seen that the edge map does not show the lane marking in the direction of the host lane. Fig. 4.9(d) shows the Hough space of the edge map giving a peak at $(-80, -58^\circ)$. Fig. 4.9(e) shows this line superimposed on the edge map resulting in the curvature of the lane as shown in Fig. 4.9(f).

4.4 Results & Evaluation

In this section, we first evaluate the effectiveness of the proposed lane detection technique for far-view. Then we discuss the proposed linear approximation based curved lane detection technique.

4.4.1 Straight lane detection in far-view

The proposed technique was applied on images with straight lanes with varying road scenes. Some of the test cases are shown in Table 4.1, which will be used to illustrate the effectiveness of the proposed techniques. Since the proposed technique detects the presence and location of the lanes in far-view using the information obtained from

FIGURE 4.9: Illustration showing linear approximation of curved lane in far-view

the near-view, we show the near-view information for each lane in every test image in Table 4.1 and then list its far-view information. The sub-tables that are shown in the second column in Table 4.1 shows the *lane parameters* of the near and neighbor lanes. By “lane parameters”, we mean the following near-view and far-view lane information:

- $\rho - \theta$ values of the lanes in near-view and far-view, shown by ρ_{nv}, θ_{nv} and ρ_{fv}, θ_{fv} respectively.
- The peak heights p of the lanes in near-view and far-view, shown by p_{nv} and p_{fv} respectively.
- The difference of ρ s, θ s and p s between near and far-view, shown by δ_ρ , δ_θ and δ_p .

These different lane parameters are shown for the left and right host lanes (L_{HL} and L_{HR}), and the left and right neighbor lanes, i.e. L_{HL-1} and L_{HR+1} as shown in Table 4.1.

As discussed in Section 4.2, a lane is detected correctly in far-view if δ_ρ and δ_θ are less than a small deviation and δ_p is high. In other words, the location of the lanes in near and far-view should be same or similar but the Hough accumulation peak height should be high indicating the presence of the same lane in far-view. In Table 4.1, in order to detect the presence of straight lane in far-view, the thresholds for δ_θ and δ_ρ are set to 5° and 20 respectively. In most cases, the angle deviation is less than 5° . The deviation is ρ

[illegible]

4.4.2 Curved lanes in far-view

We now discuss the case of curved lanes. Firstly, it is observed that in highway conditions, the roads are usually straight. It was seen that in over 30 kms of test video that was captured, the curvatures of most curves were too gradual to be considered as curved roads. Also, it was observed that the curvature in far-view occurs more than 30-40 meters away from the host vehicle. In the image plane, the lane features forming such curvatures in far-view are not prominent enough. They are either too faint to be picked up by the edge detection phase or even if the edges are detected, they appear to be like dots which do not form any peaks in the Hough space. This is illustrated in the image shown in Fig. 4.10. The image is divided into grids to show the near-view and far-view blocks. The edge map of each block is shown in Fig. 4.10 shows that the edges are not prominent in the far-view, especially when the curve is too far away in the far-view (more than 30-40 meters).

FIGURE 4.10: Curved road markings in far-view not picked up by edge detection.

Therefore, we can detect the curvature of the road only if the curvature occurs in what we call as the near-far-view. Near-far-view is defined as the lower-half of the blocks in the 7th row. If the curvature is not prominent in this region, we consider that the road is curving (because the far-view straight lane detection is saying so) but the curvature is not traced in such cases.

We will now apply the proposed techniques to detect the curved roads in the near-far-view region, which is defined as region less than 30m in front of the host vehicle. Table 4.2 shows two examples of curved road cases, where the curvature is seen in the near-far-view. We explain the results using (a) in Table 4.2. As discussed in Section 4.3, before looking for the curved lanes, we look for the straight lanes in the far-view. It can be seen from the second column in Table 4.2, that for case (a) the peak height does not show too much of a difference (less than 10) for the right host lane. The deviation

in the peak height for the left host lane is 14 (> 10) indicating that the lane could be straight. However the right lane does not show a peak for the same (ρ, θ) . When such a case is detected, there is a possibility of the lane curving to the right in the far-view. Using the proposed technique of traversing through blocks as discussed in Section 4.3, we detect that the right host lane h_r shows peak at $(818, 68^\circ)$. This indicates that the lane is curving on the right. The figure in Table 4.2 shows the plot of the lines that are detected in the two blocks where the lane edges are prominent enough to deduce that the lane is curving to the right. The reason for the left lane showing a straight nature is also evident from the figure as the left lane is indeed straight partly in the far view. The curvature of the left lane is prominent later in the far-view as indicated by the pink lines in the figure. The other cases in Table 4.2 can also be explained similarly.

TABLE 4.2: Curved lane detection using proposed technique

	ρ_{nv}	θ_{nv}	p_{nv}	ρ_{fv}	θ_{fv}	p_{fv}	δ_ρ	δ_θ	δ_p
h_l	726	70	42	726	70	56	0	0	14
				658	82	32			
h_r	24	-45	66	24	-45	72	0	0	6
				818	68	21			

(a)

	ρ_{nv}	θ_{nv}	p_{nv}	ρ_{fv}	θ_{fv}	p_{fv}	δ_ρ	δ_θ	δ_p
h_r	-39	-49	78	-39	-49	80	0	0	2
				477	-15	25			

(b)

Effect of Block Size for Curved Lane Detection The accuracy of the linear approximation of the curved lane is dependent on the size of the edge block which is being used to find the curve. The block size determines the sensitivity to which the curve can be approximated as a straight line. The actual curvature of the lane and the block size together can be used together for better linear approximation of the curve. This is explained as follows. If the curvature of the lane is large, it implies that lane is curving fast. If an $M_b \times N_b$ sized image block is used to capture such a curve (with large curvature), the linear approximation will not accurately capture the entire curvature, because the entire curvature in the block is being approximated using only one line equation. However, if the image block is further divided into smaller blocks, the curvature of the lane is captured as smaller straight lines across the smaller image block. This is illustrated in Fig. 4.11. It shows an input image having curved right lane in blocks $B(7, 6)$ and $B(8, 6)$, block size being 90×128 . The near view is captured as a straight line in $B(8, 6)$. On applying the proposed straight lane detection in far-view, we see the following (ρ, θ) values for the right lane:

$$\begin{aligned} (\rho_{nv}, \theta_{nv}) &= (-51, 122) \\ (\rho_{fv}, \theta_{fv}) &= (-51, 122) \\ p_{nv} &= 64 \\ p_{fv} &= 62 \end{aligned} \tag{4.10}$$

The above values show that there is no straight lane in the far-view. With the block size set to 90×128 , when we look at $B(7, 6)$, we see a peak at $(\rho, \theta) = (-17, 629)$. Therefore, for a block size of 90×128 , the curved right lane is approximated using the two lines $(-51, 122)$ and $(-17, 629)$ as shown in Fig. 4.11 (d). Now, if we divide $B(7, 6)$ further as shown in Fig. 4.11 (c), on applying the proposed curved lane detection technique in these smaller blocks, we get two lines: $(-33, 406)$ and $(-13, 691)$. These values approximate the curvature better as shown in Fig. 4.11 (e).

This division of the image in smaller blocks is needed only if the curvature of the curve is higher. However, if the curvature is not so curved, larger block sizes give good approximation of the curvature of the road, if straight lanes are found in the far-view. Most highway roads do not show such steep curves as shown in Fig. 4.11 (this image was taken in one of the interior minor roads in the University). More work on the detecting the nature of the curvature and extracting curves in an image using GAH and varying the block sizes is presented in our work in [201].

FIGURE 4.11: Illustration to show the effect of block size on linear approximation of the lane curvature. (a) Input image divided into blocks of size 90×128 , (b) Blocks showing edge map of the curved right lane, (c) Smaller blocks dividing the curvature further, (d) Linear approximation of the curve using bigger blocks, (e) Linear approximation of the curve using smaller blocks.

4.5 Computational Challenges

The proposed techniques discussed in the previous sections involve block-based Hough transform computation on the edge maps generated by the GMH-GAH-HT engines in every block. However, processing them serially in a block based manner is computationally inefficient. In the case of straight lane detection for far-view, Hough transform is computed for 2 or sometimes 3 rows of blocks. And thereafter, the Hough spaces of each block are summed together. Since the edge maps of each block are available together, computing the HT of all these edge maps in a parallel fashion can improve the computational efficiency. Computing the HT at block level in parallel using the conventional HT method is common. However, this is computationally inefficient as every block needs to apply the HT equation. The computational complexity can be illustrated using the following sample scenario. Let us recall the HT computation equation:

$$\rho = x \cos \theta + y \sin \theta \quad (4.11)$$

Therefore, every HT computation involves 2 multiplications, 2 trigonometric operations and 1 addition. For a given block of size $M_b \times N_b$ with $p \times (M_b N_b)$ number of edge

pixels, the total computations for computing the HT of edge block equal:

$$\begin{aligned}
 &2pM_bN_b \text{ multiplications} \\
 &2pM_bN_b \text{ trigonometric operations} \\
 &pM_bN_b \text{ additions}
 \end{aligned} \tag{4.12}$$

These are the total number of operations for one block and ‘one’ angle in Hough space. For $N_\theta = 180$ angles, we will have an addition N_θ factor in the above terms. For two rows of 10 blocks each, we need 20 such processing engines, which compute the above HT equation N_θ times. Therefore, it can be seen that parallelizing the HT computation across blocks using conventional means can be computationally costly.

Another major challenge is that each of the blocks will give the Hough accumulation spaces (for N_θ angles) that are local to the image block coordinates. The Hough spaces need to be first normalized to the origin of the entire image and then added together (as shown in Fig. 4.4 for far-view lane detection). This involves enormous amount of computation resources. An estimate of the computation resources in shifting the Hough spaces to image origin and then adding them can be derived as follows. Assume we have an image of size 720×1280 , then the $\rho \times \theta$ Hough space size for 180 angles is typically given by 2900×180 . If the image is divided into blocks of size say 90×128 , then the block level Hough space is of size 312×180 . Each entry in the block level space (per angle) needs to be mapped first to Hough space of the entire image. Therefore, 312×180 memory accesses are needed. Thereafter, every entry in the 2900×180 for every block is added to each other. If there are 16 blocks, the total number of additions would be nearly 8.35 million. Therefore, a serial combination of these Hough spaces is computationally inefficient.

In the sections that follow, these two challenges will be addressed. Firstly, a novel computation efficient method called the Additive Hough Transform (AHT) is proposed to compute HT of the edge blocks in *parallel*. This method generates Hough spaces for every block in a parallel fashion, that are already normalized to the image origin. Next, the properties of AHT are further extended to develop a hierarchical technique to combine the block-level Hough spaces in a computationally efficient manner.

4.6 Parallelizing Hough Transform Computation

Most existing HT computation methods are not parallelized to compute the HT of several pixels simultaneously because they are constrained by serial reading of the edge map itself. The parallel implementations either parallelize the angle space but not the pixel

coordinates, or employ extensive mesh arrays and pyramidal processors. In this section, an efficient block-based parallel implementation is proposed based on a novel parallel HT computation method called the Additive Hough Transform (AHT), first introduced by us in [202]. It is shown that the proposed method yields significant performance gains. Further, optimizations for managing area costs are also proposed. The proposed AHT based design is compared against some of the existing HT computation implementations to evaluate the timing, area cost and area-time tradeoffs.

4.6.1 Parallel Additive Hough Transform (AHT)

We propose a novel algorithm that lends well for parallelizing the computation of HT. Let us recall the Hough transform computation as shown below:

$$\rho = x \cos \theta + y \sin \theta \quad (4.13)$$

Using the above equation, we propose what we call as the additive property of HT as the following. Referring to Fig. 4.12, the additive property of HT is given by:

$$HT(A, O) = HT(A, B) + HT(B, O), \quad (4.14)$$

where A and B are two points in the image space, O is the origin and $HT(X, Y)$ represents the HT of point X with respect to (w.r.t) point Y . This implies that the Hough transform of point A w.r.t image origin O is the summation of the Hough transform of A w.r.t a local point B and the Hough transform of the local point B w.r.t O . This property can be proved in the following way. Let l_1 and l_2 be two parallel lines passing through points A and B respectively. Since $l_1 \parallel l_2$, the angle made by the perpendicular ($\perp$) from origin O to l_1 with the x -axis is equal to the angle subtended by the $\perp$ from B to l_2 as shown in Fig. 4.12. If we consider B as origin, for the given angle θ , the $\perp$ from B to l_1 is the HT of A w.r.t B , i.e. $HT(A, B)$. Also, the $\perp$ from the image origin O to l_2 is the HT of B w.r.t O , i.e. $HT(B, O)$. From Fig. 4.12, since $l_1 \parallel l_2$, the HT of A w.r.t O , i.e. $HT(A, O)$ is equal to the summation of $HT(A, B)$ and $HT(B, O)$. The additive property is illustrated in detail with an example in [202]. For the rest of the section, the HT of a point A w.r.t a local point B , i.e. $HT(A, B)$, is referred as ‘local HT’ and the HT of A w.r.t image origin O , i.e. $HT(A, O)$, is referred as ‘global HT’.

This additive property of HT can be exploited to compute the HT of an edge map in a parallel fashion. We call the proposed parallel computation technique as the Additive Hough Transform (AHT). A binarized edge map is considered as the input for HT computation wherein a pixel value of ‘1’ indicates an edge pixel and ‘0’ indicates a non-edge pixel. For the sake of explanation, we initially assume $m \times m$ sized images, where

FIGURE 4.12: Additive property of the Hough transform; l_1 and l_2 are two parallel lines passing through A and B respectively.

m is a power of two, such as 64×64 , 128×128 and so on. The derivations are later extended to $m \times n$ sized images ($m \neq n$) commonly used in practical video processing applications.

FIGURE 4.13: Edge map divided into 16 blocks using a 4×4 grid and showing spatially equivalent points in two different blocks.

In order to explain the AHT, let us consider an $m \times m$ sized edge map that is divided into k^2 blocks using a $k \times k$ grid, where k is the grid resolution. An example is shown in Fig. 4.13 for a 4×4 grid. For now, let us assume m is divisible by k . The case for m not divisible by k will also be considered later. For each block $b(u, v)$, the bottom-left pixel coordinate is considered as the local origin $O_{b(u,v)}$. The global HTs of the local origins are given by $HT(O_{b(u,v)}, O)$, where O is the image origin. If we take a point P in a block $b(u, v)$, the local HT of P w.r.t the local origin $O_{b(u,v)}$ is given by $HT(P, O_{b(u,v)})$. According to the additive property of HT, the global HT of P in $b(u, v)$ can be computed by summing $HT(P, O_{b(u,v)})$ and $HT(O_{b(u,v)}, O)$.

Observation 1: Points from different blocks with the same relative position w.r.t their local origins have the same local HTs. For example, in Fig. 4.13, $HT(P, O_{b(u,v)}) = HT(P', O_{b(m,n)})$ and $HT(Q, O_{b(u,v)}) = HT(Q', O_{b(m,n)})$. This implies that the local

HTs can be computed for only one block and the same values can be reused to compute the global HTs of all points in the entire edge map.

We now formulate the parallel AHT technique using the following set of equations. Let us consider a $k \times k$ matrix $G^{(\theta)}$, which has the global HTs of the local origins of all k^2 blocks. $G^{(\theta)}$ is defined as:

$$G^{(\theta)} = \begin{bmatrix} G_{b(0,k-1)}^{(\theta)} & G_{b(1,k-1)}^{(\theta)} & \cdots & G_{b(k-1,k-1)}^{(\theta)} \\ \vdots & \vdots & \ddots & \vdots \\ G_{b(0,1)}^{(\theta)} & G_{b(1,1)}^{(\theta)} & \cdots & G_{b(k-1,1)}^{(\theta)} \\ G_{b(0,0)}^{(\theta)} & G_{b(1,0)}^{(\theta)} & \cdots & G_{b(k-1,0)}^{(\theta)} \end{bmatrix}, \quad (4.15)$$

where $G_{b(u,v)}^{(\theta)}$ is the global HT of local origin $O_{b(u,v)}$ of block $b(u,v)$ for a given angle θ in the Hough space. When an $m \times m$ sized edge map is divided into blocks using $k \times k$ grid, we get blocks of size $(m/k) \times (m/k)$. Let $L^{(\theta)}$ be an $(m/k) \times (m/k)$ sized matrix comprising of the local HTs of all the $(m/k)^2$ pixels in the block w.r.t to the local origin. From Observation 1, $L^{(\theta)}$ is computed for only one block and not all the k^2 blocks. $L^{(\theta)}$ is written as:

$$L^{(\theta)} = \begin{bmatrix} L_{(0, \frac{m}{k}-1)}^{(\theta)} & L_{(1, \frac{m}{k}-1)}^{(\theta)} & \cdots & L_{(\frac{m}{k}-1, \frac{m}{k}-1)}^{(\theta)} \\ \vdots & \vdots & \ddots & \vdots \\ L_{(0,1)}^{(\theta)} & L_{(1,1)}^{(\theta)} & \cdots & L_{(\frac{m}{k}-1,1)}^{(\theta)} \\ L_{(0,0)}^{(\theta)} & L_{(1,0)}^{(\theta)} & \cdots & L_{(\frac{m}{k}-1,0)}^{(\theta)} \end{bmatrix}, \quad (4.16)$$

where $L_{(i,j)}^{(\theta)}$ is the local HT of the pixel coordinate (i,j) w.r.t the leftmost bottom pixel position considered as local origin for a given angle θ in the Hough space.

Now, from the $m \times m$ sized edge map E , we generate a $k \times k$ matrix $I_{(i,j)}$ consisting of binary pixels from E which are sampled at an interval of k . In other words, $I_{(i,j)}$ contains k^2 edge pixels, one from each of the k^2 blocks in E , which are located at the same relative position (i,j) w.r.t their respective local origins. $I_{(i,j)}$ matrix is defined as:

$$I_{(i,j)} = \begin{bmatrix} E_{(i,j+(k-1)k)} & \cdots & E_{(i+(k-1)k,j+(k-1)k)} \\ \vdots & \ddots & \vdots \\ E_{(i,j+(k-1))} & \cdots & E_{(i+(k-1)k,j+(k-1))} \\ E_{(i,j)} & \cdots & E_{((i+(k-1)k),j)} \end{bmatrix}, \quad (4.17)$$

where $0 \leq \{i,j\} < m/k$. Using (4.15), (4.16) and (4.17), the parallel Additive Hough Transform technique can be mathematically written as

$$H_{(i,j)}^{(\theta)} = G^{(\theta)} \cdot I_{(i,j)} + L_{(i,j)}^{(\theta)} \cdot I_{(i,j)}, \quad (4.18)$$

where $H_{(i,j)}^{(\theta)}$ is a $k \times k$ matrix consisting of global HTs of the k^2 edge pixels in $I_{(i,j)}$. In (4.18), for a given (i, j) and θ , k^2 pixels in E from k^2 blocks are fetched into $I_{(i,j)}$. The scalar value $L_{(i,j)}^{(\theta)}$ from $L^{(\theta)}$ matrix gives the local HT of all the k^2 points in $I_{(i,j)}$ for θ , because all these k^2 pixels are at the same relative position (i, j) w.r.t their local origins. The $\cdot$ operator in (4.18) represents element-wise multiplication. A one in $I_{(i,j)}$ indicates an edge pixel for which the global HT has to be computed. So in (4.18), for an edge pixel in $I_{(i,j)}$, the local HT $L_{(i,j)}^{(\theta)}$ is added to the global HT of its local origin in $G^{(\theta)}$ to give the global HT of that edge pixel for θ in the Hough space. Thus the AHT enables computation of global HTs of k^2 edge pixels in parallel. As (i, j) is scanned from $(0, 0)$ to $((m/k) - 1, (m/k) - 1)$, the Hough space of the entire edge map E for the angle θ is computed in $(m/k)^2$ iterations.

Fig. 4.14 shows the edge maps of two 256×256 sized images and their corresponding Hough spaces generated using conventional HT and the proposed AHT techniques. For each test image, the Hough spaces from both the methods were compared and it was found that they are identical, thereby validating the functionality of the proposed AHT technique.

FIGURE 4.14: Edge maps of two test images in (a) and (b) followed by Hough spaces generated by conventional HT in (c) and (d) and the proposed AHT in (e) and (f).

4.6.2 Block-based AHT Computation Architecture

In this subsection, we derive a parallel implementation of the proposed AHT technique. Since the computation of HT in (4.13) depends only on the image coordinates and angles in the Hough space, $G^{(\theta)}$ and $L^{(\theta)}$ in (4.15) and (4.16) are constants and can be

precomputed for all θ s in the Hough space. It should be noted that $L^{(\theta)}$ is computed for one $(m/k) \times (m/k)$ sized block only and each value in $L^{(\theta)}$ is used to compute HTs of k^2 pixels. LUTs are used in the proposed implementation to store only the $L^{(\theta)}$ matrices for all angles. As it will be shown later, the $G^{(\theta)}$ matrices do not incur any additional memory cost.

Firstly, the block size is given by $(m/k) \times (m/k)$ and hence the total LUT size required to store $L^{(\theta)}$ matrices for all n_θ angles in Hough space is equal to $A_L = (m/k)^2 n_\theta$ words. We reduce this LUT space by nearly 50% by exploiting the symmetry of coordinates in a square block based on the proposed theorem below:

Theorem 1: In an $l \times l$ sized image with origin at bottom-left corner pixel,

$$\begin{aligned}\rho(x, y, \theta) &= \rho(y, x, 90^\circ - \theta) \\ \rho(x, y, 90^\circ + \theta) &= -\rho(y, x, 180^\circ - \theta),\end{aligned}\tag{4.19}$$

where $(x, y) \in \{(x, y) | x \leq l-1, y \leq l-1, y \leq x\}$, angle $0 \leq \theta \leq 90^\circ$ and $\rho(a, b, \theta) = a \cos\theta + b \sin\theta$.

Proof: Using (4.13) and trigonometric equalities for complimentary and supplementary angles, the above two equations can be proved for the ranges stated for θ and (x, y) .

This theorem and its implication on memory savings are illustrated using an 8×8 block in Fig. 4.15. Each square represents a pixel and the coordinates of the shaded squares along the diagonal satisfy the relationship $x = y$. The coordinates of all the squares, which are on and below this diagonal (indicated by crosses) satisfy $y \leq x$. According to Theorem-1, it is sufficient to store the HTs of only these crossed pixel locations. The HTs of the symmetrically opposite pixel locations (the diagonal being axis of symmetry) in the upper half can be found from the HTs of their corresponding mirror pixel coordinates in the lower half. This implies that the local HTs of only about half the block need to be precomputed and stored. This reduces the size of the LUT from $(m/k)^2 n_\theta$ to $((m/k)^2 + m/k)n_\theta/2$ words, where n_θ is the number of angles in the

FIGURE 4.15: An 8×8 block to illustrate Theorem 1. The local HTs of pixel coordinates marked with $\times$ only need to be stored.

Hough space. For example, for the 8×8 block in Fig. 4.15, the LUT space reduces from $64n_\theta$ words to $36n_\theta$ words.

The proposed LUT-based parallel implementation is shown in Fig. 4.16 for $k = 4$. It comprises a bank of LUTs and a two-dimensional (2-D) computation array of processing elements (PEs). The LUT-bank has n_θ LUTs. Each LUT stores the local HTs of coordinates in the lower half of the block only (applying Theorem - 1).

The 2-D computation array has k^2 PEs. In Fig. 4.16 for $k = 4$, there are sixteen PEs where each PE is associated with its spatially corresponding block of pixels in the edge map. Matrix $I_{(i,j)}$ in (4.17), consisting of k^2 spatially equivalent binary pixels for k^2 blocks, is fed as an input to this array. (i, j) is also used to generate the address for the LUT-bank. The LUT-bank outputs n_θ number of $L_{(i,j)}^{(\theta)}$ local HT values for the coordinate position (i, j) , one for each of the n_θ angles. These are also sent as inputs to the computation array. Thus, a single read per angle results in the local HT value to be sent to all k^2 PEs in the computation array.

Each PE in the array is a bank of n_θ constant adders (CA) as shown in Fig. 4.16. The square box $\square$ below the CA in Fig. 4.16 represents the output register to hold the ρ values. Using a constant adder serves two purposes. Firstly, a CA in $PE(u, v)$ acts a memory element to store the global HT of the local origin of the corresponding block $b(u, v)$ for a given θ . This is possible because in the AHT, for a given image size, the global HTs of the local origins are constants. In other words, $G_{b(u,v)}^{(\theta)}$ in (4.15) is stored as a constant input to the adder. Using a CA helps to eliminate the need for a separate

FIGURE 4.16: Proposed AHT-based parallel implementation.

LUT to store $G^{(\theta)}$ matrices. The second advantage of using a CA is that it reduces the hardware cost by almost 50% due to reduced combinational logic [203].

To summarize, in every clock cycle, the k^2 PEs receive k^2 pixels from the edge map in I . They also receive the local HTs for all angles for a given local coordinate position (i, j) , i.e. $L_{(i,j)}^{(\theta)}$ s. A ‘1’ in $I_{(i,j)}$ indicating an edge pixel is used as an enable signal to constant adders that add the local HTs to the global HTs. This process results in global HTs for all k^2 pixels and for all n_θ angles in parallel.

In the case when m is not a perfect multiple of k , each block in the interior of the grid is set to have $\lceil m/k \rceil \times \lceil m/k \rceil$ pixels. The blocks at the boundary will then have lesser number of pixels. However, this does not affect the computations in the proposed implementation. The only difference in this case is that the PEs at the boundaries of the computation array have to be either fed with zeros or shut down for missing pixel positions, i.e. for pixel positions exceeding the image size. It is important to note that the PEs still function as explained above for all the ‘valid’ pixel positions.

Although the proposed implementation of the AHT is LUT-based, variations using alternate computing engines like CORDIC or DA are possible, depending on user requirements like scalability and flexibility. The distinct advantage of our approach is that, even if the LUT is replaced with alternate computing engines such as CORDIC and DA, the order of speed-up will be similar, as they are based on the AHT.

4.6.3 Performance & Computation Cost Evaluation of AHT

In this section, we analyze the total computation time T , total area cost A and area-time product AT of the proposed AHT based implementation. These metrics are specifically compared with different kinds of efficient hardware implementations of HT computation like iterative and pipelined CORDIC [80, 82], parallel-angle serial addition [83], DA-based [84], multi-sector based [98] and gradient angle based [93]. The analysis is done on application specific HT computation implementations only and does not include HT computation on reconfigurable mesh array processors like PARBUS, NCUBE and RMESH [85, 86].

Computation Time

After a detailed study of the different HT computation implementations in literature, it was found that all the architectures in [79–84, 92, 93] read the edge map serially in a raster scan method. Though the implementations in [81, 83, 84, 98] are parallelized to compute the HT of different angles, they still read the edge map serially. Consequently,

irrespective of whether the pixel is an edge pixel or not, m^2 cycles are needed to read an $m \times m$ sized edge map. In addition, for every edge pixel, α additional cycles are required to compute the HT. If p is the fraction of edge pixels in the edge map such that $0 \leq p \leq 1$, the total computation times of these techniques can be generalized using the following expression:

$$T = m^2 + \alpha p m^2 = (1 + \alpha p) m^2. \quad (4.20)$$

Table 4.3 lists T of the various HT computation implementations that are chosen for comparison. It can be seen that α is either a constant or a function of the number of angles n_θ in the Hough space. For the CORDIC based implementations in [80, 82], α also depends on the number of iterations n_c of the CORDIC engine. In the parallel multi-sector based architecture in [98], α is given by $C_{rescale} + M_{steps}$, which depends on the number of angle spaces and the processors used to parallelize HT computation across angles in Hough space. In the gradient based approach [93], where HT is computed only for the gradient angle in a pipelined manner, a pipeline latency of N_s is incurred, as shown in Table 4.3.

In the proposed implementation based on the AHT, not only are the ρ values for all the n_θ angles computed in parallel, the ρ values of k^2 pixels are also obtained in parallel. For all the k^2 pixels located at the same relative position in the blocks, the addition of local HT to the global HTs in the computation array takes one cycle. So $\lceil (m/k)^2 \rceil$ cycles are needed for computing the HT of the entire edge map. The reading of the local HTs from the LUTs and k^2 edge pixels from memory units (which store the sub-edge maps) can be pipelined with the HT computation. This would lead to a pipeline latency of one cycle in addition to the $\lceil (m/k)^2 \rceil$ cycles. The ceiling function is necessary for cases when m is not divisible by k . It can be seen from Table 4.3 that T is reduced by a factor of at least k^2 as compared to existing implementations.

TABLE 4.3: Total Hough Transform Computation Times

Architectures	Total Computation Time T (cycles)
Iterative CORDIC-based [80]	$\{1 + \frac{n_c n_\theta}{2} p\} m^2$
Pipelined CORDIC-based [82]	$\{1 + (\frac{n_\theta}{2} + n_c) p\} m^2$
Parallel angle serial addition based [83]	$(1 + p) m^2$
Parallel angle DA based [84]	$(1 + \log_2 m p) m^2$
Parallel angle multi-sector based [98]	$\{1 + (C_{rescale} + M_{steps}) p\} m^2$
Gradient-based pipelined [93]	$m^2 + N_s$
Proposed AHT-based	$(\lceil \frac{m}{k} \rceil)^2 + 1$

Fig. 4.17 plots the total computation time T against different image sizes $m \times m$ for

the different implementations. n_c , n_θ , $C_{rescale}$, M_{step} and N_s are taken to be 16, 180, 6, 50 and 4 respectively based on the values set in [80, 82, 98]. It is assumed that 10% of the edge map has edge content, i.e. $p = 0.1$ [82]. The grid resolution is taken as 8, i.e. $k = 8$, for the proposed implementation. The computation time T on y-axis is plotted in log scale. Fig. 4.17 shows that the proposed parallel implementation is faster than implementations in [80, 82–84, 204] by as high as 4 orders of magnitude for all image sizes.

FIGURE 4.17: Comparison of total computation time (T) in cycles vs. image size $m \times m$.

Fig. 4.17 shows the speedup of the AHT for a grid resolution of 8 only. However, the speed up can be increased further by increasing the grid resolution k . This is because increasing k introduces more parallelism. This dependency of T on k is shown in Fig. 4.18. It can be seen that for all the image sizes, the performance of the proposed implementation can be increased continuously by incrementing k . The ceiling function for T is responsible for the step-like decrease, observable for lower values of m like 128 with high values of k . Though the performance increases continuously with increase in k , the area cost of the implementation escalates. The area-time tradeoffs of the proposed implementation are detailed in Section 4.6.3.

The minimum permissible clock period of the proposed implementation is also less than the implementations in [80, 82–84, 93, 98]. Table 4.4 shows the critical path estimates of the different HT computation implementations. In order to compute the delay in Table 4.4, the following assumptions are made. The memory access times are ignored for all implementations. All 2-operand w -bit adders are implemented as ripple-carry adders (RCA). In RCA, the delay is contributed by the carry chain [203]. The delay of a 1-bit full adder is taken from one of the inputs to the carry output resulting in

FIGURE 4.18: Total computation time (T) in cycles in log scale vs. grid resolution k for the proposed AHT-based implementation. For smaller m like 128, step-like decrease is seen because of the ceiling function introduced to cater for cases where k is not a factor of m .

four 2-input gate delays. Therefore the delay of a w -bit 2-operand adder is equal to $4w$ gate delays. A constant adder is implemented as a chain of half adders, which results in a delay of $2w$ gate delays for a w -bit constant adder. A barrel shifter, which is used for single cycle multiple bit shift in [80, 98], is assumed to be made of layers of 2-to-1 multiplexers. Each 2-to-1 multiplexer is assumed to have a delay of 2 gates resulting in a total delay of $2\log_2 n_c$ gate delays for each barrel shifter, where n_c is the number of iterations. It can be seen that the proposed implementation can operate with the fastest clock as compared to all other implementations.

TABLE 4.4: Critical Paths and Delays

Architecture	Critical Path	Delay in 2-input gates
Iterative CORDIC-based [80]	One barrel shifter + one 2-operand adder	$2\log_2 n_c + 4w$
Pipelined CORDIC-based [82]	Two 2-operand adders	$8w$
Parallel angle serial adder-based [83]	Two 2-to-1 multiplexers + 2-operand adder	$4 + 4w$
Parallel angle DA-based [84]	One 2-operand adder	$4w$
Parallel multi-sector based [98]	One barrel shifter + one 2-operand adder	$2\log_2 6 + 4w$
Gradient-based [93]	One 2-operand adder	$4w$
Proposed AHT-based	One constant adder	$2w$

TABLE 4.5: Number of Computation Elements and Area Cost

Design	Components			
	Adders/ Subtractors	Registers	Multiplexers	Memory bits
Iterative CORDIC-based [80]	6	6	$2\log_2 n_c$	$wn_c + m^2$
Pipelined CORDIC-based [82]	$6n_c$	$4n_c$	-	$wn_c + m^2$
Parallel angle serial addition based [83]	$3n_\theta$	$3n_\theta$	$5n_\theta$	$2n_\theta w + m^2$
Parallel angle DA based [84]	n_θ	n_θ	-	$4n_\theta w + m^2$
Parallel multi-sector [98]	16	16	$16\log_2 6$	m^2
Gradient based [93]	$2w + 1$	$2w + 5$	2	$2n_\theta w + m^2 w + m^2$
Proposed AHT based	$\frac{n_\theta k^2}{2}$ $+4k^2 \lceil m/k \rceil^2$	$n_\theta k^2$	-	$\frac{1}{2} \left(\lceil \frac{m}{k} \rceil^2 + \lceil \frac{m}{k} \rceil \right) n_\theta w$

Design	Total Gate Count
Iterative CORDIC-based [80]	$(90 + 8\log_2 n_c +) w + 4wn_c + 4m^2$
Pipelined CORDIC-based [82]	$78n_c w + 4wn_c + 4m^2$
Parallel angle serial addition based [83]	$73n_\theta w + 4m^2$
Parallel angle DA based [84]	$23n_\theta w + 4m^2$
Parallel multi-sector [98]	$406w + 4m^2$
Gradient based [93]	$(30w + 47 + 8n_\theta + 4m^2)w + 4m^2$
Proposed AHT based	$\frac{n_\theta w}{2} \left(21k^2 + 4 \lceil \frac{m}{k} \rceil^2 + 4 \lceil \frac{m}{k} \rceil \right)$ $+4k^2 \lceil m/k \rceil^2$

Area Cost Analysis

For the implementations in [80, 82–84, 93, 98], we derive expressions for area in terms of the number of adders, registers, 2-to-1 multiplexers and memory bits needed for the HT computation module. Table 4.5 lists the number of computation elements in each of implementations. All adders, registers and 2-to-1 multiplexers are assumed to be w bits wide. The memory cost includes the memory to store the edge maps. All the implementations in [80, 82–84, 93, 98] store the binary edge map as m^2 bits. In the proposed implementation k^2 binary sub-edge maps are stored resulting in a total of $k^2 \lceil m/k \rceil^2$ memory bits. The computation array comprises k^2 PEs and each PE contains n_θ constant adders (CAs) and registers. We have discussed in Section 4.6.2 that CAs not only help in storing the global HTs but they also reduce the hardware cost. This is because the Boolean expressions of sum and carry from a 1-bit full adder are simplified to that of a half-adder if one of the inputs is a constant. Employing CAs instead of full-adders results in the reduction of total adder cost from $n_\theta k^2$ to $n_\theta k^2/2$. Also, as discussed in Section 4.6.2, Theorem-1 is used to reduce the memory space to store the local HTs by nearly half from $\lceil m/k \rceil^2 n_\theta$ words to $(\lceil m/k \rceil^2 + \lceil m/k \rceil)n_\theta/2$ words.

The last column in Table 4.5 gives the total area cost in terms of number of 2-input gates for all the implementations. The gate counts of a 1-bit FA, 2-to-1 multiplexer,

1-bit register and 1-bit memory are assumed to be 9, 4, 6 and 4 two-input NAND gates respectively [205]. The total area cost of the proposed implementation is higher than the other implementations because of the parallelism introduced in two ways - the number of angles for which HT is being computed in parallel and the number of pixels that are being processed in parallel using the proposed AHT method.

Area-Time Tradeoff

Usually, parallelizing an algorithm increases its throughput depending on the level of parallelism at the cost of higher area. This trend was observed in the proposed implementation also, as shown in the previous sections. The degree of parallelism is controlled by the grid resolution k . As shown in Fig. 4.18, the total computation time decreases as k increases. To analyze the effect of k on area, we plot the area cost versus grid resolution in Fig. 4.19. It can be seen that as k increases the area cost first drops and then starts to increase. This is because of a push-pull effect introduced by k on the area costs of LUT space and computation array. For lower values of k , the block size is bigger and results in larger LUT space but lesser adder/register cost in the computation array. As k increases, the LUT space reduces but the area occupied by the adders and registers in the PEs increases. Fig. 4.20 illustrates these trends for a 256×256 sized edge map. From Fig. 4.18 and 4.19, the performance of the proposed implementation can be increased by increasing the degree of parallelism (the grid resolution) but there is an optimal k where the area cost is minimal. Table 4.6 lists the optimal grid resolutions k_{opt} for the different image sizes that are obtained from the area cost plot in Fig. 4.19. The area cost starts to increase as we traverse along the grid resolution axis in either direction away from the k_{opt} values listed in Table 4.6.

FIGURE 4.19: Area of the proposed implementation plotted against grid resolution k .

FIGURE 4.20: LUT memory area and computational array area contributing to the total area cost curve for 128×128 sized edge map plotted against grid resolution k to show the push-pull effect on area cost. All area costs are indicated in the number of gates as shown in Table 4.5.

TABLE 4.6: Optimal Grid Resolutions for $m \times m$ Sized Edge Maps

Image Size m	Optimal Grid Resolution k_{opt}
128	8
256	10
512	16
1024	21
2048	29

VLSI Metric - Area $\times$ Time: The AT measure is defined as:

$$AT = Area \times (Total\ cycles). \quad (4.21)$$

The AT measure for the proposed implementation for varying image and grid resolutions is plotted in Fig. 4.21. Although the area A shows an increasing trend for values of k beyond k_{opt} , the AT measure continues to decrease for the entire range of k considered. This trend can be explained from the following observation. For a 128×128 edge map, as k increases to 12, which is beyond the optimal grid resolution of 8, the computation time reduces by 2.12 times whereas area increases by 1.45 times. This implies that the rate of decrease in T of the proposed implementation is higher than the rate of increase in A , which results in decreasing trend of AT as k increases. Similar rates of change were found for other cases of m and k also. It should be noted that the optimal grid resolution corresponding to the least A value in Fig. 4.19 does not necessarily give the lowest AT measure. So if area is not a constraint in designing the implementation, the grid resolution can be increased beyond k_{opt} to get lesser AT .

The AT measure of the proposed implementation is compared against the different implementations in Fig. 4.22 for varying image sizes. For the proposed implementation, AT is determined for the optimal grid resolution with respect to area, i.e. k_{opt} , for each image size. The percentage AT savings was found to be at least 43% and as high as 90%. If a higher grid resolution is used, the AT measure can be decreased further by trading area for performance. For example, for $m = 128$, AT of the proposed implementation at $k_{opt} = 8$ is lesser than its closest competitor - pipelined CORDIC-based design by a factor of 5.4 times. This increases to 7.9 times when the grid resolution is increased to 12.

FIGURE 4.21: Comparison of AT measure of the proposed implementation against different implementations for varying m and k .

The extreme cases of the proposed method are obtained when $k = 1$ and when $k = m$. Setting $k = 1$ means the entire image is the only block. This is equivalent to having a serial implementation and hence the benefits of the parallel AHT technique are not visible with this setting. When $k = m$, we get the other extreme case of the proposed implementation wherein each block contains one pixel only. Although it may appear that in such a case, a large ROM storing the global HTs of all m^2 pixel coordinates can replace the PEs, such a proposition is not practicable due to severe constraints for parallel read access to obtain the m^2 HT values from the ROM.

Analysis for Rectangular Images

So far, the AHT implementation was discussed for square $m \times m$ images using square $k \times k$ grids, resulting in square blocks which are processed in parallel. k was chosen to be the parameter with respect to which optimal area costs were derived. We now extend the implementation for any given image size of $m \times n$, specifically for those used in standard video processing applications. It was shown in Theorem-1 in Section

FIGURE 4.22: VLSI metric AT in log scale vs. image size $m \times m$. AT measure of the proposed implementation is computed at optimal grid resolution k_{opt} .

4.6.2 how the properties of a *square* block can be exploited to optimize LUT costs. For the same reason, we maintain the division of the edge map into square blocks even for the rectangular images. This however would mean that the grid is not necessarily of dimension $k \times k$ grid. Let us say the grid is now of size $k \times l$.

Now, instead of rewriting and optimizing the area-time expressions in terms of two parameters k and l , we derive the expressions in terms of block size. We assume the $m \times n$ edge map is divided into $b_s \times b_s$ sized square blocks with the number of pixel points equal to b_s^2 , denoted by S_b . The total computation time T required to compute the HT of the entire edge map using the AHT is now equal to the number of points in the square block, i.e. $T = S_b = b_s^2$. The area cost of the proposed implementation is reformulated in terms of S and b_s . The number of PEs in the computation array is equal to $\lceil m/b_s \rceil \cdot \lceil n/b_s \rceil$. In the case when m and n are not perfect multiples of b_s , the number of pixels in the boundary blocks of the edge map is lesser than S_b . The boundary PEs can be fed either with zeros or shut down for missing pixel positions. Each PE has n_θ constant adders and registers. The LUT size to store the local HTs of the square block is given by $n_\theta w(b_s^2 + b_s)/2$. The memory to store the edge map is equal to $\lceil m/b_s \rceil \cdot \lceil n/b_s \rceil b_s^2$. So the total area cost of the proposed implementation for an $m \times n$ sized edge map can be expressed as:

$$A = \frac{n_\theta w}{2} \left(21 \left\lceil \frac{m}{b_s} \right\rceil \left\lceil \frac{n}{b_s} \right\rceil + 4S_b + 4b_s \right) + 4 \left\lceil \frac{m}{b_s} \right\rceil \left\lceil \frac{n}{b_s} \right\rceil b_s^2 \quad (4.22)$$

Similar to the case of $m \times m$ sized edge maps, the area expression above also shows a push-pull affect with respect to the image size and the block size. As b_s increases,

the number of PEs starts to decrease but the amount of LUT space increases. There is a optimal block size b_{opt} where the area cost is minimum. (4.22) is used to compute the area cost of the proposed implementation for varying image sizes to determine the optimal block sizes. Table 4.7 lists the b_{opt} s for different image resolutions [206]. The area cost of the proposed implementation in terms of 2-input gates is also indicated for the optimal block size. Fig. 4.23 compares the total computation times of the proposed implementation against existing implementations for these image resolutions. b_{opt} is used to compute timing for the proposed implementation. It can be seen the proposed parallel implementation is orders of magnitude faster than existing designs to compute the HT of the entire edge map. The performance can be further increased by reducing the block size but at the cost of higher area. In Fig. 4.24, the optimal block size for minimum area is used again to compute and compare the AT measure of the proposed implementation. The AHT-based parallel implementation reduces the AT measure by atleast 3.3 times.

TABLE 4.7: Optimal Block Size b_{opt} for Image Sizes in Video Applications

Application	Image resolution $m \times n$	b_{opt}	Area 2-input gates
Slow motion video	176×120	20	4138560
QCIF format	176×144	18	4492800
Video conference/ LL MPEG-2 spec.	352×240	24	8337600
ML MPEG-2 spec.	352×288	24	9313920
ML MPEG-2 spec.	720×480	40	17360640
ML MPEG-2 spec.	720×576	36	19008000
HDTV	1280×720	48	29527200
H1140 MPEG-2 spec.	1440×1080	54	39735360
H1140 MPEG-2 spec.	1440×1152	58	41558720
HL MPEG-2 spec.	1920×1080	60	46794240
HL MPEG-2 spec.	1920×1152	55	48860700
SHD images	2000×2000	69	69268644

4.7 Hierarchical AHT (HAHT) for Efficient Global Hough Space Generation

In Section 4.2, it was shown that the block based Hough spaces are added to generate a global Hough space in order to detect long straight lanes in far-view. The parallel AHT computation technique discussed in the previous section generates these Hough spaces in parallel with low computation cost and high speed. In this section, we propose an architecture that efficiently combines these block-level Hough spaces to generate a global Hough space. In order to do this, we extend the proposed additive property to compute HT at multiple hierarchy levels. This property is then used to derive what we

FIGURE 4.23: Comparison of total computation time in cycles (log scale) of the proposed implementation against existing designs for different video processing applications.

FIGURE 4.24: Comparison of VLSI metric AT in log scale vs. image size $m \times n$ for different video processing applications.

call as Hierarchical Additive Hough Transform (HAHT), that combines the block-level Hough spaces to generate global Hough spaces in a computationally efficient manner.

4.7.1 Extending the additive property to multiple levels

According to the additive property of HT stated in Section 4.6.1, the Hough Transform (HT) of point A with respect to (w.r.t) the image origin ‘O’ is equal to the sum of the

HT of A w.r.t any intermediate point B and HT of B w.r.t O , i.e.

$$HT(A, O) = HT(A, B) + HT(B, O) \quad (4.23)$$

where $HT(X, Y)$ represents the HT of point X w.r.t point Y . In other words, the HT of a point w.r.t to a global origin can be broken into two parts: the HT of that point w.r.t a local origin and the HT of the local origin w.r.t the global origin.

FIGURE 4.25: Extending the additive property of HT across multiple levels

We extend this property further by breaking the HT into multiple parts instead of two, as stated in the corollary below.

Corollary of (4.23): Given a point A and a set of n points $\{B_1, B_2, \dots, B_n\}$ defined w.r.t origin O , HT of A w.r.t O is given by:

$$HT(A, O) = HT(A, B_n) + \sum_{i=1}^{n-1} HT(B_i, B_{i-1}) + HT(B_1, O) \quad (4.24)$$

Proof: Consider a line l passing through point A as shown in Fig. 4.25. Let $l_1, l_2, \dots, l_n$ be lines passing through a set of points $\{B_1, B_2, \dots, B_n\}$ respectively, such that l is parallel to $l_1, l_2, \dots, l_n$, i.e. $l_1 \parallel l_2 \parallel \dots \parallel l_n \parallel l$. Let $\rho_O(A, \theta)$ be length of the perpendicular from O to the line l passing through point A subtending angle θ with the x-axis. Similarly, we can define $\rho_\theta(B_1, O)$, $\rho_\theta(B_2, B_1)$, $\rho_\theta(B_i, B_{i-1})$ and $\rho_\theta(A, B_n)$ as indicated in Fig 4.25.

From Fig. 4.25, it can be seen that:

$$\rho_\theta(A, O) = \rho_\theta(A, B_n) + \sum_{i=n}^2 \rho_\theta(B_i, B_{i-1}) + \rho_\theta(B_1, O) \quad (4.25)$$

Each term in (4.25) can be mapped directly, on a one-to-one basis, to the corresponding HT terms in (4.24) resulting in the HT of A w.r.t O for angle θ .

This extended additive Hough property for multiple levels is now applied to compute HT of edge pixels using the proposed Hierarchical Additive Hough transform (HAHT).

FIGURE 4.26: Division of edge map into blocks and macroblocks and their origins

4.7.2 Deriving the Hierarchical AHT

Consider an edge map divided into k^2 blocks using a $k \times k$ grid. The steps of the HAHT can be explained in the following manner, using the example of 4×4 grid as shown in Fig. 4.26. The lowest level of hierarchy is indicated by $L(1)$ wherein Hough space is computed for each of the k^2 edge map blocks. The ρ values are computed for edge pixels in every block w.r.t to the origin of each block. The Hough space thus generated is called local Hough space and the origin is called local origin. In Fig. 4.26(a), the local origins of the blocks at level $L(1)$ are given by $O_{1,1}(L1), O_{1,2}(L1), \dots, O_{4,4}(L1)$. These Hough spaces are given by $H(L1)$ as shown below:

$$H(L1) = \left[\begin{array}{cc|cc} H_{4,1}(L1) & H_{4,2}(L1) & H_{4,3}(L1) & H_{4,4}(L1) \\ H_{3,1}(L1) & H_{3,2}(L1) & H_{3,3}(L1) & H_{3,4}(L1) \\ \hline H_{2,1}(L1) & H_{2,2}(L1) & H_{2,3}(L1) & H_{2,4}(L1) \\ H_{1,1}(L1) & H_{1,2}(L1) & H_{1,3}(L1) & H_{1,4}(L1) \end{array} \right] \quad (4.26)$$

At each subsequent hierarchy level $L(i)$, where $i > 1$, the local Hough spaces of previous level $L(i - 1)$ are grouped into groups of four. In each group, the constituent Hough spaces are used to generate a combined Hough space, wherein the ρ values are computed w.r.t the local origin of the group of four image blocks. For example, in Fig. 4.26(b), four image blocks are now combined into one macroblock and the local origins for $L(2)$ are

given by $O_{1,1}(L2), \dots, O_{2,2}(L2)$. This process of combining blocks is similarly extended to the next hierarchy level $L(3)$. The Hough spaces in $L(2)$ then become the local Hough spaces which will be combined in $L(3)$ to generate a combined Hough space w.r.t a local origins in $L(3)$.

In order to compute the Hough space of a macroblock at $L(2)$, we apply the extended additive property of Hough transform. The HTs of four local origins of the constituent blocks are computed w.r.t the origin of the macroblock. The same values can be used for all the macroblocks because the four local origins are placed at the same relative positions w.r.t to the origins of all the macroblocks. Referring to one macroblock, say top right corner, the local Hough spaces, $H_{4,3}(L1)$, $H_{4,4}(L1)$, $H_{3,3}(L1)$, and $H_{3,4}(L1)$, of the constituent blocks from $L(1)$ are added with the HTs of their respective local origins w.r.t to the macroblock origin as shown below:

$$\begin{aligned} H_{4,3}(L2) &= H_{4,3}(L1) + HT(O_{4,3}(L1), O_{2,2}(L2)) \\ &\vdots \\ H_{3,4}(L2) &= H_{3,4}(L1) + HT(O_{3,4}(L1), O_{2,2}(L2)) \end{aligned} \quad (4.27)$$

$H_{4,3}(L2)$, $H_{4,4}(L2)$, $H_{3,3}(L2)$ and $H_{3,4}(L2)$ are the Hough spaces of the four blocks w.r.t the macroblock origin at level $L(2)$. The Hough space of the macroblock, i.e. $H_{2,2}(L2)$, w.r.t its origin $O_{2,2}(L2)$ is the sum of these Hough spaces:

$$H_{2,2}(L2) = H_{4,3}(L2) + H_{4,4}(L2) + H_{3,3}(L2) + H_{3,4}(L2). \quad (4.28)$$

Similarly, the Hough spaces of the rest of the three macroblocks are computed to give four Hough spaces in $L(2)$:

$$H(L2) = \left[\begin{array}{c|c} H_{2,1}(L1) & H_{2,2}(L1) \\ \hline H_{1,1}(L1) & H_{1,2}(L1) \end{array} \right] \quad (4.29)$$

Following the extended additive property of Hough transform, the four Hough spaces generated in $L(2)$ can be further combined to generate $L(3)$ Hough space. This Hough space would correspond to the macroblock generated by combining the four blocks from $L(2)$. The Hough transforms of the local origins in $L(2)$ are computed w.r.t the origin of the macroblock in $L(3)$. Applying the extended additive Hough property, the HTs of these origins are added to the Hough spaces from $L(2)$ hierarchy level and then combined to give $L(3)$ Hough space.

This hierarchical addition of Hough spaces using the extended additive property of HT can thus be used to generate the Hough spaces at different levels of hierarchy.

The proposed HAHT method of adding Hough spaces at different hierarchy levels can be further optimized to reduce the number of computations. This is done by thresholding the Hough spaces at every hierarchy level. If a threshold for Hough space at every hierarchy level is considered, the (ρ, θ) candidates that cross a threshold at a given hierarchy level $L(i)$ represent the line segments that cross the minimum length criteria in the macroblocks at $L(i)$. In the next hierarchy level $L(i+1)$, only the selected candidates from $L(i)$ are added to the HTs of their respective block origins. This computation process is continued across hierarchy levels till the accumulation counts cross an overall minimum length condition. This process also leads to significant savings because the redundancy of processing noisy edge pixels (that contribute to false lines) is controlled.

FIGURE 4.27: Detection of long lines using HAHT

4.7.3 HAHT for Detection of Straight Lanes

We now demonstrate the use of the proposed HAHT on detection of straight lanes. Lanes are usually characterized by straight lines. Depending on the block size and length of these lines, a line can be distributed across different blocks. The HAHT approach aids in accumulating the vote counts of the straight lines across different blocks. To illustrate this, we refer to Fig. 4.27. As shown in Fig. 4.27, consider a macroblock having four blocks, say $B1$, $B2$, $B3$ and $B4$. Let line l be divided as l_1 and l_2 in $B3$ and $B4$ respectively. At level $L(1)$, the local Hough spaces of $B3$ and $B4$ are given by $H_{B3}(L1)$ and $H_{B4}(L1)$ with peaks at two different ρ values, $\rho_\theta(l_1, O_{B3})$ and $\rho_\theta(l_2, O_{B4})$ corresponding to the edge pixels lying on l_1 and l_2 . However, what we need is to locate

the lanes w.r.t to global origin, i.e. the origin of the entire edge map, O . At level $L(2)$, the blocks are grouped into one macroblock. The local Hough spaces of $B3$ and $B4$ are combined to compute the accumulation count of the entire line l that is made up of l_1 and l_2 . The proposed HAHT method can now be used to generate the combined Hough space of line l w.r.t the global origin O . This is done by adding the HT of the origins of $B3$ and $B4$ to the local Hough spaces of $B3$ and $B4$ that were generated in $L(1)$ and then adding the resulting Hough spaces into one Hough space for the macroblock. This follows from the extended Hough property i.e.

$$\begin{aligned}\rho_\theta(l_1, O) &= \rho_\theta(l_1, O_{B3}) + \rho_\theta(O_{B3}, O) \\ \rho_\theta(l_2, O) &= \rho_\theta(l_1, O_{B4}) + \rho_\theta(O_{B4}, O)\end{aligned}\tag{4.30}$$

and the overall accumulation count for the line l is given by:

$$n(\rho_l, \theta_l) = n(\rho_{l_1}, \theta_{l_1}) + n(\rho_{l_2}, \theta_{l_2})\tag{4.31}$$

where $n(\rho_l, \theta_l)$ is the accumulation count for the line l . Depending on the block size, the additive property of HT can be applied repetitively to different hierarchy levels to generate the Hough space at the desired levels.

4.7.4 Performance & Computation Cost Evaluation of HAHT

Computation complexity analysis: In all the hierarchical approaches considered [207–209], since a filtering process to allow only prominent peaks through thresholding is done at each level, peak detection also is performed along with voting. This is unlike the conventional HT computation approaches where peak detection is performed after the entire voting is completed. The proposed HAHT leads to a drastic reduction in computational complexity of the voting step as compared to the existing hierarchical approaches. This is achieved because of two main reasons. Firstly, existing hierarchical HT approaches [207–209] perform a recomputation of HT at every level. That is, for all the qualifying points upon thresholding at a lower level, the HT is recomputed w.r.t the new origin of the hierarchy level. This operation is performed at every level of the hierarchy. The cost incurred for performing a HT in terms of computations is 2 multiplications and 1 addition, assuming the trigonometric values are stored in an LUT. However, in HAHT, because of the additive property, an incremental update on the vote is achieved at every level through just one addition (of the HT of block origins to selected candidates from previous hierarchy level), rather than a recomputation of the HT value at the current level.

FIGURE 4.28: Edge maps (a) Lane1, (b) Lane2, (c) Lane3 and (g) Lane4 and corresponding detected lanes in (d), (e), (f) and (h) using the proposed HAHT

TABLE 4.8: Comparison of HT Computations

Image	Edge content (no. of pixels)	Proposed HAHT	Hierarchical HT in [209]	
			Adds	Multiplies
Lane1	8498	948	1530588	3061176
Lane2	4902	371	882731	1765462
Lane3	6137	488	1105148	2210296
Lane4	4063	11588	742928	1485856

Secondly, at the lowest level $L(1)$, even the additions need not be performed as the precomputed local HT's and the HT's of the origins of the blocks need to be accessed only from LUTs as shown in [200, 202]. After accumulation, only those that exceed the threshold are subjected to addition in the next level. Fig. 4.28 shows four edge maps of lanes - Lane1, Lane2, Lane3 and Lane4, and the corresponding lanes extracted by the HAHT. Table 4.8 shows the number of computations involved to compute the Hough spaces till level-2 for the different lane images shown in Fig. 4.28. It can be seen that the number of addition operations are drastically reduced in the proposed method as compared to [209]. Also, the proposed HAHT does not involve any multiplications as it uses the LUT based method in [200, 202]. This shows the number computations is lower by as high as 98-99%. Similar savings were found in other test cases also. As far as the

peak detection step is concerned, the complexity is governed by the comparisons that are done in the thresholding process. If we assume that the same thresholding scheme is employed for both the HAHT and the hierarchical method described in [209], then the peak detection processes would be similar in terms of complexity.

Timing analysis: The main motivation for most hierarchical HT approaches has been to allow for parallelism, thereby accelerating the HT process. Hence, in terms of timing, these methods are usually far superior compared to the serial approaches in [83, 210]. Further, due to the reduced computational complexity of the voting step in HAHT compared to other existing hierarchical HT methods, the HAHT leads to savings in time costs as well. This is because the timing cost at every hierarchy level involves only *one* addition per (ρ, θ) tuple in the proposed HAHT whereas a recomputation of HT in [207–209] involves a multiplication and addition. If a 16-bit multiplication and addition are employed, each HT computation for every (ρ, θ) tuple in [207–209] would require $5T_{16-bitAdd}$, assuming $T_{16-bitMult} = 4T_{16-bitAdd}$ [95]. In the proposed HAHT, each HT computation would require only $T_{16-bitAdd}$ time units.

4.8 Computation Cost Analysis for Lane Detection using Proposed AHT and HAHT

In this section, we will revisit the evaluation of the computational complexity of the proposed lane detection method. In Section 3.4.3 of Chapter 3, we have shown that proposed GMH-GAH-HT engine results in 70% lesser computation cost in an image block as compared to conventional HT based lane detection methods in [64, 65, 69]. The conventional parameterized Hough transform equation (2.1) was considered in Section 3.4.3 to evaluate the computational complexity of the proposed method against [2, 52, 64, 65, 69]. In this section, we will show a further reduction in computational complexity of the proposed lane detection method because of the use of the AHT and HAHT engines. We will first show the cost reduction in a single block, and then show the computational cost for entire lane detection process in near and far-views of the host vehicle.

Equations (3.25) to (3.29) in Section 3.4.3 show the complexity of HT computation for an $M_b \times N_b$ sized image block using the conventional HT method. As shown in Section 4.6, the AHT reduces the computation cost by replacing all the multiplications and trigonometric operations with addition operations. For lane detection, as discussed in HAHT in Section 4.7, we compute the local Hough spaces only. Therefore, in order to determine lanes in each block, the Hough space accumulations are the only computations involved in generating the local Hough spaces. Combining this cost with the GMH and

GAH computation as discussed in Section 3.4.3, we have the following expressions for locating lane features in each block using AHT:

$$\# \text{ of additions for GMH-GAH generation} = 8M_bN_b \quad (4.32)$$

$$\# \text{ of accumulations for local Hough space computation} = p_{prop}N_{\theta_{prop}} \quad (4.33)$$

Using the above expressions, we compare the computation cost of the lane feature extraction step in the proposed method using AHT for each block against conventional HT based lane detection methods in [64, 65, 69] and steerable filter based method in [2]. Table 4.9 lists the number of computations involved in each method. It can be seen from the fourth and fifth columns in Table 4.9 that introducing AHT into the proposed lane detection method further reduces the computation cost by nearly 9.5 times (in terms of equivalent addition operations) and eliminates all multiplications and trigonometric operations.

TABLE 4.9: Comparison of computation complexity of proposed lane detection method using AHT

Operations	Conventional [64, 65, 69]	Steerable Filters [2]	Proposed using conventional HT	Proposed using AHT
Additions	115200	864000	106230	92160
Multiplications	92160	910080	28140	0
Trigonometric operations	92160	39600	28140	0
Accumulations	46080	14070	14070	14070
Total computations (equivalent additions)	3110400	16072950	1020780	106230

Given the computational cost savings in each block as shown in Table 4.9, the combining of the Hough spaces of all the blocks to determine the lanes in far-view is also simplified using the proposed HAHT engine. Let us consider two rows of blocks, each having eight 90×128 blocks. The computation cost of methods in [64, 65, 69] and [2] to determine the lane features in the region of interest specified by the two rows of blocks will be 20 times the number of computations listed in Table 4.9 under their respective columns. This is because each row of the image is assumed to have 10 blocks each of size 90×128 in the current image calibration setup. However, in the proposed block-based lane detection method, we determine the lanes in the near-view first. The lane positions in the near-view are used to predict the possible blocks in far-view. Therefore, a maximum of 6 blocks (worst case scenario) will be processed in the second row of blocks, instead of 10 blocks. In addition to the block based computations, the HAHT would also involve additional $\sqrt{M_b^2 + N_b^2} \times N_{\theta_{prop}}$ additions per hierarchy level to add local Hough spaces between two hierarchy levels. Considering that we have 3 hierarchy levels, we compute the overall computation cost of determining the lanes in near and far views for the proposed method using AHT and HAHT, and compare it with methods in [64, 65, 69]

and [2] in Table 4.10. It can be seen that the total number of computations of the proposed method is less than 3% of the total number of computations of techniques in [2, 64, 65, 69].

TABLE 4.10: Comparison of computation complexity of proposed lane detection method using AHT and HAHT for detecting lanes in both near and far views

Operations	Conventional [64, 65, 69]	Steerable Filters [2]	Proposed using AHT & HAHT
Additions	2304000	17280000	1493280
Multiplications	1843200	18201600	0
Trigonometric operations	1843200	792000	0
Accumulations	921600	281400	225120
Total computations (equivalent additions)	62208000	321459000	1718400

With such significant savings in computation cost, the proposed method provides different design options that can be used to develop architectures of varying throughput, including real time operation. This would depend on the embedded platform and the architectures that are chosen for the implementation.

4.9 Summary

In this chapter, techniques were proposed to detect host and neighbor lanes in far-view using the GMH-GAH-HT engine and block-based HT computation. The proposed techniques were shown to be effective in detecting multiple lanes in far-view, that are both straight and curved. The effectiveness of the techniques was shown on various kinds of real-world road scenarios including rainy and night images. The computational costs involved in the proposed techniques were then addressed by proposing efficient computation architectures that compute the HT in a parallel fashion using the block-based AHT method. In order to combine the block-level Hough spaces from AHT, the HAHT was proposed. It was shown that the proposed architectures exploit the inherent parallelism and result in orders of magnitude speed up, without increasing the computational complexity. Having determined the lanes as regions of interest, we will now propose techniques that robustly detect the objects of interest, i.e. vehicles, in the next chapter.

CHAPTER 5

Estimating Proximity of Target Vehicles from Host Vehicle

5.1 Introduction

Vehicles are the primary targets that need to be continuously monitored if a driver intends to change lane. Before changing lanes, the driver constantly monitors the proximity of the vehicles that surround him. The driver will make the lane change maneuver if and only if the target vehicles are found at a safe distance and moving at acceptable speeds in the lanes. Therefore vehicle proximity estimation is a vital process that the driver performs while driving, and more importantly before changing lanes.

An LCDAS is expected to assist the driver in deciding to make the lane change maneuver based on the proximity of the vehicles in the road scene. In order to do this, we first need to detect the presence of vehicles in the road scene and if a vehicle is detected, its relative distance from the host vehicle is then estimated. In this chapter, novel vision-based techniques are proposed to detect and estimate the proximity of vehicles in road scene. Multiple visual cues are used in the proposed method to first detect the presence of a vehicle. Then, the proximity of the vehicle is estimated using a model based on the lane positions and the location of the vehicle signature in the image.

The proposed method is based on the following observations and assumptions that are made about vehicles, in the context of lane change assistance:

1. Lanes are the primary regions of interest where vehicle signatures need to be searched. In this work, it is assumed that lane change assistance will be provided

to the driver based on the positions of the vehicles that are approaching the host vehicle from rear and front in their respective lanes.

2. The *non-vehicle* objects that are found on the lanes of highway and major roads are shadows of trees, vehicles and road fixtures, and road markings. It is assumed that pedestrians and animals are not found in the regions of interest.
3. If the driver intends to change lane, he is interested in monitoring the *closest vehicle* to the ego-vehicle in each lane. In other words, if there are two vehicles in the same lane, the closer vehicle must be detected because the maximum risk of collision is posed by the closer vehicle.

These observations are used in the proposed technique to detect vehicles closest to the ego-vehicle in the road scene. For the rest of the chapter, the term “vehicle” refers to the vehicle closest to the host-vehicle in each available lane as seen by a camera either in front of the vehicle or to the rear of the vehicle. These vehicles are also referred to as “target vehicles”.

FIGURE 5.1: Envisaged steps of the proposed vehicle proximity estimation module for lane change assistance

Fig. 5.1 shows the envisaged steps in detecting and estimating the proximity of target vehicles in the road scenes. Given an input image frame I it is envisaged to first detect the vehicles in the lanes and then determine their proximity to the host-vehicle. In order to do this, the first step is to determine these lanes, which we call as regions of interest or RoIs. The host lane and neighbor lanes form the RoIs for the host vehicle to detect the

target vehicles. These RoIs are created using the lane positions, that were determined using the techniques proposed in Chapters 3 and 4. In the subsequent sections, we will show how these RoIs play a vital role in detecting the target vehicles and their proximity.

5.2 Deriving RoIs for Vehicle Detection

Host and neighbor lanes form the RoIs for detecting vehicles. Therefore, the lane markings that were detected using the techniques proposed in Chapters 3 and 4 are used to define the RoIs for vehicle detection. If the ego-vehicle lane or host lane is indicated by L_H , the neighboring lanes to the left are represented by L_{H-1} . Similarly, the neighbor lane to the right of L_H is represented by L_{H+1} .

Given the positions of the lane markings, we first create masks for each of the lanes and extract the lane regions. This is illustrated in Fig. 5.2. If l^l and l^r represent two straight lines that form the left and right lanes markings respectively, the area enclosed by l^l and l^r is the lane or region of interest. This area E_M is given by the intersection of area E^l (represented by slanted lines, which is right of l^l) and the area E^r (represented by shadowed region, which is to the left of l^r). These areas are given by:

$$\begin{aligned} E^l &:= (l^l > 0) & E^r &:= (l^r < 0) \\ E_M &= E^l \cap E^r \end{aligned} \quad (5.1)$$

E_M is the mask that selects the lane bound by l^l and l^r . Now let us see how to determine E^l and E^r , given the parametric representations of l^l and l^r in terms of (ρ, θ) as shown below:

$$\begin{aligned} l^l : \rho^l &= x \cos \theta^l + y \sin \theta^l \\ l^r : \rho^r &= x \cos \theta^r + y \sin \theta^r \end{aligned} \quad (5.2)$$

Given the above equations, we find all (x, y) that satisfy l^l and l^r . In other words, while traversing in the X -direction, if a point (x_1, y_1) satisfies l^l , then any point (x_2, y_1) , such that $x_2 > x_1$, will give ρ value greater than ρ^l . Therefore, if we find x_1 for a given y_1 , which gives $\rho > \rho^l$, then the coordinates to the right of x_1 will give the region E^l in Fig. 5.2. Similarly, E^r can be found using the equation of l^r . The intersection of E^l and E^r gives the required mask E_M .

From Chapters 3 and 4, l^l and l^r need not be defined by just one line each. Instead, l^l is represented by one set of line segments and l^r by another set of line segments, depending on the block size and whether the lane is straight or curved. Fig. 5.3 shows the case of a curved lane that is approximated using line segments as discussed in Chapter 4. The left lane l^l is given by the line segments l^{l1}, l^{l2}, l^{l3} and l^{l4} . Similarly l^{r1}, l^{r2}, l^{r3} and

FIGURE 5.2: Illustrating mask creation

FIGURE 5.3: Illustrating mask creation for lanes represented by line segments. The axes indicates the indices for the blocks.

l^{r4} form the right lane l^r . It was shown in Section 4.3 that each line segment is also associated with a block address. For example, l^{l1} in Fig. 5.3 is associated with block $B(8,2)$. The shadowed region in Fig. 5.3 gives the mask E_M for this lane. In order to get this mask, we employ a row-wise approach. For each row, we find the line segments defining the left lane and the right lane. In Fig. 5.3, for the 8th row, the left lane is given by l^{l1} and the right lane is given by l^{r1} ; in the 7th row, l^{l2} and l^{l3} both define l^l , and l^{r2} defines l^r .

Next in every row, we start with the line segment with the lowest block index and apply the conditions (5.1) defined for E^l and E^r in that block. Unlike the mask that was generated previously considering the entire image in Fig. 5.2, these conditions will be applied for the blocks in the rows where these line segments occur. If we consider 7th row, for the left lane, l^{l2} has a lower block index $B(7,2)$. We go to $B(7,2)$ and apply

FIGURE 5.4: Illustrating mask creation for multiple line segments

the condition for E^l to the pixel coordinates in $B(7, 2)$ resulting in the region, say E^{l2} . This is shown in Fig. 5.4. Next we go to the next line segment (if any) for the left lane. In the case of 7th row, l^l has a second segment l^{l3} . Applying the above process, we get the region E^{l3} (Fig. 5.4). Now the same process is applied to line segments forming the right lane l^r the blocks of 7th row. In this example, l^{r2} is the only line segment for l^r in 7th row resulting in region E^{r2} . These regions are used in the following way to give the lane RoI $E_M(B(7, :))$ for 7th row of blocks:

$$E_M(B(7, :)) = \{E^{l2} \cup E^{l3}\} \cap \{E^{r2}\} \quad (5.3)$$

This can be generalized in the following way. Let $E^{l1}, E^{l2}, \dots, E^{l\alpha_l}$ be the regions for α_l line segments forming the left lane in k^{th} row of blocks. Similarly, let $E^{r1}, E^{r2}, \dots, E^{r\alpha_r}$ represent the regions for right lane. Then, the region of interest $E_M(B(k, :))$ enclosed by these line segments in the k -th row of blocks is given by:

$$E_M(B(k, :)) = \left\{ \bigcup_{i=1}^{\alpha_l} E^{li} \right\} \cap \left\{ \bigcup_{j=1}^{\alpha_r} E^{rj} \right\} \quad (5.4)$$

The mask for the entire RoI is given by combining the masks from every row till the vanishing point. Therefore, in Fig. 5.3, E_M is given by:

$$E_M = E_M(B(8, :)) \cup E_M(B(7, :)) \cup E_M(B(6, :)) \quad (5.5)$$

The neighbor lane RoIs can also be extracted in a similar fashion. Given these RoIs, we will now detect vehicles in these RoIs and estimate their relative proximities to the host vehicle.

5.3 Vehicle Detection using Multiple Cues

In this section we describe an adaptive method to detect vehicles in the lanes. The proposed method uses multiple cues to ascertain the presence of a vehicle in the lanes. These visual cues include under-vehicle shadow and vehicle edges. The proposed technique depends on a combination of these cues in a specific manner to robustly detect vehicles in lanes. The techniques proposed in this section are devised to first address the detection of *cars or four wheeled vehicles*. We will then show how these techniques can be further extended for *motorcycles* also in Section 5.4.

5.3.1 Detecting Under-vehicle Shadow

Under-vehicle shadow is one of the distinctive signatures of a vehicle, especially during day [78]. As discussed in Section 2.4 (Chapter 2), under-vehicle shadow can be used quite effectively to hypothesize the presence of a vehicle [78]. However, there are still some open challenges, especially in terms of robustness to varying illumination in road scenes. We first describe these challenges and present novel techniques, that are adaptive to such variations in road scenes.

The under-vehicle shadow is relatively the darkest image patch when compared to other shadows on road surface. Fig. 5.5 shows a few examples of the under-vehicle shadows cast by the vehicle on the road surface under different lighting conditions. It can be seen that the under-vehicle shadow region is comparatively darker than the surrounding regions of the vehicle. If an image is binarized with an *appropriate* threshold, the shadow can be extracted. However, the variations in a road scene make this a difficult process. Having the lane or region of interest identified beforehand reduces this uncertainty of thresholding to an extent because the threshold is now dependent on the intensities in the region of interest only. In this way, the first step of lane region extraction discussed in Section 5.2 helps in reducing the complexity of thresholding process.

However, there are still some variations in the lane and the binarization threshold must cater to these variations. For example, in the images shown in Fig. 5.5 (d), (e) & (f), the under-vehicle shadow is found in three different situations showing two different kinds of illumination variations. Firstly, there is global variation because of the overall weather and lighting of the image. For example, the illumination in Fig. 5.5 (d) is less brighter than Fig. 5.5 (e) & (f). In addition to such global variation, there are local illumination variations due to shadows of trees and other objects. This is seen in Fig. 5.5 (e) & (f) where the under-vehicle shadow is affected by the presence of the shadow of trees in Fig. 5.5 (e) but not in Fig. 5.5 (f). In such scenarios, under-setting or over-setting

FIGURE 5.5: Examples of under-vehicle shadows under different conditions (a) bright day light (b) presence of shadows of trees completely (c) change in overall illumination due to clouds (d)-(f) brighter illumination with shadows showing different variations of under-vehicle shadow intensities

the threshold can lead to false positives or false negatives. This particular aspect of automating the threshold setting, which considers local illumination variations, is not adequately addressed in existing works that use the under-vehicle shadow as a cue for vehicle detection. We will now present a study on the shadows cast by the vehicles, which will be used to propose a technique that extracts the under-vehicle shadow in the presence of varying local illumination.

Characteristics of Shadows in Road Scene: Jiang et al in their work on shadows [211], presented in CVPR'92, describe different kinds of shadows that are cast by objects with different light sources. According to [211], there are two kinds of shadows: umbra and penumbra. A shadow caused by an object is a complete shadow or umbra shadow if it conceals light totally. The second type of shadow is a partial shadow or penumbra shadow, which is caused by an object partially concealing the light. In the case of a road scene, the shadows cast by a vehicle, nearby buildings, railings etc. on the road surface are umbra shadows because they conceal the light from the light source completely. The shadows cast by the trees and foliage can be considered as penumbra shadows because they partially block the light source. Fig. 5.6 shows these two kinds of shadows cast on

the road surface. In Fig. 5.6 the umbra shadows are cast by the vehicle (shown by the solid box) and the penumbra shadows are cast by the trees (shown by the dashed box). The vehicle blocks the sun light completely and hence an umbra shadow is created. The foliage of trees are farther away from the road surface and have obvious gaps letting the sun light creep through. Consequently, a penumbra shadow is cast on the road surface.

FIGURE 5.6: Umbra and penumbra shadows seen on a road scene. The solid box shows umbra shadow cast by the vehicle and the dashed box shows a penumbra shadow by the foliage of trees.

According to [211], the umbra shadows are darker than the penumbra shadows. This is because the umbra shadows do not receive any light from the light source. This is observed in the case of road scenes also as shown in Fig. 5.6. The umbra shadows cast by vehicles or buildings are darker (of lesser intensities) than the penumbra shadows cast by foliage of trees. Also, based on the kind of obstruction, the penumbra shadows usually have more edge content than the umbra shadows. Another observation is that an umbra shadow is darker if the object is closer to the surface of reflection as compared to object farther away from the surface.

Based on these properties of shadows, the following observations are made about the under-vehicle shadows:

1. The under-vehicle shadow is an umbra shadow cast by the vehicle on the road surface.
2. The under-vehicle shadow is always found under the vehicle only, irrespective of the direction of the light source. This is unlike the vehicle shadow cast by the sides of the vehicle which can fall anywhere around the vehicle based on the position of the light source.
3. The under-vehicle shadow is darker than other shadows cast on the road surface in its vicinity because it is cast by the bottom of the vehicle on the road surface.

The bottom of the vehicle is the closest to the road surface that casts a darker shadow as compared to any other obstruction on the road surface.

In other words, the under-vehicle shadow is the *darker of the dark intensities* present in *its vicinity*. The reason why “vicinity” is emphasized is because being dark or bright is a relative term. The under-vehicle shadow is the darkest region when compared to its immediate surroundings. In this way, we can cater to both global and local illumination variations in the road scenes.

Determining Binarization Threshold: Based on these observations about under-vehicle shadows, we now find an *appropriate* threshold for binarization that will extract the under-vehicle shadow. The threshold is desired to be such that it locates the under-vehicle shadow accurately and does not give any false positives or false negatives. In order to do this, we conducted an experiment, the objective being: find if there is any relationship between a known intensity in the image and the unknown intensity of the shadow region. In other words, let us assume we have a known region A_1 with average intensity $\bar{I}_{A_1}$ that is around the vehicle and is not the under-vehicle shadow. Let A_2 be the under-vehicle region in the image with average intensity given by $\bar{I}_{A_2}$. We want to find a function $f(\cdot)$ such that

$$\bar{I}_{A_2} = f(\bar{I}_{A_1}) \quad (5.6)$$

The above function can then be used to set the binarization threshold to extract under-vehicle shadows in the RoIs. Based on the observations about shadows described above, it is expected that the under-vehicle shadow region will have lower average intensity than the region around the vehicle, i.e. $\bar{I}_{A_2} < \bar{I}_{A_1}$.

FIGURE 5.7: Regions A_1 and A_2 selected on the image $\mathbf{I}$ for the experiment.

In order to do this, we must select the known region A_1 that is nearest to the under-vehicle shadow. This is because such a region will best capture the local and global

variations around the under-vehicle shadow. After going through the test videos, we select A_1 as the region just outside the vehicle, where the under-vehicle shadow is found. It will be shown later in this chapter, how these regions are selected automatically. In this experiment, 1300 images with vehicles on major roads were used to generate the training data for the experiment. These images were taken under different weather conditions, lighting conditions and distances from the host vehicle. A study of these images showed that if the lanes are determined and if the vehicles are present in the lanes (i.e. they are not changing lanes), then the regions A_1 next to the vehicle tyres captures road surface in 99.8% of the cases. Fig. 5.7 shows the chosen A_1 and A_2 regions. In all the images $\mathbf{I}^{(i)}$ in the dataset, we manually select the regions $A_1^{(i)}$ and $A_2^{(i)}$. The average intensities of A_1 and A_2 represented by $\bar{I}_{A_1}^{(i)}$ and $\bar{I}_{A_2}^{(i)}$ are computed to obtain the following sets of data samples:

$$\bar{\mathbf{I}}_{A_1} = \begin{bmatrix} \bar{I}_{A_1}^{(1)} \\ \bar{I}_{A_1}^{(2)} \\ \vdots \\ \bar{I}_{A_1}^{(K)} \end{bmatrix} \quad \bar{\mathbf{I}}_{A_2} = \begin{bmatrix} \bar{I}_{A_2}^{(1)} \\ \bar{I}_{A_2}^{(2)} \\ \vdots \\ \bar{I}_{A_2}^{(K)} \end{bmatrix} \quad (5.7)$$

We first plotted the average intensities of the A_1 and A_2 training samples to see if

FIGURE 5.8: Scatter plot between $\bar{\mathbf{I}}_{A_1}$ and $\bar{\mathbf{I}}_{A_2}$

there is any relationship between these two intensities. Fig. 5.8 shows the scatter plot between $\bar{\mathbf{I}}_{A_1}$ and $\bar{\mathbf{I}}_{A_2}$ of the training samples. It can be seen that there is no perceivable relationship between these two quantities.

However, after a closer look at the training samples and input images, the following observation was made: if intensity of the image around the vehicle is bright, i.e. if $\bar{I}_{A_1}$ is high, the under-vehicle shadow is darker, i.e. $\bar{I}_{A_2}$ is low. In other words, on a bright sunny day, the difference between the intensity of the shadow region and the non-shadow region is greater than what is observed on a cloudy day. Given I_{A_1} and I_{A_2} as the average intensities of non-shadow region and shadow regions respectively, the following are observed:

- I_{A_1} is high $\Rightarrow$ bright day $\Rightarrow$ shadow is darker $\Rightarrow I_{A_1} - I_{A_2}$ is higher.
- I_{A_1} is lower $\Rightarrow$ cloudy or less bright day $\Rightarrow$ shadow is 'less' darker $\Rightarrow I_{A_1} - I_{A_2}$ is lower.

In order to see if there is any relationship between I_{A_1} and $I_{A_1} - I_{A_2}$, $\bar{I}_{A_1}$ and $\bar{I}_{A_2}$ are used to compute difference in average intensity $\mathbf{I}_\Delta$ as shown below:

$$\mathbf{I}_\Delta = \bar{I}_{A_1} - \bar{I}_{A_2} = \begin{bmatrix} \bar{I}_{A_1}^{(1)} - \bar{I}_{A_2}^{(1)} \\ \bar{I}_{A_1}^{(2)} - \bar{I}_{A_2}^{(2)} \\ \vdots \\ \bar{I}_{A_1}^{(K)} - \bar{I}_{A_2}^{(K)} \end{bmatrix} = \begin{bmatrix} \bar{I}_\Delta^{(1)} \\ \bar{I}_\Delta^{(2)} \\ \vdots \\ \bar{I}_\Delta^{(K)} \end{bmatrix} \quad (5.8)$$

The differences in average intensities, $\mathbf{I}_\Delta$, are plotted against the average intensities of the regions A_1 , i.e. $\bar{I}_{A_1}$. Fig. 5.9 shows this plot.

FIGURE 5.9: Scatter plot between $\bar{I}_{A_1}$ and $\mathbf{I}_\Delta$

It can be seen from Fig. 5.9 that the training data samples show a linear relationship. If a linear hypothesis model $h_\theta(x) = \theta_1 x$ is applied to these data samples and using least squares regression model [212] with input training samples $X = \bar{\mathbf{I}}_{A_1}$ and output responses $Y = \mathbf{I}_\Delta$, we can solve for $h_\theta(x)$ using the following equation:

$$\theta = (X^T X)^{-1} X^T Y. \quad (5.9)$$

Applying the above equation to the data samples, we get $h_\theta(x) = 0.597x$. This linearization of the training data samples is shown in Fig. 5.10. The mean absolute error μ and the standard deviation σ of the data samples when compared with the linear model were found to be equal to 13.05 and 16.05 respectively.

FIGURE 5.10: $\mathbf{I}_\Delta$ versus $\bar{\mathbf{I}}_{A_1}$

This model is now used to set the binarization threshold to extract the under-vehicle shadow regions. For a given intensity $x = \bar{I}_{A_1}$, the binarization threshold T_b is given by:

$$T_b = x - h_\theta(x) + \epsilon \quad (5.10)$$

where ϵ is a tolerance value to accommodate the deviation of $h_\theta(x)$, i.e. $\sigma = 16.05$. Changing the value of ϵ either makes the binarization threshold more rigid or flexible. However, in order to find the binarization threshold T_b using equation (5.10), the region with known average intensity around the vehicle, i.e. $x = \bar{I}_{A_1}$, needs to be found first. In other words, for a given input image with the RoIs marked, we need to do the following:

1. Find the known non-under-vehicle shadow region A_1 to find the average intensity $\bar{I}_{A_1}$.

2. Find the threshold T_b using (5.10).
3. Binarize the *vicinity* of A_1 , using T_b , where we expect to see the under-vehicle shadow, i.e. region A_2 .

We now propose an algorithm that performs the above three steps. The proposed algorithm systematically first finds the region A_1 and then determines the under-vehicle shadow region using equations (5.10).

Algorithm to Detect Under-vehicle Shadows: We first define some matrices which are used in describing the algorithm. Given image $\mathbf{I}$ and the RoI mask $\mathbf{E}_M$, two matrices are defined to make the explanation easy to follow. First is what we call an RoI block or $\mathbf{R}$, which is the rectangular region in $\mathbf{E}_M$ which binds the RoI in $\mathbf{E}_M$. Each pixel in $\mathbf{R}$ is either a 0 or a 1. Fig. 5.11 shows these blocks using a sample image. For the input image with mask in Fig. 5.11(a), $\mathbf{R}$ is shown in Fig. 5.11(b). When $\mathbf{R}$ is applied on $\mathbf{I}$, we get the image block or $\mathbf{I}_R$, as shown in Fig. 5.11(c).

FIGURE 5.11: The image block (c) and RoI block (b) are shown for an input image with mask (a). The division of the image block into bands is shown in (d) and (e). The bands are further divided into sub-blocks in (f) and (g).

Given $q \times r$ sized $\mathbf{R}$ and $\mathbf{I}_R$, the RoI block is first divided into bands of a pre-defined width w_q . Therefore, we have

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}_1 \\ \mathbf{R}_2 \\ \vdots \\ \mathbf{R}_\alpha \end{bmatrix} \quad \mathbf{I}_R = \begin{bmatrix} \mathbf{I}_{R1} \\ \mathbf{I}_{R2} \\ \vdots \\ \mathbf{I}_{R\alpha} \end{bmatrix} \quad (5.11)$$

where every $\mathbf{R}_i$ and $\mathbf{I}_{Ri}$ is a $\frac{q}{w_q} \times r$ matrix. These bands are shown in Fig. 5.11(d) for an input image.

We start processing with the lowest band $\mathbf{R}_\alpha$ and move upwards. This direction is chosen because in a lane change assistance system, we are interested in the nearest vehicle to the ego-vehicle. A band $\mathbf{R}_i$, starting from the α^{th} band, is further divided into λ blocks, each of size $\frac{q}{w_q} \times \frac{r}{w_r}$ given by:

$$\mathbf{R}_i = \begin{bmatrix} \mathbf{R}_{i,1} & \mathbf{R}_{i,2} & \cdots & \mathbf{R}_{i,\lambda} \end{bmatrix} \quad \mathbf{I}_{Ri} = \begin{bmatrix} \mathbf{I}_{Ri,1} & \mathbf{I}_{Ri,2} & \cdots & \mathbf{I}_{Ri,\lambda} \end{bmatrix} \quad (5.12)$$

Depending on the mask, a block $\mathbf{R}_{i,j}$ in the i^{th} band will either have all pixels set to zero, or some pixels set to zeros or all pixels set to 1. We compute the vector $\mathbf{N}_{\mathbf{R}_{i,j}}$ given by:

$$\mathbf{N}_{\mathbf{R}_i} = \begin{bmatrix} N_{i,1} & N_{i,2} & \cdots & N_{i,\lambda} \end{bmatrix} \quad (5.13)$$

where $N_{i,j}$ is given by the summation of all pixels in the block $\mathbf{R}_{i,j}$, i.e.

$$N_{i,j} = \sum_{u=1}^{\frac{q}{w_q}} \sum_{v=1}^{\frac{r}{w_r}} \mathbf{R}_{i,j}(u, v). \quad (5.14)$$

$N_{i,j}$ gives the total number of valid image pixels that are there inside block $\mathbf{R}_{i,j}$. Using the values in $\mathbf{N}_{\mathbf{R}_i}$ for a given band, $\mathbf{R}_i$, we now select the region A_1 , which is used to find the binarization threshold in equation (5.10).

Observation: Given that the RoI defines the lane in which the vehicle is being searched, it was observed that if a band has the under-vehicle shadow, the corner blocks in the band have sufficient road intensity information. These blocks can be considered as regions A_1 to determine the binarization threshold in equation (5.10). This is illustrated in Fig. 5.12. For the image block shown in Fig. 5.12, the under-vehicle shadow occurs in the 6^{th} band as indicated by the red band. This particular band $\mathbf{I}_{R6}$ is further zoomed in and divided into blocks. It can be seen that blocks $\mathbf{I}_{R6,8}$ and $\mathbf{I}_{R6,20}$, that lie at the corners of the band will not be under-vehicle shadow regions. They can therefore be considered as regions A_1 .

FIGURE 5.12: The image block $\mathbf{I}_{\mathbf{R}_6}$ showing blocks $\mathbf{I}_{\mathbf{R}_{6,8}}$ and $\mathbf{I}_{\mathbf{R}_{6,20}}$ with road intensities.

Given this observation, we run through $\mathbf{N}_{\mathbf{R}_i}$ the i^{th} band that is under consideration. The leftmost and rightmost values in $\mathbf{N}_{\mathbf{R}_i}$ which are above a threshold give the regions, that can be considered as A_1 for binarization threshold computation. For the example considered in Fig. 5.12, $N_{6,7} = 12, N_{6,8} = 84, \dots, N_{6,20} = 73, N_{6,21} = 20$. If we set a threshold on the minimum number of mask pixels to be 25, then we get $N_{6,8}$ and $N_{6,20}$ giving the regions for A_1 .

We then find the average intensity of these image blocks, i.e. $\mathbf{I}_{\mathbf{R}_{6,8}}$ and $\mathbf{I}_{\mathbf{R}_{6,20}}$, which gives $x = \bar{I}_{A_1}$ in equation (5.10). This value is used to determine the binarization threshold T_b using equation (5.10). Then each image sub-block $\mathbf{I}_{\mathbf{R}_{i,j}}$ in $\mathbf{I}_{\mathbf{R}_i}$ is binarized using this threshold to give the binarized image sub-block $\mathbf{I}_{\mathbf{BR}_{i,j}}$, i.e.

$$\mathbf{I}_{\mathbf{BR}_{i,j}} := \{0 < \mathbf{I}_{\mathbf{R}_{i,j}} < T_b\} \quad (5.15)$$

After binarizing the sub-blocks in the image band $\mathbf{I}_{\mathbf{R}_i}$, we find the number of binarized image sub-blocks (represented by b) that indicate intensities lower than T_b . If this number b is found to be greater than a ration, with respect to the number of the sub-blocks in $\mathbf{I}_{\mathbf{R}_i}$, then that band is considered to have an under-vehicle shadow. This check on the number of sub-blocks is made to ensure that there is a minimum number of blocks that have the dark intensity region. This minimum number is varied depending on the position of the image band because the number of sub-blocks with a valid under-vehicle shadow in upper bands is lesser than those found in the lower bands. Referring to the image sub-blocks shown in Fig. 5.12, sub-blocks 11 to 18 will give non-zero pixels on binarization indicating the presence of a valid vehicle shadow. Fig. 5.13 shows the application of the proposed method on an image.

Algorithm 1 summarizes all the steps involved in the under-vehicle shadow detection in the form of an algorithm. We start with the lowest band, i.e. the α^{th} band and move upwards, and follow the steps described above till we either get a possible under-vehicle shadow band or the vanishing point. If an under-vehicle shadow band is found then its position is noted and further vehicle signature confirmation steps are performed using

FIGURE 5.13: Example showing under-vehicle shadow detection.

the steps described in the next sections. Otherwise, it is assumed that a vehicle is not present in the RoI.

Algorithm 1 Adaptive Under-vehicle Shadow Detection

- 1: **Given** mask block $\mathbf{R}$ and image block $\mathbf{I}_{\mathbf{R}}$
 - 2: $\mathbf{R}_i, \mathbf{I}_{\mathbf{R}_i} \leftarrow$ Divide $\mathbf{R}$ and $\mathbf{I}_{\mathbf{R}}$ into bands
 - 3: **for** Every band $i = \alpha$ **to** 1 **do**
 - 4: Divide $\mathbf{R}_i$ and $\mathbf{I}_{\mathbf{R}_i}$ into λ blocks
 - 5: For each sub-block, compute $\mathbf{N}_{\mathbf{R}_{i,j}}$
 - 6: $A_1 =$ Find outermost non-zero blocks in $\mathbf{R}_i$
 - 7: $\bar{I}_{A_1} =$ Compute average intensities of blocks in A_1
 - 8: $T_b = \bar{I}_{A_1} - h_{\theta}(\bar{I}_{A_1}) + \epsilon$
 - 9: For each sub-block, compute $\mathbf{I}_{\mathbf{BR}_{i,j}} := \{0 < \mathbf{I}_{\mathbf{R}_{i,j}} < T_b\}$
 - 10: **if** # of binarized sub-blocks with non-zero pixels $> b$ **then**
 - 11: Under-vehicle shadow found in band i
 - 12: $(x_{u1}, y_u), (x_{u2}, y_u) \leftarrow$ Find coordinates of shadow w.r.t image frame
 - 13: **break;**
 - 14: **end if**
 - 15: **end for**
-

5.3.2 Confirming Vehicle Signature using Horizontal & Slant Edge Symmetries

From the previous computation step, the presence of the under-vehicle shadow indicates that there is a possibility of presence of a vehicle above this shadow. In this section, we describe the next step in the proposed vehicle detection method. In this step, given the possible location of the under-vehicle shadow, we check if that shadow is indeed that of a vehicle. The GMH-GAH-HT engine is deployed in a selective manner to check the presence of vehicle signature above the under-vehicle shadow in the image.

Given the coordinates of the under-vehicle shadow, i.e. $(x_{u1}, y_u), (x_{u2}, y_u)$, we extract an image block $\mathbf{I}_v$ above the vehicle shadow. The width of the shadow, i.e. $x_{u2} - x_{u1}$,

is used to determine the size of the image block. For our experiments, a square image block was considered. Fig. 5.14 gives an example of these steps. For the input image shown in Fig. 5.14(a), the image block $\mathbf{I}_v$ is selected as shown in Fig. 5.14(b).

FIGURE 5.14: The image block $\mathbf{I}_{\mathbf{R}_6}$ showing blocks $\mathbf{I}_{\mathbf{R}_{6,8}}$ and $\mathbf{I}_{\mathbf{R}_{6,20}}$ with road intensities.

Next, an edge map for $\mathbf{I}_v$ is generated using gradient magnitude histogram (GMH). The gradient maps $\mathbf{G}_x$ and $\mathbf{G}_y$ are used to compute the gradient magnitudes map $\mathbf{G}$, which is then used to compute the GMH. The threshold for edge detection is computed using the 10% rule, i.e. for an $M \times N$ image, edge content is no more than 10% of the image size. Therefore, if we have an $M_v \times N_v$ sized image block $\mathbf{I}_v$, it will have no more than $n_v = 0.1 \times M_v \times N_v$ edge pixels. The threshold for edge detection T_e is set to bin in GMH which has a cumulative bin count till that bin to be equal (or close) to n_v .

We now employ the different gradient maps, i.e. $\mathbf{G}_x$, $\mathbf{G}_y$ and $\mathbf{G}$, in different ways to study the symmetry of edges (if present), and confirm the presence of a vehicle above the detected shadow. We first generate the edge map $\mathbf{E}_v$ by thresholding the *horizontal* gradient map $\mathbf{G}_x$ with T_e . In this process only the horizontal gradients are used because we observe that vehicles predominantly have horizontal edges. Therefore, the horizontal component of the gradients should be more predominant and contributing more to the

overall gradient magnitude. Therefore, the edge map is extracted using the following equation,

$$\mathbf{E}_v := \mathbf{G}_x > T_e \quad (5.16)$$

$\mathbf{E}_v$ is then split into two halves given by $\mathbf{E}_{vl}$ and $\mathbf{E}_{vr}$ for the left and right halves respectively. Fig. 5.14(c), (d) & (e) show $\mathbf{E}_v$, $\mathbf{E}_{vl}$ and $\mathbf{E}_{vr}$ respectively for the chosen input image.

The left and right edge maps $\mathbf{E}_{vl}$ and $\mathbf{E}_{vr}$ are then analyzed for horizontal lines using GAH and Hough Transform (HT). GAH is generated separately for $\mathbf{E}_{vl}$ and $\mathbf{E}_{vr}$ and it is scanned for peaks that correspond to horizontal lines, i.e. at $\Theta = [-90^\circ, -85^\circ]$. When a peak is found in these angle ranges, a straight line edge map (SLEM) is generated thereby filtering all other straight line edges. If a peak is not found in these angle ranges, then no further processing is done with regards to the checking of horizontal edge symmetry.

On finding possible horizontal edges in GAH, we now compute the HT to further ascertain the location of the edges in $\mathbf{E}_{vl}$ and $\mathbf{E}_{vr}$. If these horizontal edges correspond to a vehicle, both the edge maps should show the edges at similar locations. Since we are looking for horizontal lines only, HT is computed for a narrow range of angles given by $\Theta = [-90^\circ, -85^\circ]$. If there are any horizontal lines, HT computed for these angles will pick them up as peaks in the Hough spaces, $\mathbf{HS}_l$ and $\mathbf{HS}_r$. Let $\mathbf{P}_l$ and $\mathbf{P}_r$ be the set of peaks detected in $\mathbf{HS}_l$ and $\mathbf{HS}_r$ respectively and are defined as:

$$\mathbf{P}_l = \begin{bmatrix} (\rho_1^l, \theta_1^l) \\ (\rho_2^l, \theta_2^l) \\ \vdots \\ (\rho_{nl}^l, \theta_{nl}^l) \end{bmatrix} \quad \mathbf{P}_r = \begin{bmatrix} (\rho_1^r, \theta_1^r) \\ (\rho_2^r, \theta_2^r) \\ \vdots \\ (\rho_{nr}^r, \theta_{nr}^r) \end{bmatrix} \quad (5.17)$$

The peaks in $\mathbf{P}_l$ and $\mathbf{P}_r$ are first cleaned up such that peaks which are higher than p_{min} are selected. p_{min} is set based on the width of the image block, i.e. a factor of N_v . This is because the if the image block has a vehicle, the length of horizontal edges in the block will be proportional to the width of the image block.

After shortlisting the peak candidates in $\mathbf{P}_l$ and $\mathbf{P}_r$, we check for symmetry between the left and right peaks. Peak pairs are formed by checking every peak candidate in $\mathbf{P}_l$ with peaks in $\mathbf{P}_r$. A peak $(\rho_i^l, \theta_i^l) \in \mathbf{P}_l$ is said to have formed a peak pair with a peak $(\rho_j^r, \theta_j^r) \in \mathbf{P}_r$ if,

$$\delta_\rho = |\rho_i^l - \rho_j^r| \quad \delta_\theta = |\theta_i^l - \theta_j^r| \quad (5.18)$$

where δ_ρ and δ_θ are small allowances (usually set to less than 5).

These peak pairs imply that there are horizontal lines of similar widths at the same vertical positions in both left and right edge maps. If we get atleast two such peak pairs which are spatially located β pixels away, the signature is confirmed as a vehicle. The additional condition using the parameter β is to ensure that close by lines are not mistaken as two different lines. This parameter β is set based on the height of the image block M_v because the horizontal lines are spatially located along the height of the image block. Fig. 5.14(f) & (g) show the Hough spaces $\mathbf{HS}_l$ and $\mathbf{HS}_r$, respectively. The Hough peaks are detected at

$$\mathbf{P}_l = \begin{bmatrix} (-52, -87^\circ) \\ (-35, -90^\circ) \\ (-18, -90^\circ) \end{bmatrix} \quad \mathbf{P}_r = \begin{bmatrix} (-54, -90^\circ) \\ (-35, -90^\circ) \\ (-18, -90^\circ) \\ (-11, -90^\circ) \end{bmatrix} \quad (5.19)$$

Three peak pairs are found: $\{(-52, -87^\circ), (-54, -90^\circ)\}$, $\{(-35, -90^\circ), (-35, -90^\circ)\}$ and $\{(-18, -90^\circ), (-18, -90^\circ)\}$. These peak pairs also satisfy the minimum separation criteria given by $\beta = 10$ pixels in this example. Since all these conditions are satisfied by the signature found with an under-vehicle shadow, this signature is confirmed to be a vehicle.

In addition to the horizontal edge symmetry, we also check one additional edge symmetry, which is the slant edge symmetry. This is particularly useful in detecting cars. The same can be extended for other vehicles like buses also. The explanation given here is applicable for cars. In this symmetry we consider the slant edges made by the wind shield of the vehicle. It was observed that wind shields cast strong edges because of the contrast they usually have against their surroundings and the vehicle body that encloses the wind shield. Fig. 5.15 is used to illustrate this symmetry. For the vehicle image block shown in the lowest row in Fig. 5.15, the gradients are shown in the first row for the left and right halves of the vehicle image block. It can be seen that the edges of the wind shields form high gradients. This was seen in most daytime images of vehicles. The GAHs in the second row in Fig. 5.15 show that in addition to the horizontal edges (given by the peaks in the middle of the bar plots), there are two additional peaks formed corresponding to the slant edges formed by the wind shield. In the GAH for the left half, the peak is located between 60° to 80° , and for the right half, the peak is found between -80° and -60° . These peaks are circled in Fig. 5.15. When the SLEMs were generated for these angle ranges in GAH, we get the slant edges as shown in the third row in Fig. 5.15. This symmetry in the slant edges between the left and right halves of the vehicle image block is used as an additional cue for confirming the presence of the vehicle. The steps for detecting horizontal edge symmetry, that were discussed previously, can be used again to detect slant edge symmetry for the angle ranges of the

slant edges. These angle ranges can be varied for other vehicles like buses which will usually show vertical edges for the wind shields.

FIGURE 5.15: Illustration on slant edge symmetry. From top: the first row shows the gradient magnitudes of the left and right halves of the vehicle image block (shown in the lowest row); the second row shows the GAHs for the left and right halves of the vehicle image block, with slant edge peaks circled.

FIGURE 5.16: Flowchart for the proposed vehicle detection algorithm.

5.3.3 Overall Algorithm for Vehicle Detection in Lanes

We now describe the entire algorithm for detecting vehicles in lanes that are nearest to the ego-vehicle. Given the lane markings information, we first generate the RoIs corresponding to the lanes present in the image. Starting from the host lane, we divide each RoI into bands. Starting from the bottom-most band, going upwards towards the vanishing point, each RoI band is checked for the presence of an under-vehicle shadow using linear thresholding function described in the previous section. Presence of this shadow is the first cue for the presence of a vehicle. If an under-vehicle shadow is detected in an RoI band, the vehicle signature based on the edge symmetry is checked in the image block that is selected using the location of the under-vehicle shadow. If a vehicle signature is confirmed, then we move to the next RoI. If a vehicle signature is not confirmed, we move upwards to the next RoI band in the same RoI and the whole process is repeated again. Fig. 5.16 shows the flowchart for this algorithm.

5.4 Motorcycle Detection

As shown in literature survey (Section 2.4), motorcycle detection is still a challenging topic. This is primarily because the features of a motorcycle are too varied. There are too many uncertainties and variations in the features of motorcycles that makes them a difficult target to detect using cameras. Fig. 5.17 shows some of the many variations in positions and orientations of the motorcycle on roads. The vehicle detection algorithm that was proposed in Section 5.3 also cannot be directly applied to detect motorcycles. This is because, unlike cars or four-wheel vehicles, motorcycles do not cast a strong umbra shadow on the road surface. Also, they can appear in different positions and orientations in an image scene and hence symmetry of edges as seen in cars is not evident in motorcycles, as shown in Fig. 5.17. Therefore, we propose a novel two-step method to detect motorcycles in lanes using monocular cameras. The first step is a modified version of the shadow detection technique that was proposed previously for vehicles. This step hypothesizes the presence of a motorcycle. In the second step, we propose an intensity based method that further scrutinizes around the possible position of a motorcycle (that is obtained from the first step). A combination of these steps confirms the presence of a motorcycle in the proposed motorcycle detection technique.

Computing Binarization Threshold for Locating Motorcycle Candidates: In order to detect motorcycles, we studied a dataset of images with motorcycles, which are captured from an on-board camera, i.e. a camera on a vehicle behind the motorcycle. Though motorcycles do not cast any shadow on the ground, it was observed that the

FIGURE 5.17: Variations in motorcycle position & orientation.

lower half of the motorcycle is relatively darker than the road surface in most cases. This particular property was also used in [160, 162] to detect motorcycles using a support vector machine. In order to locate this dark region as an initial hypothesis for detecting motorcycle, we use the binarization technique proposed in Section 5.3.1 in a different way. For vehicle detection, we defined the regions A_1 and A_2 , where A_1 is the region on the road surface and close to the tyre of the vehicle, and A_2 is the under-vehicle shadow region. In the algorithm described in Section 5.3.1, we consider the sub-blocks on the corners of each image RoI band as regions A_1 to get the known intensity $\bar{I}_{A_1}$. This is because in the case of a vehicle in the lane, considering the size of the vehicle and a more predictable movement pattern as compared to motorcycles, these sub-blocks are usually unoccupied by the vehicle. However, in the case of a motorcycle, this is not the case. This is because the movement of a motorcycle is less predictable and more wavering. In fact, the corner-most blocks may be covered by the motorcycle in most cases because motorcycles usually overtake between two lanes and hence appear close to the lane boundaries.

Therefore, for motorcycle detection, we redefine the regions A_1 and A_2 . A_1 is the region on the road surface and comprises the entire image RoI band, i.e. $\mathbf{I}_{R_i}$, unlike the corner sub-blocks in the vehicle detection algorithm. A_2 is the region on the tyre of the motorcycle. We conducted a similar experiment as described in Section 5.3.1 with the average intensities of these two regions, i.e. $\bar{I}_{A_1}$ and $\bar{I}_{A_2}$. The scatter plots of $\bar{I}_{A_1}$ versus $\bar{I}_{A_2}$, and $\bar{I}_{A_1}$ versus $\bar{I}_{\Delta} = \bar{I}_{A_1} - \bar{I}_{A_2}$ are shown in Fig. 5.18 using 125 training samples.

It can be seen that the unknown intensities of the tyre also show a similar relationship as those shown by the under-vehicle shadow.

FIGURE 5.18: (a) Scatter plots of $\bar{I}_{A1}$ versus $\bar{I}_{A2}$ and (b) $\bar{I}_{A1}$ versus $\bar{I}_{\Delta}$

Using normal linear regression model given in equation (5.9), we get the linear function as

$$\begin{aligned} h(\theta) &= 0.5012x \\ \Rightarrow \bar{I}_{\Delta} &= 0.5012\bar{I}_{A1} \end{aligned} \quad (5.20)$$

This is used to set the threshold T_b for binarization in every image RoI sub-block to detect the presence of the tyre region. Given an image RoI, i.e. $\mathbf{I}_{\mathbf{R}}$, we divide it into RoI bands. Starting from the bottom RoI band, we move upwards and in a given RoI band $\mathbf{I}_{\mathbf{R}_i}$, we determine the average intensity of the image pixels in $\mathbf{I}_{\mathbf{R}_i}$, i.e.

$$\bar{I}_{A1} = \frac{1}{N_{\mathbf{I}_{\mathbf{R}_i}}} \sum \mathbf{I}_{\mathbf{R}_i}(x, y) \quad (5.21)$$

where $N_{\mathbf{I}_{\mathbf{R}_i}}$ are the number of pixels in $\mathbf{I}_{\mathbf{R}_i}$ and $\mathbf{I}_{\mathbf{R}_i}(x, y)$ is an image pixel in $\mathbf{I}_{\mathbf{R}_i}$. Given $\bar{I}_{A1}$ now, the image RoI band $\mathbf{I}_{\mathbf{R}_i}$ is binarized using T_b obtained from (5.20). In the vehicle detection algorithm in Section 5.3, a minimum number of b blocks need to qualify as white pixel blocks to indicate that there is a vehicle shadow. This value of b is set to a lower value in the case of motorcycle detection. If an RoI band indicates the presence of a motorcycle based on the value of b , we proceed to the next step to confirm the presence of a motorcycle. The next step is important because b being a smaller value (than the vehicle detection case), there can be false positives which could be detected by stray dark patches on the road scene. Such false positives are rejected in the next step.

FIGURE 5.19: Illustrating the blocks around the motorcycle.

Examining Tyre Region: Another visual cue is used in the proposed algorithm to detect motorcycles in lanes. Based on the width w of the binarized region in the image RoI band, and the vertical position y of the detected tyre region, we draw a $w \times h$ sized bounding box B at the vertical position y . Several experiments were conducted to understand the nature of the image content in the bounding box B that should be considered as a motorcycle. One of the observations is that if B is divided into two parts B_1 and B_2 , the lower part B_2 of the motorcycle is relatively darker than the upper part B_1 . However, this is not always true, because the intensity of B_1 block is unpredictable unlike the lower block B_2 . But it was found that B_2 is always darker than its immediate surroundings. Therefore, we define two rectangular regions B_3 and B_4 based on the width and location of the lower block B_2 . These blocks define the road surface region just beside the vehicle. These blocks are shown in Fig. 5.19. Given the average intensities of blocks B_2 , B_3 and B_4 as $\bar{I}_{B_2}$, $\bar{I}_{B_3}$ and $\bar{I}_{B_4}$ respectively, we define two features $\bar{I}_{\Delta l}$ and $\bar{I}_{\Delta r}$ in the following way:

$$\bar{I}_{\Delta l} = \bar{I}_{B_2} - \bar{I}_{B_3} \quad (5.22)$$

$$\bar{I}_{\Delta r} = \bar{I}_{B_2} - \bar{I}_{B_4} \quad (5.23)$$

$$(5.24)$$

Qualitatively these features give the difference of the average intensity of the lower part of the motorcycle and the road surface on either side. It was observed that $\bar{I}_{\Delta l}$ and $\bar{I}_{\Delta r}$ are negative because the lower part of the motorcycle is darker than the road surface.

In order to verify this observation, we conducted an experiment with 275 input training samples to see if these features can be used to create a decision boundary in a two-class classification problem. The two classes are vehicles and non-vehicles. Of these 275 input images, 150 images were valid vehicle images and the remaining 125 images were non-vehicle lane images. In each image, for the vehicle class, the tyre region detection algorithm was used to generate the block B and then B_1 to B_4 . The average intensities are computed in these blocks and then $\bar{I}_{\Delta l}$ and $\bar{I}_{\Delta r}$ are computed using equations (5.22).

For the non-vehicle images, we selected possible ambiguous patches on lanes in a road scene which could be mistaken as vehicle tyre regions. We selected images with strong shadows of trees, overhead fixtures, guard rails etc. to create the training dataset of non-vehicle training data. The two features were computed for the blocks B_2 to B_4 in these non-vehicle image using manually selected coordinates.

The scatter plot in Fig. 5.20 shows the spread of the vehicle and non-vehicle training data samples. The *s indicate non-vehicle data and o's indicate vehicle data. It can be seen from Fig. 5.20 that the non-vehicle samples are concentrated near $\bar{I}_{\Delta l} = 0$ and $\bar{I}_{\Delta r} = 0$ but the vehicle samples are found in the third quadrant, i.e. $\bar{I}_{\Delta l} < 0$ and $\bar{I}_{\Delta r} < 0$. This supports the observation made regarding the motorcycle region B_2 being darker than the road surface adjacent to its left and right (B_3 and B_4 respectively).

FIGURE 5.20: Scatter plot of $\bar{I}_{\Delta l}$ versus $\bar{I}_{\Delta r}$ for vehicle and non-vehicle training data. The red asterisks indicate non-vehicle samples and blue circles indicate vehicle samples.

This scatter plot was used to define a decision function f_{MC} to classify a motorcycle from a non-vehicle input sample. This decision function f_{MC} is defined as the intersection of three different linear functions f_{MC1} , f_{MC2} and f_{MC3} such that,

$$f_{MC} = f_{MC1} \cap f_{MC2} \cap f_{MC3} \quad (5.25)$$

where f_{MC1} , f_{MC2} and f_{MC3} are given as,

$$f_{MC1} : -\frac{30}{17}\bar{I}_{\Delta l} - 30 - \bar{I}_{\Delta r} < 0 \quad (5.26)$$

$$f_{MC2} : \bar{I}_{\Delta r} < 10 \quad (5.27)$$

$$f_{MC3} : \bar{I}_{\Delta l} < 10 \quad (5.28)$$

$$(5.29)$$

The above linear functions are plotted in Fig. 5.21. f_{MC1} gives the linear boundary separating the motorcycle and non-vehicle samples. In addition to f_{MC1} , we also consider f_{MC2} and f_{MC3} to give a tolerance value for $\bar{I}_{\Delta l}$ and $\bar{I}_{\Delta r}$. The intersection of these three decision boundaries will give the overall decision boundary f_{MC} . If the average intensity of B_1 lies in the above decision boundary f_{MC} , then B_1 is likely to be a motorcycle. We call this the second level of confirmation.

FIGURE 5.21: Scatter plot of $\bar{I}_{\Delta l}$ versus $\bar{I}_{\Delta r}$ for vehicle and non-vehicle training data with the linear decision functions f_{MC1} , f_{MC2} and f_{MC3}

In order to further confirm the presence of the signature of the motorcycle, we also look at the relative positions of the horizontal edges in the image block above the motorcycle shadow. In the image block B , we apply the horizontal Sobel gradient operator to get the horizontal gradients. After thresholding the gradient map, we get the horizontal edges. Since motorcycle with the driver is a rigid body, these edges will remain in the

same relative positions with respect to the base of the motorcycle across frames. Fig. 5.22 is used to illustrate this. The top row in Fig. 5.22 shows the motorcycle detected at t -th frame. The horizontal edge map is shown and the prominent edge is found at 0.1288 units with respect to the bottom on the image (the tyre location). The bottom row in Fig. 5.22 shows the motorcycle detected at $t+5$ -th frame. It can be seen that the motorcycle is smaller (moving away from the ego-vehicle). The prominent edge in this case is found at 0.1210 units with respect to the bottom. Since the motorcycle is in the vertical plane, only the scale of the motorcycle changes in the vertical plane. Therefore, the relative position of the horizontal edges will remain similar across frames. Fig. 5.23 summarizes the entire motorcycle detection algorithm with all the three steps described above.

FIGURE 5.22: Illustration showing relative positions of the edges remains fixed as the motorcycle moves across frames. Top row: motorcycle and edge map detected at time t . Bottom row: at time $t+4$ frame.

5.5 Estimating Relative Proximity using Monocular Camera

After detecting vehicles, the next step is to estimate the relative distance between the host and target vehicles. It was shown in the literature survey (Section 2.5) that stereo vision is usually used to determine the depth of an object in an image scene. Some techniques for vehicle distance estimation using monocular cameras were also surveyed in Chapter 2. It was seen that most monocular camera based techniques use target vehicle properties in the image plane like width of the vehicle, aspect ratio of the vehicle, distance between head lights etc. to determine the relative distance in real-world coordinates. The sensitivity and accuracy of the distance measurement is dependent on the accuracy of these parameters. Also, all these methods can be applied for vehicle images only, and are not directly applicable for motorcycles. For example, vehicle width or aspect

FIGURE 5.23: Flowchart for the motorcycle detection algorithm.

ratio based depth estimation cannot be used to determine the distance of a motorcycle in front of the vehicle.

In order to better understand this scenario, an experiment was conducted inside the lab premises. One of the corridors in the lab as shown in Fig. 5.24 was considered as a lane and distance markers were placed a different positions. A camera was moved in different

FIGURE 5.24: Lab experimental setup

directions to study the perspective effects on the lane markers in the image plane. The following observations were made as part of this study:

- The first obvious observation is that the farther the markers are, the lesser is the distance between two markers on the image plane. This is a common observation in all perspective images. Farther objects appear to be smaller than when the same object is close by the camera.
- The vertical displacement of the markers on the image plane is not affected by sideward movements of the camera across the lane (corridor), given that the lane is straight. However, changing the pitch angle of the camera does affect the position of markers in the image plane. Monocular camera based techniques surveyed in Section 2.5 also discuss about this property and address pitch angle variation.
- The main observation is that a relationship between image perspective coordinates and the real-world coordinates needs to be established to determine the relative distance in the real-world coordinates. This relationship defines the perspective effect between the two coordinate systems. In stereo vision based techniques, epipolar geometry is used to determine the depth using the information obtained from two stereo cameras. However, in monocular based methods, the relationship between the two coordinate systems is made based on the image plane information and its *known* real-world information. The monocular based methods presented in the literature survey use target vehicle properties in the image plane like width of the vehicle, aspect ratio of the vehicle, distance between head lights etc. and their corresponding real-world standard measurements to establish this relationship.

Based on these observations, we propose a model for relative proximity estimation using monocular cameras. The proposed model is less dependent on the target vehicle properties because such properties can vary too much depending on the model of the vehicles

ahead. The proposed model is only dependent on the position of the detected vehicle in the image plane and the host lane positions (which were shown to be detected with over 95% accuracy in the previous chapters). Also, the proposed method is developed for straight road cases. Considering that highways are usually straight in nature and lane change is not safe in curved lanes, it is more important to find the proximity in the case of straight roads. If the road curves ahead in the near-view, the lane change is not safe and the proposed risk assessment framework in the next chapter informs this to the driver.

5.5.1 Proposed Model for Relative Proximity Estimation

In this section, we describe a technique to estimate the proximity of the vehicle in a straight lane, that was detected using the techniques discussed above. In order to do this, let us first analyze the camera model and the real-world model of the lanes. Fig. 5.25 shows the camera model and the real-world lane model.

As shown in Fig. 5.25, the real world coordinates are given by the $X - Y - Z$ axes; the camera coordinates are given by the $u - v - w$ axes. Let f be the focal length of the camera. Therefore, the image plane is formed at a distance of f from the center of the camera C . Also, let the camera be tilted by an angle φ as shown in the figure.

FIGURE 5.25: Real world model of the lanes with the camera model

Let l_l and l_r be the left and right lanes respective in the real-world axes. Let the vehicle be at a distance D from the vertical Y axis (which is also the vertical v axis for camera coordinates). Let us now draw a line P_1P_2 along X -axis on the road, intersecting the lanes l_r and l_l at $P_1(X_1, Y_1, Z_1)$ and $P_2(X_2, Y_2, Z_2)$ in the real-world coordinates. This

line is seen as line segment I_1I_2 on the image plane I , with the camera coordinates $I_1(u_1, v_1, w_1)$ and $I_2(u_2, v_2, w_2)$.

Now, we relate these coordinates such that we find the distance D . Using projective coordinates, a camera coordinate (u, v, w) and the real world coordinate (X, Y, Z) can be related in the following manner:

$$\frac{u}{w} = \frac{X}{Z} \quad \frac{v}{w} = \frac{Y}{Z} \quad (5.30)$$

Using the above equation, we can define u_1 and u_2 of the lane marking image coordinates as the following:

$$\frac{u_1}{w_1} = \frac{X_1}{Z_1} \quad \frac{u_2}{w_2} = \frac{X_2}{Z_2} \quad (5.31)$$

With the camera tilted by φ , then the coordinates w_1 and w_2 are given by

$$w_1 = w_2 = f \cos \varphi \quad (5.32)$$

where f is the focal length. Substituting these values in (5.31), we have

$$\frac{u_1}{f \cos \varphi} = \frac{X_1}{Z_1} \quad \frac{u_2}{f \cos \varphi} = \frac{X_2}{Z_2} \quad (5.33)$$

Since points P_1 and P_2 are at the same coordinate on the Z axis, $Z_1 = Z_2$. This relationship along with (5.34) can be used to deduce the following:

$$\begin{aligned} \frac{u_1}{f \cos \varphi} - \frac{u_2}{f \cos \varphi} &= \frac{X_1}{Z_1} - \frac{X_2}{Z_2} \\ \Rightarrow \frac{u_1 - u_2}{f \cos \varphi} &= \frac{X_1 - X_2}{Z_1} \end{aligned} \quad (5.34)$$

In the above equation:

- Z_1 is the actual distance of the vehicle from the camera along Z axis, i.e.

$$Z_1 = D \quad (5.35)$$

- The difference $u_1 - u_2$ is the width of the lane in the image plane and $X_1 - X_2$ is the actual width of the lane in the real-world coordinates, i.e.,

$$W_{im} = u_1 - u_2 \quad W_R = X_1 - X_2 \quad (5.36)$$

Using equations (5.34), (5.35) & (5.36), we get the following relationship:

$$\begin{aligned} \frac{W_{im}}{f \cos \varphi} &= \frac{W_R}{D} \\ \therefore D &= f \cos \varphi \frac{W_R}{W_{im}} \end{aligned} \quad (5.37)$$

In the above equation, the width of the lane on the image plane W_{im} is known from the RoI where we detect the vehicle. The actual width of the lane in real world W_R can be considered as a constant for a certain types of roads. The focal length f and the tilt of the camera φ are also known. Therefore, the actual distance D of the vehicle from the ego-vehicle can be computed using the above relationship. Adding the camera calibration and image size transformation constants as one single constant κ to the above equation, we get

$$D = \kappa f \cos \varphi \frac{W_R}{W_{im}}. \quad (5.38)$$

5.5.1.1 Experimental Verification of the Model

The model was experimentally verified using the test vehicle. In this experiment, the actual distances of the vehicle were measured using a laser gun from different distances D_{actual} . The laser gun [213] has a range of 100m. Also the width of the lane/road where the vehicle was parked was measured manually. From the same points where the actual distances were measured, images of the vehicle were taken using a camera with known calibration constants. In these images, the location of the vehicle was manually selected to compute the width of the lane at those locations. These values were then used to determine the distances using equation (5.38) to give the computed distances D_{comp} . Table 5.1 lists these distances and the errors between D_{actual} and D_{comp} . The test images for which the experiment was conducted are shown in Fig. 5.26. It can be seen that the error found is less than 5m.

FIGURE 5.26: Test images for distance estimation

TABLE 5.1: Experimental results of the distance measurement

	Test Frame	Actual Distance D_{actual}	Computed Distance D_{comp}	Error (m) $D_{comp} - D_{actual}$
(a)	4761	29.263	24.686	-4.577
(b)	5721	35.295	30.857	-4.438
(c)	7608	40.538	36.000	-4.538
(d)	9749	46.791	43.200	-3.591
(e)	12806	13.853	11.444	-2.409
(f)	14696	10.722	8.816	-1.906
(g)	15804	7.330	6.376	-0.954

5.5.2 Estimating Proximity of Vehicles in Neighboring Lanes

The lane width found in the image is a critical parameter in estimating the proximity of the vehicles. The lane width of the neighboring lanes can be more prone to errors owing to their less prominence as compared to the host lane. In order to reduce this error in the distance of vehicles in neighboring lanes, we use the following method. The vehicle is detected in the neighboring lane using the techniques proposed in Section 5.3.3. However, the distance of these vehicles is estimated using the host lane information. If a vehicle is found at the coordinates (u_1, v_1) and (u_2, v_2) in the neighboring lane, we extend this horizontal line to the host lane and find the two points for the host lane and the distance is estimated. This is illustrated using Fig. 5.27. It can be seen from Fig. 5.27 that the distance D computed with respect to the host lane will be the same as the distance of the vehicle on the neighboring lane.

FIGURE 5.27: Top view of a vehicle in neighboring lane and its relationship with host lane

5.6 Results & Discussion

5.6.1 Accuracy Evaluation

Vehicle detection The proposed vehicle detection algorithm was tested on a series of test images to detect vehicles in varying illumination and shadow conditions. There are existing car datasets like Caltech Computational Vision dataset [214] but most of the images in such datasets have the same kind of illumination. They are usually taken on bright days. Some datasets in Caltech dataset do not apply to the LCDAS application because they were created for vehicles in car parks. Therefore, we generated our own dataset. A dataset containing images with vehicles in different lanes was created from the datasets that were described in Table 3.1 in Section 3.3 in Chapter 3. Images were selected from these datasets (which capture different road conditions) that had vehicles in different lanes. The images were captured from video sequences in Table 3.1 that were taken during different times of the day, different weather conditions like sunny, cloudy, overcast, rainy etc. In each of the image, the lanes were pre-determined using the proposed techniques and the RoIs were annotated. The proposed vehicle detection algorithm was applied on all the different RoIs. A total of over 665 RoIs with and without vehicles were tested. The image dataset is hosted at [215]. The y coordinate of the bottom of the vehicle in each input image was annotated as ground truth data.

The proposed algorithm was applied on the different images in the dataset. It was ensured that the linear function binarization parameters and the other block parameters described in Section 5.3 are maintained the same across all images in the dataset so that the adaptability of the algorithm to varying conditions can be tested. A detection rate of nearly 95% was seen. A detection is considered as correct if the algorithm either flags a true positive (when there is vehicle in the RoI) or a true negative (when the vehicle is not present in the RoI). If the position of the vehicle detected by the proposed algorithm was within $\epsilon_y = 10$ pixels of the ground truth data, it was considered a correct detection.

Some sample vehicle detection results are shown in Fig. 5.28, wherein vehicles are detected using the proposed techniques in different road scenarios. It can be seen that the proposed method is able to detect vehicles in both near and far-views. The detection results in Fig. 5.28 (e)-(h) show that proposed algorithm detects the vehicles in the presence of shadows and other road markings also.

A study on the mis-detections showed that the proposed method failed to detect the vehicles in heavy rain scenarios or when the wet road becomes reflective after rain. This is because the binarization threshold of the under-vehicle shadow in such scenarios does not follow the proposed linear regression model. The algorithm was able to detect vehicles

FIGURE 5.28: Some sample vehicle detection results in varying road scenarios in host lane: vehicles in far-view in (a) & (b), vehicles on bright roads in (c) & (d), vehicles in the presence of shadows of trees in (e), vehicles in the presence of shadows and road markings in (f), and vehicles in the presence of other road markings in (g) & (h)

in late evening times and also during night times in some cases but it failed to detect the presence of an under-vehicle shadow when the host vehicle and the front vehicle move too close to each other. This is because the lighting from the headlights of the host vehicle brightens the under-vehicle area of the target vehicle in front. Consequently, the under-vehicle shadow is not found. More details about nighttime vehicle detection is presented later in this section.

The proposed proximity estimation algorithm was also applied on real road scene images. It was practically not feasible to verify the proximity estimates using the hand-held radar guns while the vehicle was moving on highways. Therefore, we performed a visual test of the estimated proximities. We applied the detection algorithm on a video sequence with vehicles in front of the host-vehicle moving, either away from the host-vehicle or approaching host-vehicle. If the target-vehicle is moving away from the host-vehicle,

the estimated proximity values from the algorithm should show an increasing trend, and vice versa for the approaching vehicles. This increasing and decreasing trends were visually verified on a series of images. Some proximity estimation results are marked on individual images in Fig. 5.28.

Motorcycle detection The proposed motorcycle detection technique was tested using test images taken under different lighting and weather conditions. The test image database had two different sets of images. First set had motorcycles in lanes and the second set was empty lanes. The ground truth data is generated by going through each image and manually determining the y coordinate of the point where the tyre is perceived to be touching the road surface in the image. A true positive (TP) is defined as correct detection of the motorcycle in terms of the location along y axis. If a motorcycle is absent and the algorithm results in a negative, then it is a true negative (TN). Similarly the false positives (FP) and negatives (FN) are defined. These are used to define the detection rate as:

$$D = \frac{TP + TN}{TP + TN + FP + FN} \times 100 \quad (5.39)$$

The proposed algorithm gives nearly 90% detection rate on a test dataset comprising 105 test images. The robustness of the proposed method was found to be dependent on the following factors:

- The first step of hypothesizing the tyre position using the binarization threshold is vital for the rest of the detection process. It was found that the proposed method is effective in detecting motorcycles close to the host vehicle at a distance of less than 30 meters. When the motorcycle is farther than this, the tyre feature is too narrow for the proposed algorithm to conclusively detect the motorcycle. However, in a lane change assistance system, it is important that the motorcycles close to the host vehicle be detected. A vehicle that is closer to the host vehicle is of higher risk for lane change than a vehicle farther away. Therefore, if the detection RoI is restricted to near view of 20-30 meters, the proposed algorithm has higher accuracy.
- The next item is related to the interference of other vehicles in the vicinity of the motorcycles. If we consider the RoI as defined above, there is no interference of other vehicles on detecting the tyre region of the motorcycle. This also ensures that blocks B_3 and B_4 , in the second step of the proposed algorithm, cover the road surface and are not corrupted by intensities of the surrounding vehicles. It was observed that if the motorcycle is present in the far view, the intensities of other surrounding vehicles in far-view interfere in computing the average intensities around the motorcycle. This results in possible false negatives.

- When the intensities of the road surface are as dark as the motorcycle region B_2 , then the proposed algorithm fails. However, such cases are fewer in number. In the datasets that were considered, there only a few meters of such dark road surfaces where the tar is being re-laid on the road surface.

Note on detection during night time : The proposed techniques for vehicle and motorcycle detection were derived primarily for day time scenarios. Though the vehicle shadow appears during night time also, the intensities of the shadows, both under-vehicle and other shadows, during night time change depending on the light source. In the presence of streetlights, the shadow strengths appear differently. Also, if the target vehicle is close to the host vehicle, the lighting from the headlights of the host vehicle usually removes the under-vehicle shadow of the target vehicle in front of the host vehicle. Therefore, applying the proposed techniques directly on the night time scenarios will not give high detection rates. However, vehicle detection during night time is considered as a relatively less complex problem as compared to day time, which is the reason for a large amount literature (as shown in Chapter 2) on vehicle detection during day time. This is because vehicles can be distinctly detected using the braking lights and headlights during night time, which form strong visible features during night time. This is not the case during day time. Techniques for vehicles during night time can be combined with the proposed lane-restricted vehicle detection methods to further improve the robustness and reducing computational complexity.

5.6.2 Computational Complexity Analysis

In this subsection, the computational complexity of vehicle detection process is discussed in detail. We also compare the computation cost of the proposed vehicle detection method against existing similar recent vehicle detection techniques. The proposed method comprises of two parts: (1) under-vehicle shadow detection, and (2) vehicle signature confirmation. We will compare the proposed method comprising the above-mentioned steps against methods proposed in literature such as [140, 151].

For the sake of equivalent comparison, we consider that the same region of interest in which the vehicles will be detected. Let us consider a lane region in which vehicle detection is to be performed. Let the vanishing point be present at $V_{pt}(x_{pt}, y_{pt})$ where y_{pt} is along the vertical axis, with origin on the top left corner of the image. Therefore, if the image size is $M \times N$, then height of the lane region seen in the image is $h_{roi} = N - y_{pt}$. Let w_{roi} be the lane width at the bottom of the image, i.e. at $y = M$. This results in a triangular RoI for the lane region with the base and height equal to w_{roi} and h_{roi} respectively.

Let us consider the worst possible computation cost to determine the vehicles in the proposed method. In such a scenario, the entire lane RoI needs to be scanned for the under-vehicle shadow region in the first step of the proposed method. If w_r is the width of each block in the bands for shadow detection, the total cost of finding the under-vehicle shadow using the proposed method is given by:

$$\begin{aligned} C_{ADD}^1 &= 2w_r h_{roi} \\ C_{COMPARATOR}^1 &= \frac{1}{2} w_{roi} h_{roi} \end{aligned} \quad (5.40)$$

After determining the under-vehicle shadow, the vehicle signature confirmation step using edge symmetry is used if a possible under-vehicle shadow is found. Given that the bounding box of the vehicle is $M_v \times N_v$, the cost of confirming the vehicle signature can be derived as:

$$\begin{aligned} C_{ADD}^2 &= n_v + p_v n_v n_{\theta=90} \\ C_{COMPARATOR}^2 &= p_{pks}^2 \end{aligned} \quad (5.41)$$

where $n_v = 0.1M_v N_v$ is the number of edge pixels in the vehicle bounding box above the detected under-vehicle shadow, p_v is the number of edge pixels along horizontal edges, $n_{\theta=90}$ are the number of angles around 90° for which Hough transform is computed, and p_{pks} is the number of Hough peaks detected in each half of the bounding box corresponding to horizontal lines.

We will use the above formulations to compare the proposed method with two existing vehicle detection methods. We first compare the proposed method with a similar method in [151] that uses under-vehicle shadow as a pre-processing step to generate a hypothesis of the presence of a vehicle. It then uses an SVM classifier to confirm the presence of the vehicle. Let us consider the cost for the shadow detection algorithm [151]. This requires the computation of the mean μ and variance σ^2 of the pixel intensities in the RoI, and the pixel intensities are thresholded using the threshold set to $\mu - 3\sigma$. The computation cost for this method is given by:

$$\begin{aligned} C_{ADD}^{E1} &= \frac{3}{2} w_{roi} h_{roi} \\ C_{MUL}^{E1} &= \frac{1}{2} w_{roi} h_{roi} \\ C_{COMPARATOR}^{E1} &= \frac{1}{2} w_{roi} h_{roi} \end{aligned} \quad (5.42)$$

After detecting the shadow, the method in [151] uses an SVM to further confirm the presence of a vehicle in a bounding box above the detected shadow. This requires additional computations to generate the feature vectors such as HOG features. We

discount this computation cost from this method and we will still be able to show that the proposed method that includes both shadow detection and vehicle signature confirmation has lower computation cost as compared to the [151]. For the sake of comparison, we consider the following values for the different variables in the above equations (these values are taken based on observations from real images): $w_r = 20$, $h_{roi} = 180$, $w_{roi} = 400$, $M_v = N_v = 400$ (worst case scenario), $n_v = 0.1M_vN_v = 16000$, $p_v = 0.5$ (worst case). It can be seen from Table 5.2 that although the proposed method has less than 1% more comparators, it has nearly 5% lesser number of addition operations and no multiplication operations. It should also be noted that this comparison does not include the cost of the vehicle confirmation stage in [151]. It can be seen that even after discounting the vehicle signature confirmation stage in [151], the proposed method outperforms the method in [151] in terms of computational cost.

TABLE 5.2: Comparison of computation complexity

Operations	[151]	Proposed Method
Additions	108000	103200
Multiplications	36000	0
Comparators	36000	36100

We now consider appearance based classification technique such as the one presented in [140] in IEEE IV 2012. The following computation cost is applicable to most appearance based vehicle detection methods, which are commonly used for vehicle detection. In such methods, a moving window of multiple scales is run over the region of interest to extract feature vectors. These feature vectors are then sent to an SVM. In [140], HOG-like features are extracted, which are used to train an SVM. The weights obtained from the SVM are then used to classify if a feature vector generated from the window is a vehicle or not. The computation cost of such methods depends on the number of scales of vehicles that will be detected. For the sake of comparison let us assume we want to detect a vehicle in a 100×100 window, and the windows are moved in a overlapping fashion such that 25% of the window is overlapping with its adjacent window. In the triangular lane region described previously with $h_{roi} = 180$ and $w_{roi} = 400$, we will have $n_{wins} = 1440$ number of windows. In each window, if we compute a HOG feature (though the actual HOG like feature vector in [140] is computationally more complex than a simple HOG), then cost of computing a HOG feature in a 100×100 window is given by 10000 additions. This corresponds to the histogram computation using gradient values for the HOG. Therefore, the number of additions alone for the appearance based method is over 14 million. This cost itself is orders of magnitude higher than the proposed method. Additionally, there is a cost of classification using SVM. Usually, HOG vectors, each comprising 72 values, are used for a 4×4 celled window as shown in [140]. The SVM classification cost for such a HOG vector is given by $n_{wins} \times 72$ inner

product computations followed by similar number of comparisons. Each inner product computation is a sum of products operation resulting in 72 multiplication and addition operations. Using the above values, we tabulate the comparison of computation cost between the proposed method and the appearance based method in [140] in Table 5.3. It can be seen that the overall computation cost is reduced by orders of magnitude as compared to [140].

TABLE 5.3: Comparison of computation complexity with appearance based vehicle classifiers

Operations	[140]	Proposed
Additions	14400000	103200
Multiplications	103680	0
Comparators	1440	36100

5.7 Computational Architecture for Vehicle Detection

The proposed vehicle detection algorithm enables region-level parallelism. Each lane in the image can be considered as a region in which vehicle detection can be performed independently of the other regions, i.e. after finding the lane boundaries of the host and neighbor lanes, the lane regions enclosed by each lane can be processed in parallel. This is possible because the proposed vehicle detection algorithm uses the lane information for localizing the vehicle signatures in the first step of under-vehicle shadow detection. In contrast, existing vehicle detection methods like [140] scan the entire image region below the vanishing point, which is done by a moving window approach is applied serially on image coordinates. Fig. 5.29 shows a possible computational framework for vehicle detection which employs multiple processing engines that can operate in parallel. After obtaining the lane positions in the image, each region processor (RP) operates in its corresponding lane region and detects the nearest vehicle localized in the lane.

5.8 Summary

In this chapter, techniques were proposed to detect vehicles and motorcycles in lanes. The proposed techniques were shown to be adaptive to local illumination changes and hence robustly detect the target vehicles based on the scene around them in lanes. Detection rates as high as 95-96% were seen in the case of vehicles and about 90% in the case of motorcycles. In addition to detection, the proximity of the vehicles was estimated using the width information of the lanes on the image plane. A 3D model relating the image perspective coordinates and the real-world coordinates using the lane width was

FIGURE 5.29: Possible computational framework for vehicle detection using parallel processing engines for different lane regions

proposed and it was shown that the model gives an estimate of relative distance with an error of not more than 5m in an experimental setup. The proposed vehicle proximity estimation method was shown to be independent of vehicle characteristics and hence can be applied to both vehicles and motorcycles.

The vehicle positions detected using the proposed techniques will be used to propose an overall LCDAS system in the next chapter. Different parameters of the target vehicles will be defined using their relative proximities and their positions in lanes. These parameters will be used to develop a framework for an LCDAS in the next chapter.

C H A P T E R 6

Techniques for Lane Change Decision Aid

6.1 Introduction

In this chapter, we propose techniques that aid the driver in taking lane change decisions. The proposed techniques use the information obtained from the different visual cues from the road scene. We define the following main objectives of the proposed LCDAS, that are concurrent with the goals of an LCDAS by ISO standards [36]:

- It should constantly monitor the following areas of interest for the presence of any vehicles: (1) lanes in front of the host-vehicle, (2) lanes in the rear of the host-vehicle and (3) blind spot areas of the host-vehicle.
- The relative distances and speeds of vehicles found in these areas of interest are *monitored* continuously.
- The relative distances and speeds of these vehicles are *analyzed* continuously to determine if it is safe to change lane. This decision will be based on the risk that is involved in changing lanes at that time instant. This risk needs to be computed based on the state of the road scene.

The different techniques proposed in Chapters 3, 4 and 5 will be brought together to accomplish the above tasks of the proposed LCDAS. In order to do this, we first define state parameters for different RoIs in the road scene. The state parameters tell the state of the lane or RoI, and these state parameters are then analyzed to determine the

risk involved in changing lanes. We first propose a sensor configuration that enables monitoring of the road scene around the host vehicle. The road scene, as captured by the proposed sensor configuration, is then analyzed to determine the state of the each RoI, which are then used to find the risk in changing lanes. The forthcoming sections will describe the proposed techniques with illustrations on real road scenes.

6.2 Sensor Configuration for Surround Scene Sensing

FIGURE 6.1: Sensor configuration in the proposed LCA system showing front and rear cameras, C_F and C_R respectively. The front and rear RoIs are indicated as shown. The blind spots where the cameras cannot sense are given by the striped shaded areas.

In an LCDAS, the road scene surrounding the host-vehicle must be continuously monitored. In order to do this, we propose a sensor configuration with two cameras. The first camera senses the road scene in the front of the host-vehicle. The second camera captures the road scene in the rear of the host-vehicle. Fig. 6.1 shows the two cameras C_F and C_R capturing the front and rear road scenes of the host-vehicle and the different regions of interest listed in the objectives of lane change assistance in Section 6.1, i.e. the front lanes, the rear lanes and the blind spot areas. The front lanes are represented by F_i , where $i = 0$ represents the host lane, a positive i represents the i^{th} right lane and a negative i represents the i^{th} left lane. Therefore, if there is one lane to the left and one lane to right, the lanes will be represented as F_{-1} , F_0 and F_{+1} , as shown in Fig. 6.1. These front lanes are monitored by the front facing camera C_F . Similarly, the

lanes in the rear road scene are given by R_i , which are continuously monitored using the rear facing camera C_R . The blind spots of the vehicle are represented by B_i , where $i = -1$ represents the left blind spot and $i = +1$ represents the right blind spot. For each RoI F_i and R_i seen by the front and rear cameras respectively, we determine the state parameters that describe the state of the RoI. these state parameters are denoted by $\Psi_{F_i}^{(t)}$ and $\Psi_{R_i}^{(t)}$ respectively. It can be seen in Fig. 6.1 that the blind spots B_{+1} and B_{-1} are not covered in the field of view of C_F and C_R . In the subsequent sections, we will show how the state parameters from front and rear cameras are used to describe all the RoIs around the vehicle, including the blind spot areas.

6.3 Defining Vehicle State Parameters

In order to analyze the environment around the host-vehicle, we define *state parameters* for different RoIs in the road scene. A *state parameter* for an RoI L at a time instant t is represented by $\Psi_L^{(t)}$ and defined as the following:

$$\Psi_L^{(t)} = \begin{pmatrix} p & d & v \end{pmatrix}^T \quad (6.1)$$

Each variable in $\Psi_L^{(t)}$ are also independently represented with time stamp and RoI as $p_L^{(t)}$, $d_L^{(t)}$ and $v_L^{(t)}$. In the above equation, p represents the presence or absence of a vehicle in the RoI L at the time instant t . The values that p can take are

$$p = \begin{cases} 1 & \text{if vehicle is found} \\ 0 & \text{if vehicle is not found} \end{cases} \quad (6.2)$$

The variable $d^{(t)}$ in equation (6.1) gives the relative distance of the vehicle (if found) with respect to the host-vehicle in L at time instant t . This value is in meters. If a vehicle is not found, $d = 0$. The variable $v^{(t)}$ in equation (6.1) represents the relative velocity of the vehicle found in L and is computed as:

$$v^{(t)} = \frac{d^{(t)} - d^{(t-1)}}{\Delta t} \quad (6.3)$$

where $\Delta t = (t - 1) - t$. In the above equation, it should be noted that $d^{(t)}$ is the *relative distance* between the host-vehicle and target vehicle at time t , which means that it takes into account the actual distances covered by each vehicle and the difference between their real world coordinates is quantified in $d^{(t)}$. Qualitatively, $v^{(t)}$ in (6.3) can be understood in the following way. If $v^{(t)}$ is positive, then $d^{(t)} > d^{(t-1)}$. This implies that the distance between the host-vehicle and the vehicle ahead (target vehicle) is increasing, i.e. either the host-vehicle is slowing down or the target vehicle is increasing its speed or both.

Similarly, if $v^{(t)}$ is negative, then $d^{(t)} < d^{(t-1)}$ implying the host-vehicle is pacing up or the target vehicle is slowing down or both, leading to the decreasing distance between the two vehicles. Summarizing these observations, we have the following values that $v^{(t)}$ can take:

$$\begin{aligned} v^{(t)} > 0 &\Rightarrow \text{Target vehicle is moving away from host-vehicle} \\ v^{(t)} < 0 &\Rightarrow \text{Target vehicle is moving closer to host-vehicle} \\ v^{(t)} = 0 &\Rightarrow \text{Both vehicles are moving at steady speed (given } p^{(t)} = 1) \end{aligned} \quad (6.4)$$

Given these variables, we define the state parameters for different RoIs in the road scene, that need to be monitored for arriving at a decision before changing lanes. The state parameter of any RoI L is represented by $\Psi_L^{(t)}$ defined by equation (6.1). Referring to the lanes shown in Fig. 6.1, the state parameters for three lanes or RoIs in the front, i.e. F_{-1}, F_0, F_{+1} , and three lanes in the rear, i.e. R_{-1}, R_0, R_{+1} are given by: $\Psi_{F_{-1}}^{(t)}, \Psi_{F_0}^{(t)}, \Psi_{F_{+1}}^{(t)}$ for front lanes, and $\Psi_{R_{-1}}^{(t)}, \Psi_{R_0}^{(t)}, \Psi_{R_{+1}}^{(t)}$ for rear lanes. As defined in the beginning of this section, if we want to refer to a state variable, say distance d , of a particular RoI (say R_0) at a time instant t , we represent it by $d_{R_0}^{(t)}$.

In addition to these state parameters, we define a different state parameter for the blind spots. This is represented by $\gamma_{B_i}^{(t)}$ and defined as follows:

$$\gamma_{B_i}^{(t)} = \begin{cases} 1 & \text{if vehicle is blind spot area} \\ 0 & \text{if vehicle is not in blind spot area} \end{cases} \quad (6.5)$$

These state parameters for different RoIs will be now be used to determine the risk of changing lanes at a given time instant t .

6.4 State-dependent Risk Factor Computation

In this section, we introduce a risk factor ξ , which estimates the amount of risk a target vehicle poses to the host-vehicle. This risk factor is determined for every target vehicle in every RoI that is considered in the LCDAS. If the driver intends to change lane, this risk factor tells whether or not it is safe to change lane at that instant of time by determining the risk a particular target vehicle poses on the ego vehicle. By combining the risk factors from all the vehicles around the host-vehicle, we then look at the overall risk in changing lanes at a given time instant.

We first describe how this risk factor is determined for a single target vehicle. Let TV being the target vehicle at a relative distance $d^{(t)}$ moving at a relative velocity $v^{(t)}$. We

define the risk factor for this vehicle based on the following observations about the state parameters:

Relative distance $d^{(t)}$: If we consider two target vehicles TV_1 and TV_2 at distances $d_1^{(t)}$ and $d_2^{(t)}$ respectively such that $d_1^{(t)} < d_2^{(t)}$, the risk posed by TV_1 is greater than TV_2 . This is shown in Fig. 6.2. In other words, the risk posed by TV_1 is higher than TV_2 . Therefore, if the distance $d^{(t)}$ of a target vehicle is lower, the risk is higher, i.e. $d^{(t)} \downarrow \Rightarrow \xi \uparrow$. However, the distance state parameter alone does not completely define the risk factor.

FIGURE 6.2: Effect of relative distance on the risk factor

Relative velocity $v^{(t)}$: The proposed risk factor is also dependent on the relative velocity of the target vehicle with respect to the ego vehicle. Fig. 6.3 shows two different states of the target vehicle with respect to the ego vehicle. In Fig. 6.3 (a), the relative velocity $v_1^{(t)}$ is given by

$$v_1^{(t)} = \frac{d^{(t)} - d^{(t-1)}}{\Delta t} \quad (6.6)$$

In this case since $d^{(t)} < d^{(t-1)}$, the relative velocity is less than 0, i.e. $v^{(t)} < 0$. A negative $v^{(t)}$ implies EV and TV are approaching each other. Similarly the case shown in Fig. 6.3 (b) gives a positive relative velocity, i.e. $v^{(t)} > 0$, which implies EV and TV are moving away from each other. In this case, it is evident from the distances

FIGURE 6.3: Effect of relative velocity on the risk factor (a) $v_1^{(t)} < 0$ (b) $v_2^{(t)} > 0$

that the risk is higher in case (a) when the relative velocity is negative. In other words, for the same distance at time t , consider two scenarios as shown in Fig. 6.4. In this scenario, the distance at t is the same in both cases (a) and (b), i.e. $d_1^{(t)} = d_2^{(t)}$. In both cases, the two vehicles are approaching each other, i.e. $v_1^{(t)} < 0$ and $v_2^{(t)} < 0$. But the relative speed of approach is not the same, i.e. TV is approaching EV faster in case (b) than in case (a), i.e. $|v_1^{(t)}| < |v_2^{(t)}|$, resulting in a higher risk factor in case (b).

Therefore, the relative velocity state parameter $v^{(t)}$ affects the risk factor ξ in the following two ways:

- If $v^{(t)} < 0$, the risk is higher, if the magnitude of relative velocity is higher, i.e. $v^{(t)} < 0 \wedge |v^{(t)}| \uparrow \Rightarrow \xi \uparrow$.
- If $v^{(t)} > 0$, the risk is still there but lower, i.e. $v^{(t)} > 0 \Rightarrow \xi \downarrow$.

Determining Risk Factor using State Parameters Based on these observations, we define the risk factor at time t as a function of $v^{(t)}$ and $d^{(t)}$, i.e.,

$$\xi^{(t)} = f(d^{(t)}, v^{(t)}) \quad (6.7)$$

where $f(\cdot)$ is the risk function. From the above observations, we established that the risks posed by vehicles in front and rear of the host-vehicle are equal, and the vehicles

FIGURE 6.4: Effect of relative velocity on the risk factor with $|v_1^{(t)}| < |v_2^{(t)}|$

further away are of lesser risk than closeby vehicles. Based on this rationale, we derived an analogy to the Gaussian curve and modeled the risk function as a Gaussian curve. This risk function is defined as a Gaussian function centered at $\mu = 0$ with a standard deviation of σ , i.e.,

$$\xi^{(t)} = \exp\left(\frac{-d^{(t)^2}}{2\sigma^2}\right) \quad (6.8)$$

We choose a Gaussian curve because the risk posed by vehicles in front and rear of the ego vehicle is equal. The distance of the ego-vehicle from the target vehicle is considered along the x -axis in the risk function. Therefore, if the vehicle is approaching from rear, the distance is considered as negative (and positive if a vehicle is in front of the ego-vehicle). The risk by either vehicle is considered to be equal. In order to model this behavior, a Gaussian curve is considered for the risk function. However, this function can be changed depending on the risk we want to give to the vehicles approaching from different directions. For example, if we want to give a higher risk to vehicles approaching from rear as compared to those from front of the ego-vehicle, a Rayleigh curve can be considered, which is steeper on one side compared to the other. In this thesis, we consider a Gaussian curve with equal risk for vehicles from front and rear.

Fig. 6.5 plots $\xi^{(t)}$ versus distance $d^{(t)}$ on the x -axis using the above Gaussian function. In Fig. 6.5, let us consider the positive values of $d^{(t)}$. It can be seen that as d goes towards zero the risk factor increases in magnitude.

FIGURE 6.5: Gaussian curve representing the proposed risk function $f(\cdot)$.

Given this risk function, we now bring the next parameter to dynamically change the risk factor based on the vehicle speed. As discussed earlier, if a vehicle is approaching at a higher speed, the risk factor should increase for the same distance d . In order to do this, the standard deviation σ in (6.8) is made a function of the velocity v , i.e.

$$\sigma^{(t)} \propto f(v^{(t)}) \quad (6.9)$$

Fig. 6.6 shows the effect of changing σ on the risk factor. It can be seen that for an approaching vehicle from rear view, for the same distance $d = 10m$, $\sigma_2^{(t)} > \sigma_1^{(t)} \Rightarrow \xi_1^{(t)} > \xi_2^{(t)}$. The σ value being directly proportional to $v_e^{(t)}$, increased velocity implies higher σ , and hence higher risk.

FIGURE 6.6: Gaussian risk functions showing increased risk for the same distance.

We now define this function f that relates the risk function with the approach velocity $v^{(t)}$ and also the distance between the two vehicles. In order to do this, we refer to the stopping distance that is recommended based on the reaction times and the ego vehicle velocity $v_e^{(t)}$. This data is taken from The 2012 Official Handbook on Basic Theory of

Driving by Singapore Traffic Police [216], which computes this data based on real test drives and driver behavior information. Fig. 6.7 plots the minimum stopping distance versus the host-vehicle velocity. For a given $v_e^{(t)}$, the curve gives the minimum stopping distance d_{min} that is required by host-vehicle to stop and avoid any collision with the target vehicle. It can be seen that d_{min} can be derived as a function of $v_e^{(t)}$, i.e.

$$d_{min} = f(v_e^{(t)}) \quad (6.10)$$

Considering that these are constant values, we can get a quadratic relationship between the two quantities. However, in order to reduce the computations, we can also have simple look-up table (LUT) given that the number of entries is limited. For the simulations and experiments presented in this chapter, we used an LUT.

FIGURE 6.7: Minimum stopping distance versus host-vehicle velocity

Now, let us consider the case when we have a target vehicle moving at a velocity of $v_o^{(t)}$ in front of the host-vehicle. In this scenario, assuming that in the short Δt time, both vehicles are maintaining the same speed (i.e. there is no acceleration), we have the following minimum safe distance $d_s^{(t)}$:

$$d_s^{(t)} = d_{min}^{(t)} - v_o^{(t)} \Delta t \quad (6.11)$$

Given the relative velocity $v^{(t)} = v_e^{(t)} - v_o^{(t)}$, equation (6.11) can be modified as:

$$d_s^{(t)} = f(v_e^{(t)}) - (v_e^{(t)} - v^{(t)}) \Delta t \quad (6.12)$$

In the above equation, $d_s^{(t)}$ is the safe distance for stopping for the host-vehicle w.r.t the target vehicle in front of it. This is dependent on the host-vehicle velocity v_e and the relative velocity v . In other words, if the target vehicle is at a distance $d > d_s$, then there is enough stopping distance between the host-vehicle and the target vehicle. Hence, it is safe to change lane with this distance as the safe distance.

Now, we will show how this safe distance can be used to quantify the risk factor and give the driver three levels of risk based on the relative distance d , relative velocity v , safe distance d_s and the Gaussian risk function defined in (6.8). We consider three kinds of safety notification (S) to the driver:

1. Safe to change lane, $S = 1$
2. Change lane with caution, $S = 0$
3. Danger - don't change lanes, $S = -1$

In order to assign the level of safety, i.e. S , we first define two safety risk factors ξ_{s1} and ξ_{s2} such that $0 < \xi_{s1} < \xi_{s2} < 1$. These factors are the alarms levels that are fixed but the risk factor computed for the scene changes dynamically based on the risk function. The shape of the risk function will change dynamically based on the state parameters of the target and ego vehicles but the risk notification values remain the same. We will show how this is done.

The safety risk factors and safety levels are related to each other in the following manner:

$$\begin{array}{l|l|l}
 0 < \xi \leq \xi_{s1} & S = 1 & \text{Safe to change lane} \\
 \xi_{s1} < \xi \leq \xi_{s2} & S = 0 & \text{Change with caution} \\
 \xi_{s2} < \xi \leq 1 & S = -1 & \text{Danger - don't change}
 \end{array} \quad (6.13)$$

In the above definitions, ξ_{s1} corresponds to the safe distance d_s that we obtained from the current state of the ego and target vehicles in (6.12). Therefore, the Gaussian risk function should give ξ_{s1} for distance d_s as shown in Fig. 6.8. Depending on the state parameters of the ego and target vehicles, the safe distance measure d_s changes, which also results in a different Gaussian risk function for the given safety risk factor ξ_{s1} . This is illustrated in Fig. 6.8 for two different d_s that happen at time instants t and t' . In order to have the risk factor set to ξ_{s1} at safe stopping distances d_s^t and $d_s^{t'}$, we will get two different risk functions, f^t and $f^{t'}$, as shown in Fig. 6.8. It can be seen from Fig. 6.8 that if we take a target vehicle distance d such that $d_s^t < d < d_s^{t'}$, we will see that at time t since $d > d_s^t$, the target vehicle will have a risk factor less than ξ_{s1} using the risk function f^t . However, for the same distance d at time t' , we will get a risk factor greater than ξ_{s1} using the second Gaussian risk function $f^{t'}$.

Now, given d_s and ξ_{s1} , we find the shape of the Gaussian risk function $f(\cdot)$. In order to do this, the variance of the function needs to be determined, which is derived as shown

FIGURE 6.8: Gaussian risk functions as d_s changes.

below:

$$\begin{aligned}\xi_{s1} &= \exp\left(\frac{-d_s^2}{2\sigma^2}\right) \\ \log(\xi_{s1}) &= \frac{-d_s^2}{2\sigma^2} \\ \sigma^2 &= -\frac{1}{2} \frac{d_s^2}{\log(\xi_{s1})}\end{aligned}\tag{6.14}$$

Substituting the value of d_s from (6.12) into the above equation and putting back the time stamp t , we have

$$\sigma^{2(t)} = -\frac{1}{2} \frac{\left(f(v_e^{(t)}) - (v_e^{(t)} - v^{(t)})\Delta t\right)^2}{\log(\xi_{s1})}\tag{6.15}$$

Given this variance at time instant t of the Gaussian risk function, we can compute the risk factor $\xi^{(t)}$ due to a target vehicle at a relative distance $d^{(t)}$ traveling at a relative velocity $v^{(t)}$ by substituting $\sigma^{2(t)}$ into the Gaussian risk function, i.e.,

$$\begin{aligned}\xi^{(t)} &= \exp\left(\frac{-d^{(t)2}}{2\sigma^{2(t)}}\right) \\ \xi^{(t)} &= \exp\left(\frac{-d^{(t)2}}{-\frac{d_s^{(t)2}}{\log(\xi_{s1})}}\right)\end{aligned}\tag{6.16}$$

Applying properties of exponential to the above equation, we get the risk factor at t as:

$$\xi^{(t)} = (\xi_{s1})^{\frac{d^{(t)^2}}{d_s^{(t)^2}} \quad (6.17)$$

It can be seen from the above equation that the risk factor $\xi^{(t)}$ computation does not involve any exponentiation but a simple power transform function involving dynamic parameters $d_s^{(t)}$ and $d^{(t)}$, which depend on the state parameters of the scene. This risk factor $\xi^{(t)}$ is now used with the ranges, defined by ξ_{s1} and ξ_{s2} in (6.13), to find if it is safe to change lane or not. Fig. 6.9 shows the flowchart to compute the risk factor at time instant t using the state parameters. The Gaussian risk function shown in Fig. 6.9 illustrates how the relative distance $d^{(t)}$ results in a risk factor $\xi^{(t)}$ from the Gaussian function.

FIGURE 6.9: Flowchart for computing the risk factor at t using state parameters.

Fig. 6.10 illustrates the computation of the risk factor on an image sequence. $d^{(t)}$ and $v^{(t)}$ are plotted in the first two plots in Fig. 6.10. The third plot gives the risk factor for the target vehicle. It can be seen from the Fig. 6.10 that risk factor increases when $v^{(t)}$ is negative and when the relative distance reduces, implying an approaching vehicle towards the host-vehicle. It can be seen that when the distance is minimum at $t = 18$ and when the relative velocity is negative, the risk is found to be the maximum.

This risk factor computed from the state variables of the target vehicles will be employed in different ways to cater for the different lane change scenarios. The next few sections

FIGURE 6.10: Illustration of risk factor on a sample image sequence.

will illustrate the use of risk factor in aiding the driver to take a decision during lane change.

6.5 Lane Change Decision Aid Process

In this section, the proposed risk function is used to assess the scene around the host-vehicle and aid the driver in taking a decision to change lane. The lane change risk function is applied to different RoIs and the overall risk factor is determined that gives the risk involved in changing lane in a given road scenario with corresponding state parameters. We first propose a generic method to determine the risk factor for a given road scenario using the state parameters of different RoIs. This method is then applied to different lane change scenarios.

Fig. 6.11 gives the basic steps in determining the risk factor a given lane change scenario. Given an input image, the lanes or RoIs are determined and the vehicles in these RoIs are detected and tracked in each frame. The state parameters of each RoI from front and rear cameras, i.e. $\Psi_{F_i}^{(t)}$'s and $\Psi_{R_i}^{(t)}$'s respectively, are computed. When a lane change intention is made by the driver in a particular lane direction, the corresponding RoIs are selected for further processing. For example, if the driver intends to change lane to the right, the host lane and the right lane become RoIs of interest for the rest of the

FIGURE 6.11: Flowchart for determining the risk factor for a lane change scenario.

risk assessment analysis. The risk factors for each RoI are determined. We denote the risk of an RoI L_i at time t as $\xi_{L_i}^{(t)}$. Therefore, for all the RoIs of interest, we compute the risk factors. These risk factors are used to compute the overall risk factor for the given lane change scenario. The process of lane change decision aid ends with either an alarm or an action depending on the risk that is estimate for the given scenario. It is to be noted that we have not yet considered the blind spot monitoring in this method. We are only considering scenarios when the driver intends to change lane and there are vehicles seen by the front and rear cameras in the lanes and there are *no* vehicles in the blind spot. Blind spot monitoring is addressed in the next section.

We now describe specific road/lane scenarios that occur when the driver intends to change lanes. We present the possible risk assessment functions for each of the scenarios and illustrate each scenario with an example for the test cases captured on real roads. The proposed risk factors can be changed based on the level of risk and kind of risk that

one wants to consider. For all cases below, we consider the driver intends to change lane to the right. The same methods can be applied for left lane change also.

Case 1 - One target vehicle TV : We consider the most simple case as shown in Fig. 6.12. This is the case when the driver intends to change to right lane but there is one TV . The risk in lane change in this case is because the host-vehicle needs to accelerate forward to change lane to the right. Therefore, while accelerating, the vehicle in the host or adjacent lane should be at a safe distance away from the ego vehicle.

Therefore, in this scenario, the RoIs of interest are F_0 and F_{+1} from front view, and R_{+1} from rear view (assuming there is no vehicle in the blind spot area). State parameters are defined for these RoIs of interest, i.e. we get $\Psi_{F_t}^{(0)}$, $\Psi_{F_t}^{(+1)}$ and $\Psi_{R_t}^{(+1)}$. As shown in Fig. 6.12, since there is only one TV on host lane, the p state variable in $\Psi_{F_t}^{(+1)}$ and $\Psi_{R_t}^{(+1)}$ is set to 0, i.e., $p_{F_0}^{(t)} = 1, p_{F_{+1}}^{(t)} = 0, p_{R_{+1}}^{(2)} = 0$. Therefore, we have the risk factor determined for F_0 only, i.e. $\xi_{R_0}^{(t)}$.

FIGURE 6.12: Illustration of road scene for Case 1.

The overall risk factor $\bar{\xi}^{(t)}$ for this case is therefore determined by the risk factor of the TV is the host lane. In other words,

$$\bar{\xi}^{(t)} = f(\xi_{R_0}^{(t)}) \quad (6.18)$$

This function $f(\cdot)$ in the above equation can be defined as any function that is dependent on the risk factor of the host lane target vehicle. We consider a constant gain factor $\tau_0 < 1$ to relate the two risk factors to illustrate the scenario, i.e.

$$\bar{\xi}^{(t)} = \tau_0(\xi_{R_0}^{(t)}) \quad (6.19)$$

In other words, if the host lane TV has a risk factor of $\xi_{F_0}^{(t)}$, the overall risk factor for lane change is only a fraction of it. This fraction is constant in our definition but it can be made a function of other parameters also like vehicle yaw angle etc. These additional inputs can be brought in to define τ_0 . Fig. 6.13 shows the risk factor computation for this scenario. Since we have only one TV , the risk factor is computed for host lane. The risk in changing lane in this scenario is computed for a safe distance risk factor set

to 0.5. The risk is plotted against t axis and it can be seen that if TV is closer to the host-vehicle and approaching it, the overall risk factor increases. The relative velocity of TV also affects $\bar{\xi}^{(t)}$.

FIGURE 6.13: Plot of $d_{F_0}^{(t)}$, $v_{F_0}^{(t)}$ and $\xi_{F_0}^{(t)}$ against time for an image sequence

This case can be extended to other scenarios also when one target vehicle is present in different RoIs, i.e. either in the front or in the rear.

Case 2 - Multiple target vehicles in host and right lanes: In this case, we consider the case of multiple vehicles in the road scene. Consider the illustration shown in Fig. 6.14 where there are three target vehicles in the three RoIs of interest around the host-vehicle. Each vehicle will have its state parameters given by $\Psi_{F_0}^{(t)}$, $\Psi_{F_{+1}}^{(t)}$ and $\Psi_{R_{+1}}^{(t)}$. These state variables are used to determine the risk factor for each RoI resulting in $\xi_{F_0}^{(t)}$, $\xi_{F_{+1}}^{(t)}$ and $\xi_{R_{+1}}^{(t)}$. The overall risk factor for this case is given by

$$\bar{\xi}^{(t)} = f(\xi_{F_0}^{(t)}, \xi_{F_{+1}}^{(t)}, \xi_{R_{+1}}^{(t)}) \quad (6.20)$$

In this case, we define $f(\cdot)$ as the maximum function, i.e. the overall risk factor is given by the maximum of the individual risk factors, i.e.,

$$\bar{\xi}^{(t)} = \max(\xi_{F_0}^{(t)}, \xi_{F_{+1}}^{(t)}, \xi_{R_{+1}}^{(t)}) \quad (6.21)$$

We consider this function because the risk of the overall system is high if any of the target vehicle is posing a high risk. However, this function can be defined as any combination of the three risk factors depending on the level of risk a user wants. For example, if we

want to give higher weight to the risk from rear approaching TV and lesser risk to the TV on host lane, the function can be defined in the following way,

$$\bar{\xi}^{(t)} = \max(w_{F_0}^{(t)}\xi_{F_0}^{(t)}, w_{F_{+1}}^{(t)}\xi_{F_{+1}}^{(t)}, w_{R_{+1}}^{(t)}\xi_{R_{+1}}^{(t)}) \quad (6.22)$$

where $w_{F_0}^{(t)}$, $w_{F_{+1}}^{(t)}$ and $w_{R_{+1}}^{(t)}$ are the three weight factors such that $w_{F_0}^{(t)} < w_{F_{+1}}^{(t)} < w_{R_{+1}}^{(t)}$.

FIGURE 6.14: Illustration of the road scene when host-vehicle EV intends to change lane to the right lane. Three target vehicles TV_1 , TV_2 and TV_3 are seen in the road scene.

Fig. 6.15 shows the risk factor plot across time frames for case 2. It can be seen that the overall risk factor $\bar{\xi}(t)$ increases if any of the three target vehicles has a high risk factor. It can be seen that from $t = 0$ to $t = 12$, the overall risk for this case is dependent on TV_1 , which is approaching towards the host vehicle from front in the host lane. From $t = 12$ to $t = 26$, the risk posed by TV_3 is higher because it is approaching the host vehicle from rear on the right lane. Therefore, the overall risk also set to the risk posed by TV_3 . It can be seen that TV_2 is not posing any risk in this scenario till the end at $t = 26$ to $t = 28$, which the relative distance between TV_2 and the host vehicle is reducing into the danger zone.

Therefore, it can be seen from Fig. 6.15 that the proposed framework is able to determine the risk posed by different vehicle movements on the host vehicle.

6.6 State Machine for Blind Spot Monitoring

In this section, we describe how the blind spots of the host-vehicle are monitored using the two camera setup. Let us consider the situation where the driver in host-vehicle intends to change to his right lane. In order to do this, the right lane R_{+1} need to be monitored. Given this scenario, we want to ensure that the driver is alerted about the presence of the target vehicle in the blind spot area when it enters the blind spot area. Schubert et al in [30] have briefly mentioned on how blind spot is monitored using the front and rear cameras by tracking the movement of the approaching vehicles but no systematic method was proposed.

FIGURE 6.15: Plots of $d^{(t)}, v^{(t)}, \bar{\xi}^{(t)}$ for TV_1, TV_2 and TV_3 in case 2. *, $\triangle$, $\circ$ are the plots for TV_1, TV_2 and TV_3 respectively. The top plot gives the relative proximities of the target vehicles. The middle plot gives the risk posed by each TV . The overall risk factor $\bar{\xi}^{(t)}$ is given in the bottom-most plot.

We propose a state machine to monitor the blind spot area using the proposed state variables obtained from the front and rear RoIs. It is also assumed that there are no side exit or entry roads during the time when the target vehicle enters the blind spot area. In other words, if a target vehicle is entering a blind spot area, it will remain in the blind spot area or in the adjacent lane and it does not exit the lane through side exit roads (and vice versa for side entry roads).

FIGURE 6.16: Proposed state machine for blind spot monitoring using state parameters

Fig. 6.16 shows the proposed state machine for blind spot monitoring. In order to keep

the notations simple, the vehicle presence state parameter p and the risk factor ξ of the front and rear RoIs on the side to which lane change is intended are represented by p_F and p_R , and ξ_F and ξ_R respectively. The safety risk factor is given by ξ_s . In Fig. 6.16, each state is represented by the bubble with the state id and the value of the blind spot monitoring parameter is also indicated in the state bubble given by γ . Therefore, all states except the blind-spot full state S_B^P have $\gamma = 0$. The edges show the flow from one state to the next and the trigger conditions are given by the state parameters. For ease of understanding we omit the time stamp (t) on the variables and states but it should be noted that all the variables are time stamped with (t).

We make use of the risk factor and vehicle presence state parameters to derive the state machine. The basic idea behind the functioning of the state machine is that if a vehicle is detected in the RoI, say rear RoI, it will have to be first detected in the safe rear RoI. If TV is approaching the host-vehicle, it will enter the danger zone of the rear RoI and only then it will enter the blind spot (we are not considering the cases where the target vehicle itself changes lane during the overtaking process). After the TV enters the blind spot, it has two options, either it passes the host-vehicle and enter the danger zone of front RoI and then goes into safe zone of front RoI, or it trails back into the rear RoI. In other words, the following can happen:

- **Overtaking from rear:** TV in Safe Rear RoI $\rightarrow TV$ in Danger Rear RoI $\rightarrow TV$ in Blind spot $\rightarrow TV$ in Danger Front RoI $\rightarrow TV$ in Safe Front RoI.
- **Entering blind spot and trailing back:** TV in Safe Rear RoI $\rightarrow TV$ in Danger Rear RoI $\rightarrow TV$ in Blind spot $\rightarrow TV$ in Danger Rear RoI $\rightarrow TV$ in Safe Rear RoI.
- **Front vehicle trailing back through blind spot:** TV in Safe Front RoI $\rightarrow TV$ in Danger Front RoI $\rightarrow TV$ in Blind spot $\rightarrow TV$ in Danger Rear RoI $\rightarrow TV$ in Safe Rear RoI.

Based on the above movements, we define three states for the front and rear RoIs, and one state for the blind spot region. The following are the definitions of the states:

State	Description	(6.23)
S_F^E	Front RoI empty	
S_F^S	Front RoI full but TV at safe distance	
S_F^D	Front RoI full but TV at unsafe distance	
S_R^E	Rear RoI empty	
S_R^S	Rear RoI full but TV at safe distance	
S_R^D	Rear RoI full but TV at unsafe distance	
S_B^P	Blind-spot full	

TABLE 6.1: Blind spot state machine example

Time	Camera	$d^{(t)}$	$\xi^{(t)}$	State	γ	Time	Camera	$d^{(t)}$	$\xi^{(t)}$	State	γ
1	Rear	56	0	S_B^E	0	37	Rear	0	0	S_B^P	1
3	Rear	56	0.396	S_B^S	0	39	Front/Rear	0	0	S_B^P	1
5	Rear	50	0.529	S_B^S	0	41	Front/Rear	0	0	S_B^P	1
7	Rear	45	0.589	S_B^S	0	43	Front	1	0	S_B^P	1
9	Rear	43	0.595	S_B^S	0	45	Front	3	0.997	S_B^D	0
11	Rear	40	0.645	S_B^S	0	47	Front	4	0.995	S_B^D	0
13	Rear	37	0.687	S_B^S	0	49	Front	7	0.984	S_B^D	0
15	Rear	30	0.799	S_B^D	0	51	Front	10	0.969	S_B^D	0
17	Rear	25	0.849	S_B^D	0	53	Front	11	0.964	S_B^D	0
19	Rear	21	0.889	S_B^D	0	55	Front	15	0.929	S_B^D	0
21	Rear	19	0.904	S_B^D	0	57	Front	19	0.888	S_B^D	0
23	Rear	15	0.942	S_B^D	0	59	Front	20	0.886	S_B^D	0
25	Rear	11	0.968	S_B^D	0	61	Front	25	0.810	S_B^D	0
27	Rear	9	0.978	S_B^D	0	63	Front	28	0.778	S_B^D	0
29	Rear	6	0.990	S_B^D	0	65	Front	32	0.714	S_B^D	0
31	Rear	5	0.993	S_B^D	0	67	Front	35	0.676	S_B^S	0
33	Rear	2	0.999	S_B^D	0	69	Front	39	0.607	S_B^S	0
35	Rear	1	1.000	S_B^D	0	71	Front	41	0.593	S_B^S	0

It can be seen from Fig. 6.16 that the state machine handles all the possible combinations of vehicle movements from the front and rear RoIs into one another through the blind-spot. Table 6.1 shows an example for the blind spot monitoring using relative distances extracted from an image sequence. The relative distances and the computed risk factors are used to determine the states in the state machine. These are also shown in Fig. 6.17. It can be seen that as the vehicle approaches from rear and goes forward, the blind spot state parameter γ is set to 1 between $t = 35$ to $t = 41$.

FIGURE 6.17: Plots of $d^{(t)}$, $\xi^{(t)}$ and $\gamma^{(t)}$ versus time frame.

Note on Extending the Proposed State Machine: A few assumptions were made in designing the proposed state machine for blind spot detection. Firstly, it is assumed that there are not side exit or entry roads, which implies that a target vehicle that is detected cannot suddenly disappear into an side exit road (and vice versa for side entry roads). The second assumption is that the target vehicle is not changing lanes during the process of overtaking. We have also assumed that we are only monitoring the immediate neighbor lanes for blind spot monitoring and the vehicles in these neighbor lanes only are being monitored for entry into or exiting from the blind spot areas. The movement of vehicles from far neighbor lanes into blind spots is not being monitored in the proposed system.

Having said that, the state machine can be extended further to cater to the above mentioned assumptions by considering the far neighbor lanes also. In this thesis, we have presented a simplified and systematic approach to monitor the blind spot given the above assumptions. But the same be extended with additional states to monitor the far neighbor lanes. Additional state parameters can be added to cater for the side entry and exit roads from which vehicles may enter or leave from the neighbor lanes. Using information from additional sources like GPS, as complementary information, can also be incorporated for making the blind spot detection more robust to varying traffic scenarios. This has been proposed as an item for future research in the next Chapter.

6.7 Computation Framework for Proposed Lane Change Decision Aid System

In this section, we propose a computation architecture for the proposed LCDAS as shown in Fig. 6.18. The proposed architecture comprises of two processors, which we call as front and rear view processors, which are fed by the video streams from front and rear facing cameras respectively. Both the processors perform the same functionality, which is to detect the RoIs (lanes) and then detect the vehicles in lanes. This is achieved by the GMH-GAH-HT parallel engine in each processor. This engine employs the proposed additive HT (AHT) and hierarchical AHT (HAHT) to determine the lanes in near-view and far-view of the input images. These lane positions are then sent into the region processors, as shown in Fig. 6.18. The region processors in front and rear view detect the vehicles in the regions (lanes). They also determine the RoI state parameters which are then used to determine the risk posed by each vehicle in the lane. The RoI state parameters, with the risks of vehicles in each lane, from the front and rear view processors are then sent into inter camera information processor (shown in Fig. 6.18). This block computes the overall risk in the scene based on the driver intention to change lane and

the surround scene around the host vehicle. Blind spot monitoring is also performed in this block. This block notifies the driver whether it is safe to change lane based on the vehicle positions and states around the host vehicle.

FIGURE 6.18: Computation Architecture for Proposed Lane Change Decision Aid System

6.8 Summary

In this chapter, all the different aspects of lane change decision aid system were brought together to propose a framework for an LCDAS. The relative proximities of the vehicles in different lanes were used to propose state parameters for each target vehicle. The proposed method for risk assessment showed that the risk posed by a target vehicle to the host vehicle is dependent on the actual stopping distance based on the relative velocity and the velocity of the host vehicle. An overall risk factor is also computed based on the surround scene around the host vehicle. In addition, we also proposed a state machine to monitor the blind spot before a lane change maneuver takes place. Finally a computation framework is proposed for the overall LCDAS. In the next chapter, critical conclusions are drawn based on the different techniques and their effectiveness in solving

the lane change assistance problem using vision sensors. Future research directions are also proposed in the next chapter.

CHAPTER 7

Conclusions & Future Work

7.1 Conclusions

This thesis addresses the development of key functional blocks of a vision-based LCDAS, namely, multiple lane extraction, vehicle detection, vehicle proximity estimation, situation assessment of the road scene and lane change decision generation.

The proposed GMH-GAH-HT engine for detecting host and neighbor lanes is shown to overcome the challenges posed by varying prominence of edges of these lanes in near-view. Controlled release of edge pixels based on GMH distribution provides for an automatic assignment of thresholds in the gradient map based on similarity of the edge prominence. GAH was examined after each release of edge pixels to sieve out most non-lane marking features and to promote most likely lane-edge candidates that can be subjected to HT processing. The proposed iterative process to eliminate the unlikely candidates has led to a notable reduction in the HT computation complexity. The proposed engine is shown to reliably detect neighbor lanes in the near-view even though they are notably fainter than host lanes. It was observed that the mis-detections occur only when lane features are so faint that they are almost indistinguishable from the background information.

Lanes in near-view were used as reference to detect the faint lanes in far-view more accurately and efficiently. This was facilitated by the re-deployment of GMH-GAH-HT engine to selectively probe the far-view regions by estimating the relative positions of far-view lanes using near-view lanes. A novel AHT technique was also introduced to parallelize compute intensive HT algorithm. This has led to a notable speed up as it resulted in exploiting the inherent parallelisms of the HT algorithm and also eliminated

all multiplications and trigonometric operations of conventional HT computations. It is noteworthy that AHT uses simple constant adders that can be reconfigured based on the block size of the image.

The hierarchical AHT (known as HAHT) method was proposed in the thesis to tackle the efficient computation of the global Hough spaces from the local Hough spaces. This makes it possible to detect long lane-edges in the far-view by combining associated blocks into effectively one Hough space. This eliminates the inter-block association, thereby leading to efficient identification of long lanes in the far view. The proposed approach can be easily adapted to focus only the region of interest by tweaking the block size and level of hierarchy. It is shown that HAHT together with GMH-GAH-HT is well suited for extracting lengthy multiple lanes in far-view. The proposed AHT was also deployed to examine curved lanes when straight lanes are absent by approximating curved lanes as a set of linear segments, which are readily detected either by AHT and HAHT. It was also observed that the proposed approach for detecting curved lanes is effective when the road curvature is evident between 20-30 meters from the ego-vehicle.

The proposed vehicle detection method employs multiple cues, which include under-vehicle shadows and multiple edge symmetry of the vehicle. Only vehicles nearest to the host vehicle in the extracted RoIs (lanes) are detected as they contribute to the maximum risk associated with lane change decisions. The under-vehicle shadow is detected by setting a binarization threshold that was shown to be adaptive to local illumination changes around the target vehicle. The lane RoI was systematically divided into smaller sub-images to adaptively determine the local illumination conditions by referring to the intensities immediately adjacent to lane markings as a reference. This approach is superior to existing under-vehicle shadow detection methods that not only use a global binarization threshold but are also not capable of adapting to local illumination changes.

In order to further affirm the presence of vehicle, multiple edge symmetry cues were proposed. Unlike existing edge symmetry methods, which use only the horizontal and vertical edges, slant-edge symmetry was introduced in the proposed method to enhance the robustness of the vehicle detection process. Moreover, the GMH-GAH-HT operation was targeted selectively to determine these edges to further minimize the computational complexity. These improvements led to significant reduction in the overall computation cost when compared with the existing approaches for vehicle detection.

The proposed vehicle detection method was also further extended to cater for more unpredictable features of motorcycles in lanes. The method proposed to detect the tyre of the motorcycle was based on the strategies adopted for under-vehicle shadow detection. It was shown that scrutinizing the region immediately surrounding the tyre by examining the background road intensities and persistent edge features of motorcycles

in the current and subsequent frames, provides for additional information to ascertain the presence of a motorcycle. This method however, needs further investigation to improve its robustness in complex scenarios.

The location of the detected vehicle in the lane and the lane width information in its vicinity are then deployed to estimate the relative proximity of the target vehicle using 2D images. The proposed techniques for vehicle detection and its proximity have resulted in a low-complexity method than those presented in the literature.

In order to develop a functional LCDAS, a framework for situation assessment around the host vehicle is proposed. This relies on the proposed two-camera configuration using front and rear cameras, which capture the image scenes from the front and rear of the host vehicle. Unlike most existing LCDAS that cater mainly to blind spot monitoring only, the proposed camera configuration and situation assessment framework is also capable of determining the risk associated with lane change using a holistic model by stitching the information from front and rear views of the host vehicle to get a complete view of the road scene.

The relative positions of the target vehicles in front and rear are then used to determine whether or not a lane change is safe at a given instant of time. In particular, the proposed situation assessment method demonstrates the use of relative proximities and velocities of the target vehicles with respect to the host vehicle, in deriving a Gaussian-based risk function. The proposed risk function is shown to accommodate the relative safe stopping distance between the host and target vehicles in determining the risk associated with a lane change maneuver. Moreover, the proposed framework for LCDAS is shown to cater to different traffic scenarios, that include single and multiple target vehicles approaching the host vehicle from both front and rear directions. Unlike most existing methods that use a constant metric to indicate the risk in changing lanes at a given time instant, the proposed framework is shown to be adaptive to the speed of ego and target vehicles. This approach is very promising as there is limited information in the literature to address this part of the LCDAS in a quantitative manner.

In addition to dynamic assessment of the risk posed by the vehicles in front and rear fields of view, a well-defined state machine is proposed to determine the presence of vehicles in blind spots, thereby eliminating the need for additional sensors to monitor the blind spots. The state machine is enabled using the risk factors obtained by continuously monitoring the relative positions of vehicles, and investigations using real video sequence show that it can determine the presence of the vehicles in blind spot more deterministically.

Finally, the compute-efficient techniques proposed in this thesis to implement the core computation modules of a vision based LCDAS can be efficiently integrated to demonstrate that the proposed algorithms are a significant step towards real-time feasibility of LCDAS.

7.2 Future Work

The following research directions are recommended for future work:

- This thesis addresses efficient detection of host lanes and neighbor lanes. Detecting lanes that are beyond the immediate neighbor lanes (referred as far-neighbor lanes) may strengthen the overall effectiveness of LCDAS. Although the proposed lane detection techniques for host and neighbor lanes are sensitive to the far-neighbor lanes, the presence of features like road boundaries and pavements tend to result in false detections. Knowing the far-neighbor lanes more conclusively can give additional information for the lane change decision aid process by accounting for possible knock-on effects that will influence the lane change maneuvers of the ego vehicle. Detecting the far-neighbor markings under such scenarios more deterministically is a potential area for future work.
- Global Positioning System (GPS) based navigation systems have become an integral part of most cars, both luxury and consumer cars. Nowadays, they are equipped with extensive and accurate map data, which has information like the number of lanes, approaching junctions and curves etc. Access to such local geographical information can enable the vision-based LCDAS to make informed decisions more deterministically and proactively. For example, if it is known that the vehicle is in a single lane carriageway, the vision-based LCDAS need not search for neighboring lanes. Similarly, having the information of an approaching curve from the GPS can help the LCDAS look out for curved lane. Therefore, developing a vision-based LCDAS, which is integrated with inputs from the GPS navigation systems, can result in a robust, low-cost and highly effective solution for lane change assistance.
- In this thesis, significant contributions were made in developing low complexity techniques for the robust detection of motorcycles. However, more work is required to realize a more robust method, particularly when the salient features are less prominent than cars and other large vehicles. Particular attention must be paid when motorcycles are in far-view as the chances of mis-detection are higher in the presence of complex background scenarios and reduced perspective. Detecting

motorcycles deterministically in both near and far-views of the host vehicle must also take into account the tradeoff between the computational complexity and detection accuracy.

- Techniques for detecting curved lanes using AHT, HAHT and GAH were proposed in this thesis, which were shown to be applicable for highway scenarios, where the roads do not exhibit sharp curvatures. However, for lane change assistance in urban roads inside city limits, the curvatures can be sharper and hence the proposed techniques must consider working with smaller block sizes than those deployed in this work. Detecting such sharp curvatures accurately using linear approximation techniques in a cost-effective manner is another research direction. Determining the curvature of the road based on GPS data and vehicle state parameters is worthy of further investigation.
- In the research presented in this thesis, two kinds of contributions were made: (a) robust algorithms, (b) computationally efficient architectures. A recommended future work package is the design and development of a unified architecture that integrates the different computing modules on an embedded computing platform. Different implementation platforms could be explored like softcore processors, FPGAs etc. to realize a real-time LCDAS. Potential for distributed and parallel computations must be investigated to improve the overall responsiveness of cost-effective realization.
- The techniques proposed in this thesis depend on monocular cameras for scene sensing. With continuous advances in VLSI technologies, although monocular cameras are getting cheap and more pervasive than other sensing technologies, it is worth investigating other types of sensing technologies. Examples of such technologies include infrared sensors, range cameras etc. which not only capture the visual scene but also give additional scene information like depth, temperature etc under varying lighting conditions. Moreover, extreme weather scenarios like heavy rain, fog and snow, lane markings are not visible even to the human eye. Considering that cameras are cheaper and more pervasive than other sensors, cameras can be the primary sensor technology under normal weather conditions. They can be supported by sensors like radars, when the visibility is poor or obstructed. In such scenarios, a multi-sensor based approach, involving other sensors like radars, infrared, GPS etc. can be effective. Developing methods that combine vision-based techniques with cues from other sensors is a potential item for future work. A combination of these different sensor technologies by intelligently switching between the sensors based on the information needed, is a future research direction for developing a robust and cost-effective LCDAS. The tradeoff between the robustness

and cost of deploying such technologies needs to be thoroughly investigated to evaluate the viability of such systems in the mass market.

- In this thesis, a novel framework for risk assessment and lane change decision aid process was proposed. The main parameters that were used in the proposed framework were the relative distances, velocities and the safe stopping distance measures of the target vehicles from the host vehicle. Enhancing this framework further by incorporating human behavior makes it a formidable research work package for future work. For example, the driving style of the driver of host vehicle is an important factor in determining the risk involved in a lane change maneuver, i.e. a risk could be higher in the case of an aged driver with slower reflexes or a young inexperienced driver, as compared to one who is more experienced. Customized alert levels can be introduced to accommodate such driver-specific parameters. This can lead to a more personalized LCDAS that can further improve safety during high speed maneuvers.
- Given that the core modules discussed in this thesis can be tailored for lane departure warning (LDW) systems, a possible future work could be to develop a vision based LDW system using the proposed modules.

Bibliography

- [1] J. Golias, G. Yannis, and C. Antoniou, "Classification of driver-assistance systems according to their impact on road safety and traffic efficiency," *Transport Reviews*, vol. 22, pp. 179–196, 2002. [Online]. Available: http://users.ntua.gr/igolias/papers/paper_37.pdf
- [2] J. McCall and M. Trivedi, "Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 1, pp. 20–37, march 2006.
- [3] "Research and markets; analysis of north american market for advanced driver assistance systems - acc, ldw, bsd, night vision and park assist systems," *Journal of Transportation*, p. 129, Apr 24 2010. [Online]. Available: <http://search.proquest.com/docview/208637028?accountid=12665>
- [4] "Research and markets: The market for advanced das (adas) advanced driver assistance systems is estimated to reach 143 million euros by 2015," *Business Wire*, Aug 12 2009. [Online]. Available: <http://search.proquest.com/docview/443971451?accountid=12665>
- [5] "Research and markets: Analysis of the european and north american passenger car market for camera-based advanced driver assistance systems," *Business Wire*, Jun 13 2012. [Online]. Available: <http://search.proquest.com/docview/1020109732?accountid=12665>
- [6] M. Beecham, "Global market review of driver assistance systems - forecasts to 2017: 2010 edition: Chapter 2 the market," Aroq Limited, Tech. Rep., 2010. [Online]. Available: <http://search.proquest.com/docview/213131187?accountid=12665>

- [7] A. Lindgren, F. Chen, P. W. Jordan, and H. Zhang, "Requirements for the design of advanced driver assistance systems - the differences between swedish and chinese drivers," *International Journal of Design*, vol. 2, no. 2, Aug 31 2008.
- [8] B. Pattengale, "What you should know about advanced driver assistance systems," *Motor*, vol. 214, no. 6, pp. 34–36,38,40,42,44–45, 2010.
- [9] N. Ford, "Future evolution of advanced driver assistance systems in europe: Frost & sullivan," 2010.
- [10] A. Vahidi and A. Eskandarian, "Research advances in intelligent collision avoidance and adaptive cruise control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 4, no. 3, pp. 143 – 153, sept. 2003.
- [11] Z. Bareket, P. Fancher, H. Peng, K. Lee, and C. Assaf, "Methodology for assessing adaptive cruise control behavior," *IEEE Transactions on Intelligent Transportation Systems*, vol. 4, no. 3, pp. 123 – 131, sept. 2003.
- [12] J. Clanton, D. Bevely, and A. Hodel, "A low-cost solution for an integrated multi-sensor lane departure warning system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 1, pp. 47 –59, march 2009.
- [13] C. Chen and Y. Chen, "Real-time approaching vehicle detection in blind-spot area," in *2009 12th International IEEE Conference on Intelligent Transportation Systems*, oct. 2009, pp. 1 –6.
- [14] B.-F. Lin, Y.-M. Chan, L.-C. Fu, P.-Y. Hsiao, L.-A. Chuang, and S.-S. Huang, "Incorporating appearance and edge features for vehicle detection in the blind-spot area," in *2010 13th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, sept. 2010, pp. 869 –874.
- [15] A. Eskandarian, "Fundamentals of driver assistance," *Handbook of Intelligent Vehicles*, pp. 491–535, 2012.
- [16] O. Gietelink, J. Ploeg, B. D. Schutter, , and M. Verhaegen, "Development of advanced driver assistance systems with vehicle hardware-in-the-loop simulations," *Vehicle System Dynamics*, vol. 44, no. 7, p. 569590, 2006.
- [17] "Google driverless car," http://en.wikipedia.org/wiki/Google_driverless_car, 2012.
- [18] M. Beecham, "Global market review of driver assistance systems - forecasts to 2014: 2008 edition: The market," Aroq Limited, Tech. Rep., 2008. [Online]. Available: <http://search.proquest.com/docview/212970398?accountid=12665>

- [19] W. Mehr, “Advanced Driver Assistance Systems (ADAS): Products for Commercial Vehicles,” <http://www.unece.org/fileadmin/DAM/trans/doc/2009/wp29grrf/AEBS-LDW-01-08e.pdf>, 2009.
- [20] “Advanced Driver Assistance Systems (ADAS),” http://www.conti-online.com/generator/www/de/en/continental/automotive/themes/commercial_vehicles/safety/adas/index_en.html, 2009.
- [21] “BMW Official Site: Driver Assistance Systems,” http://www.bmw.com/com/en/newvehicles/7series/sedan_active_hybrid/2012/showroom/safety/assistance.html#t=1, 2012.
- [22] J. Clanton, D. Bevly, and A. Hodel, “A low-cost solution for an integrated multi-sensor lane departure warning system,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 1, pp. 47–59, march 2009.
- [23] D. Cualain, C. Hughes, M. Glavin, and E. Jones, “Automotive standards-grade lane departure warning system,” *IET Transactions on Intelligent Transport Systems*, vol. 6, no. 1, pp. 44–57, march 2012.
- [24] P.-Y. Hsiao, C.-W. Yeh, S.-S. Huang, and L.-C. Fu, “A portable vision-based real-time lane departure warning system: Day and night,” *IEEE Transactions on Vehicular Technology*, vol. 58, no. 4, pp. 2089–2094, may 2009.
- [25] X.-B. Cao, H. Qiao, and J. Keane, “A low-cost pedestrian-detection system with a single optical camera,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, no. 1, pp. 58–67, march 2008.
- [26] P. Kelly, N. O’Connor, and A. Smeaton, “A framework for evaluating stereo-based pedestrian detection techniques,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 8, pp. 1163–1167, aug. 2008.
- [27] L. Bi, O. Tsimhoni, and Y. Liu, “Using image-based metrics to model pedestrian detection performance with night-vision systems,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 1, pp. 155–164, march 2009.
- [28] S. Habenicht, H. Winner, S. Bone, F. Sasse, and P. Korzenietz, “A maneuver-based lane change assistance system,” in *2011 IEEE Intelligent Vehicles Symposium (IV)*, june 2011, pp. 375–380.
- [29] J. Diaz Alonso, E. Ros Vidal, A. Rotter, and M. Muhlenberg, “Lane-change decision aid system based on motion-driven vehicle tracking,” *IEEE Transactions on Vehicular Technology*, vol. 57, no. 5, pp. 2736–2746, sept. 2008.

- [30] R. Schubert, K. Schulze, and G. Wanielik, "Situation assessment for automatic lane-change maneuvers," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 3, pp. 607–616, sept. 2010.
- [31] B. Morris, A. Doshi, and M. Trivedi, "Lane change intent prediction for driver assistance: On-road design and evaluation," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, june 2011, pp. 895–901.
- [32] G. Xu, L. Liu, Z. Song, and Y. Ou, "Generating lane-change trajectories using the dynamic model of driving behavior," in *2011 IEEE International Conference on Information and Automation (ICIA)*, june 2011, pp. 464–469.
- [33] "Top 25 Causes of Car Accidents," <http://seriousaccidents.com/legal-advice/top-causes-of-car-accidents/>, 2012.
- [34] M. Tideman, M. van der Voort, B. van Arem, and F. Tillema, "A review of lateral driver support systems," in *2007 IEEE Conference on Intelligent Transportation Systems Conference*, 30 2007-oct. 3 2007, pp. 992–999.
- [35] J. D. Chovan, L. Tijerina, G. Alexander, and D. Hendricks, "Examination of lane change crashes and potential IVHS counter measures," *Washington DC, USA: NHTSA*, 1994.
- [36] S. S. Arne Bartels, Marc-Michael Meinecke, "Lane change assistance," *Handbook of Intelligent Vehicles*, pp. 729–757, 2012.
- [37] J. Miura, M. Itoh, and Y. Shirai, "Toward vision-based intelligent navigator: its concept and prototype," *IEEE Transactions on Intelligent Transportation Systems*, vol. 3, no. 2, pp. 136–146, June 2004.
- [38] B.-F. Lin, Y.-M. Chan, L.-C. Fu, P.-Y. Hsiao, L.-A. Chuang, S.-S. Huang, and M.-F. Lo, "Integrating appearance and edge features for sedan vehicle detection in the blind-spot area," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 2, pp. 737–747, june 2012.
- [39] Z. Kim, "Robust lane detection and tracking in challenging scenarios," *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, no. 1, pp. 16–26, march 2008.
- [40] H.-Y. Cheng, C.-C. Yu, C.-C. Tseng, K.-C. Fan, J.-N. Hwang, and B.-S. Jeng, "Environment classification and hierarchical lane detection for structured and unstructured roads," *IET Computer Vision*, vol. 4, no. 1, pp. 37–49, march 2010.

- [41] C. Adam, R. Schubert, N. Mattern, and G. Wanielik, "Probabilistic road estimation and lane association using radar detections," in *2011 Proceedings of the 14th International Conference on Information Fusion (FUSION)*, july 2011, pp. 1 –8.
- [42] D. Felguera-Martin, J.-T. Gonzalez-Partida, P. Almorox-Gonzalez, Burgos-Garci, and M., "Vehicular traffic surveillance and road lane detection using radar interferometry," *IEEE Transactions on Vehicular Technology*, vol. 61, no. 3, pp. 959 –970, march 2012.
- [43] W. Wijesoma, K. Kodagoda, and A. Balasuriya, "Road-boundary detection and tracking using ladar sensing," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 3, pp. 456 – 464, june 2004.
- [44] B. Ma, S. Lakshmanan, and A. Hero, "A robust bayesian multisensor fusion algorithm for joint lane and pavement boundary detection," in *Proceedings of 2001 International Conference on Image Processing*, vol. 1, 2001, pp. 762 –765 vol.1.
- [45] B. Ma, S. Lakshmanan, and A. Hero, "Road and lane edge detection with multi-sensor fusion methods," in *Proceedings of 1999 International Conference on Image Processing*, vol. 2, 1999, pp. 686 –690 vol.2.
- [46] W. A. Bauson, "Integrated Radar-Camera Sensors: The Next Generation of Sensor Fusion," www.sae.org/events/gim/presentations/2010/williambauson.pdf, 2010.
- [47] "Advanced Driver Assistance Systems (ADAS) based on radar, lidar and camera technology," <http://pi.physik.uni-bonn.de/teilchenseminar/teilchenseminar08/Glander02.pdf>, 2008.
- [48] V. Kastrinaki, M. Zervakis, and K. Kalaitzakis, "A survey of video processing techniques for traffic applications," *Image and Vision Computing*, vol. 21, pp. 359 – 381, 2003.
- [49] A. H. S. Lai and N. H. C. Yung, "Lane detection by orientation and length discrimination," *IEEE Transactions on Systems, Man and Cybernetics, Part B*, vol. 30, no. 4, pp. 539 – 548, August 2000.
- [50] H.-Y. Cheng, B.-S. Jeng, P.-T. Tseng, and K.-C. Fan, "Lane detection with moving vehicles in the traffic scenes," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 4, pp. 571 –582, dec. 2006.
- [51] Y. U. Yim and S.-Y. Oh, "Three-feature based automatic lane detection algorithm (tfalda) for autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 4, no. 4, pp. 219 – 225, dec. 2003.

- [52] C.-F. Wu, C.-J. Lin, and C.-Y. Lee, "Applying a functional neurofuzzy network to real-time lane detection and front-vehicle distance measurement," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 42, no. 4, pp. 577–589, july 2012.
- [53] A. Borkar, M. Hayes, and M. Smith, "A novel lane detection system with efficient ground truth generation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 1, pp. 365–374, march 2012.
- [54] M. Bertozzi and A. Broggi, "Gold: a parallel real-time stereo vision system for generic obstacle and lane detection," *IEEE Transactions on Image Processing*, vol. 7, no. 1, pp. 62–81, jan 1998.
- [55] R. Satzoda, S. Sathyanarayana, T. Srikanthan, and S. Sathyanarayana, "Hierarchical additive hough transform for lane detection," *IEEE Embedded Systems Letters*, vol. 2, no. 2, pp. 23–26, june 2010.
- [56] P. Jeong and S. Nedevschi, "Efficient and robust classification method using combined feature vector for lane detection," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 4, pp. 528–537, april 2005.
- [57] D. Felguera-Martin, J.-T. Gonzalez-Partida, P. Almorox-Gonzalez, and M. Burgos-Garci anda, "Vehicular traffic surveillance and road lane detection using radar interferometry," *IEEE Transactions on Vehicular Technology*, vol. 61, no. 3, pp. 959–970, march 2012.
- [58] R. Gopalan, T. Hong, M. Shneier, and R. Chellappa, "A learning approach towards detection and tracking of lane markings," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 3, pp. 1088–1098, sept. 2012.
- [59] G. Mastorakis and E. Davies, "Improved line detection algorithm for locating road lane markings," *Electronics Letters*, vol. 47, no. 3, pp. 183–184, 3 2011.
- [60] R. Danescu and S. Nedevschi, "Probabilistic lane tracking in difficult road scenarios using stereovision," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 2, pp. 272–282, june 2009.
- [61] L.-W. Tsai, J.-W. Hsieh, C.-H. Chuang, and K.-C. Fan, "Lane detection using directional random walks," in *2008 IEEE Intelligent Vehicles Symposium*, june 2008, pp. 303–306.
- [62] S. Zhou, Y. Jiang, J. Xi, J. Gong, G. Xiong, and H. Chen, "A novel lane detection based on geometrical model and gabor filter," in *2010 IEEE Intelligent Vehicles Symposium (IV)*, june 2010, pp. 59–64.

- [63] A. Saudi, J. Teo, M. H. A. Hijazi, and J. Sulaiman, "Fast lane detection with randomized hough transform," in *2008 International Symposium on Information Technology*, vol. 4, aug. 2008, pp. 1–5.
- [64] S. Lee, H. Son, and K. Min, "Implementation of lane detection system using optimized hough transform circuit," in *2010 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, dec. 2010, pp. 406–409.
- [65] J. G. Kuk, J. H. An, H. Ki, and N. I. Cho, "Fast lane detection and tracking based on hough transform with reduced memory requirement," in *2010 13th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, sept. 2010, pp. 1344–1349.
- [66] G. Liu, F. Wo andrgo andtter, and I. Markelic and, "Combining statistical hough transform and particle filter for robust lane detection and tracking," in *2010 IEEE Intelligent Vehicles Symposium (IV)*, june 2010, pp. 993–997.
- [67] Y. Jiang, F. Gao, and G. Xu, "Computer vision-based multiple-lane detection on straight road and in a curve," in *2010 International Conference on Image Analysis and Signal Processing (IASP)*, april 2010, pp. 114–117.
- [68] R. Gupta, W. Snyder, and W. Pitts, "Concurrent visual multiple lane detection for autonomous vehicles," in *2010 IEEE International Conference on Robotics and Automation (ICRA)*, may 2010, pp. 2416–2422.
- [69] Y. Jiang, F. Gao, and G. Xu, "Self-calibrated multiple-lane detection system," in *2010 IEEE/ION Conference on Position Location and Navigation Symposium (PLANS)*, may 2010, pp. 1052–1056.
- [70] M. Nieto, L. Salgado, F. Jaureguizar, and J. Arrospe, "Robust multiple lane road modeling based on perspective analysis," in *15th IEEE International Conference on Image Processing, 2008 (ICIP)*, oct. 2008, pp. 2396–2399.
- [71] C. Thorpe, M. Hebert, T. Kanade, and S. Shafer, "Vision and navigation for the carnegie-mellon navlab," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 3, pp. 362–373, may 1988.
- [72] S. D. Buluswar and B. A. Draper, "Color machine vision for autonomous vehicles," *Int. J. Eng. Appl. Artif. Intell.*, vol. 1, pp. 245–256, 1998.
- [73] J. Zhang and H.-H. Nagel, "Texture-based segmentation of road images," in *Proceedings of the Intelligent Vehicles Symposium*, oct. 1994, pp. 260–265.
- [74] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 2nd ed. Prentice Hall, 2002.

- [75] R. O. Duda and P. E. Hart, "Use of the Hough transform to detect lines and curves in pictures," *ACM Comm.*, vol. 15, no. 1, pp. 11 – 15, 1972.
- [76] B. F. Wu, W. H. Chen, C. W. Chang, C. J. Chen, and M. W. Chung, "A new vehicle detection with distance estimation for lane change warning systems," *Proceedings of IEEE International Intelligent Vehicles Symposium*, pp. 698 – 703, June 2007.
- [77] P. Viola and M. Jones, "Robust real-time object detection," *2nd Int. W. on Statistical and Computational Theories of Vision*, 2001.
- [78] Z. Sun, G. Bebis, and R. Miller, "On-road vehicle detection: A review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 5, pp. 694 – 711, May 2006.
- [79] D. Timmermann, H. Hahn, and B. J. Hosticka, "Hough Transform using CORDIC method," *Electronics Letters*, vol. 5, no. 3, pp. 205 – 206, February 1989.
- [80] F. Zhou and P. Kornerup, "A high speed Hough Transform using CORDIC," Tech. Rep. PP-1995-27, 1, 1995. [Online]. Available: citeseer.ist.psu.edu/zhou95high.html
- [81] J. D. Bruguera, N. Guil, T. Lang, J. Villalba, and E. L. Zapata, "CORDIC based parallel/pipelined architecture for the Hough transform," *Journal of VLSI Processing*, vol. 12, pp. 207 – 221, 1996.
- [82] D. D. S. Deng and H. ElGindy, "High-speed parameterisable Hough Transform using reconfigurable hardware," *Pan-Sydney Area Workshop on Visual Information Processing*, pp. 51 – 57, May 2001.
- [83] M.-Y. Chern and Y.-H. Lu, "Design and integration of parallel Hough-transform chips for high-speed line detection," *Proceedings of 11th International Conference on Parallel and Distributed Systems*, vol. 2, pp. 42 – 46, July 2005.
- [84] K. Mayasandra, S. Saheli, W. Wang, and H. M. Ladak, "A distributed arithmetic hardware architecture for real-time Hough Transform-based segmentation," *Canadian Journal on Electrical and Computer Engineering*, vol. 30, no. 4, pp. 201 – 205, Fall 2005.
- [85] M. S. Merry and J. W. Baker, "A constant time algorithm for computing the Hough Transform on a reconfigurable mesh," *Image and Vision Computing*, vol. 14, pp. 35 – 37, 1996.

- [86] F. O. Ozbek and M. D. Wagh, "A parallel Hough transform algorithm for nonuniform images," *Pattern Recognition Letters*, vol. 15, pp. 253 – 259, March 1994.
- [87] C. S. Kannan and H. Y. H. Chuang, "Fast Hough transform on a mesh connected processor array," *Information Processing Letters*, vol. 33, pp. 243 – 248, January 1990.
- [88] M. Atiquzzaman, "Pipelined implementation of the multiresolution Hough transform in a pyramid multiprocessor," *Pattern Recognition Letters*, vol. 15, pp. 841 – 851, September 1994.
- [89] H. F. Li, D. Rao, and R. Jayakumar, "Improvements and systolic implementation of the Hough transformation for straight line detection," *Pattern Recognition*, vol. 22, no. 6, pp. 697 – 706, 1989.
- [90] A. L. Fisher and P. T. Highnam, "Computing the Hough transform on a scan line array processor," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 2, no. 3, pp. 262 – 265, 1989.
- [91] L. Costa and M. B. Sandler, "A binary Hough transform and its efficient implementation in a systolic array architecture," *Pattern Recognition Letters*, vol. 10, pp. 329 – 334, November 1989.
- [92] S. Tagzout, K. Achour, and O. Djekoune, "Hough transform algorithm for fpga implementation," *Signal Processing*, vol. 81, pp. 1295 – 1301, 2001.
- [93] R. Cucchiara, G. Neri, and M. Piccardi, "A real-time hardware implementation of the Hough transform," *Journal of Systems Architectures*, vol. 45, pp. 31 – 45, 1998.
- [94] R. Andraka, "A survey of CORDIC algorithms for FPGA based computers," *Proceedings of 1998 ACM/SIGDA Sixth International Symposium on FPGA*, pp. 191 – 200, February 1998.
- [95] K. K. Parhi, *VLSI Digital Processing Systems: Design and Implementation*. Wiley-Interscience, 1999.
- [96] D. H. Ballard and C. M. Brown, *Computer Vision*. Prentice Hall, 1982.
- [97] F. O’Gorman and M. B. Clowes, "Finding picture edges through collinearity of feature points," *IEEE Transactions on Computers*, vol. C-25, no. 4, pp. 449 – 456, April 1976.

- [98] E. K. Jolly and M. Fleury, "Multi-sector algorithm for hardware acceleration of the general Hough transform," *Image and Vision Computing*, vol. 24, pp. 970 – 976, 2006.
- [99] X. Dai, A. Kummert, S. B. Park, and U. Iurgel, "Vehicle centroid estimation based on radar multiple detections," in *IEEE International Conference on Vehicular Electronics and Safety*, dec. 2007, pp. 1 –5.
- [100] J. Fang, H. Meng, H. Zhang, and X. Wang, "A low-cost vehicle detection and classification system based on unmodulated continuous-wave radar," in *2007 IEEE Intelligent Transportation Systems Conference*, 30 2007-oct. 3 2007, pp. 715 –720.
- [101] J. Gunnarsson, L. Svensson, L. Danielsson, and F. Bengtsson, "Tracking vehicles using radar detections," in *2007 IEEE Intelligent Vehicles Symposium*, june 2007, pp. 296 –302.
- [102] M. van Leeuwen and F. Groen, "Vehicle detection with a mobile camera: spotting midrange, distant, and passing cars," *IEEE Robotics Automation Magazine*, vol. 12, no. 1, pp. 37 – 43, march 2005.
- [103] H. Mallot, H. Bulthoff, J. Little, and S. Bohrer, "Inverse perspective mapping simplifies optical flow computation and obstacle detection," *Biological Cybernetics*, vol. 64, no. 3, pp. 177–185, 1991.
- [104] A. Kuehnle, "Symmetry-based recognition for vehicle rears," *Pattern Recognition Letters*, vol. 12, pp. 249–258, 1991.
- [105] W. von Seelen, C. Curio, J. Gayko, U. Handmann, and T. Kalinke, "Scene analysis and organization of behavior in driver assistance systems," *Proceedings of 2000 International Conference on Image Processing*, pp. 524–527, 2000.
- [106] J. Crisman and C. Thorpe, "Color vision for road following," *Proc. SPIE Conf. Mobile Robots*, pp. 246–249, 1988.
- [107] S. Buluswar and B. Draper, "Color machine vision for autonomous vehicles," vol. 1, no. 2, pp. 245–256, 1998.
- [108] H. Mori and N. Charkari, "Shadow and rhythm as sign patterns of obstacle detection," in *IEEE International Symposium on Industrial Electronics*, 1993, pp. 271 –277.
- [109] N. Matthews, P. An, D. Charnley, and C. Harris, "Vehicle detection and recognition in greyscale imagery," *Control Eng. Practice*, vol. 4, pp. 473–479, 1996.

- [110] N. D. C. Goerick and M. Werner, "Artificial neural networks in real-time car detection and tracking applications," *Pattern Recognition Letters*, vol. 17, pp. 335–343, 1996.
- [111] U. Handmann, T. Kalinke, C. Tzomakas, M. Werner, and W. Seelen, "An image processing system for driver assistance," *Image and Vision Computing*, vol. 18, no. 5, 2000.
- [112] P. Parodi and G. Piccioli, "A feature-based recognition scheme for traffic scenes," *Proc. IEEE Intelligent Vehicles Symp.*, pp. 229–234, 1995.
- [113] M. Betke, E. Haritaglu, and L. Davis, "Real-time multiple vehicle detection and tracking from a moving vehicle," *Machine Vision and Applications*, vol. 12, no. 2, 2000.
- [114] T. Kalinke, C. Tzomakas, and W. von Seelen, "A texture-based object detection and an adaptive model-based classification," pp. 143 – 148, 1998.
- [115] R. Haralick, B. Shanmugam, and I. Dinstein, "Texture features for image classification," *IEEE Trans. System, Man, and Cybernetics*, pp. 610–621, 1973.
- [116] T. Ito, K. Yamada, and K. Nishioka, "Understanding driving situations using a network model," in *Proceedings of the Intelligent Vehicles Symposium*, sep 1995, pp. 48 –53.
- [117] U. Regensburger and V. Graefe, "Visual recognition of obstacles on roads," *Intelligent Robots and Systems*, pp. 73–86, 1995.
- [118] V. Graefe and U. Regensburger, "A novel approach for the detection of vehicle on freeways by real-time vision," *Intelligent Vehicles*, 1996.
- [119] J. Wu and X. Zhang, "A pca classifier and its application in vehicle detection," 2001.
- [120] M. W. D. Noli and W. von Seelen, "Real-time vehicle tracking and classification," *Intelligent Vehicles*, pp. 101–106, 1995.
- [121] L. Huang and M. Barth, "Tightly-coupled lidar and computer vision integration for vehicle detection," in *2009 IEEE Intelligent Vehicles Symposium*, june 2009, pp. 604 –609.
- [122] M. Mahlich, R. Schweiger, W. Ritter, and K. Dietmayer, "Sensorfusion using spatio-temporal aligned video and lidar for improved vehicle detection," in *2006 IEEE Intelligent Vehicles Symposium*, 0-0 2006, pp. 424 –429.

- [123] C. Premebida, G. Monteiro, U. Nunes, and P. Peixoto, "A lidar and vision-based approach for pedestrian and vehicle detection and tracking," in *IEEE Intelligent Transportation Systems Conference*, 30 2007-oct. 3 2007, pp. 1044 –1049.
- [124] G. Lu and M. Tomizuka, "Lidar sensing for vehicle lateral guidance: Algorithm and experimental study," *IEEE/ASME Transactions on Mechatronics*, vol. 11, no. 6, pp. 653 –660, dec. 2006.
- [125] K. Takagi, K. Morikawa, T. Ogawa, and M. Saburi, "Road environment recognition using on-vehicle lidar," in *2006 IEEE Intelligent Vehicles Symposium*, 0-0 2006, pp. 120 –125.
- [126] H. Weigel, P. Lindner, and G. Wanielik, "Vehicle tracking with lane assignment by camera and lidar sensor fusion," in *2009 IEEE Intelligent Vehicles Symposium*, june 2009, pp. 513 –520.
- [127] F. Garcia, P. Cerri, A. Broggi, A. de la Escalera, and J. Armingol, "Data fusion for overtaking vehicle detection based on radar and optical flow," in *2012 IEEE Intelligent Vehicles Symposium (IV)*, june 2012, pp. 494 –499.
- [128] G. Alessandretti, A. Broggi, and P. Cerri, "Vehicle and guard rail detection using radar and vision data fusion," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 1, pp. 95 –105, march 2007.
- [129] X. Liu, Z. Sun, and H. He, "On-road vehicle detection fusing radar and vision," in *2011 IEEE International Conference on Vehicular Electronics and Safety*, july 2011, pp. 150 –154.
- [130] A. Broggi, P. Cerri, F. Oleari, and M. Paterlini, "Guard rail detection using radar and vision data fusion for vehicle detection algorithm improvement and speed-up," in *2005 IEEE Conference on Intelligent Transportation Systems*, sept. 2005, pp. 552 – 556.
- [131] B. Steux, C. Laugeau, L. Salesse, and D. Wautier, "Fade: a vehicle detection and tracking system featuring monocular color vision and radar data fusion," in *2002 IEEE Intelligent Vehicle Symposium*, vol. 2, june 2002, pp. 632 – 639 vol.2.
- [132] M. Herbert, "Active and passive range sensing for robotics," *Proc. of IEEE International Conference on Robotics and Automation*, vol. 1, pp. 102–110, 2000.
- [133] S. Park, T. Kim, S. Kang, and K. Heon, "A novel signal processing technique for vehicle detection radar," pp. 607–610, 2003.

- [134] C. Wang, C. Thorpe, and A. Suppe, "Ladar-based detection and tracking of moving objects from a ground vehicle at high speeds," *Proc. IEEE Intelligent Vehicles Symp.*, 2003.
- [135] J. Hancock, E. Hoffman, R. Sullivan, D. Ingimarson, D. Langer, and M. Hebert, "High-performance laser ranger scanner," 1997.
- [136] R. Chellappa, G. Qian, and Q. Zheng, *Proceeding of 2004 IEEE International Conference on Image Processing*, pp. 793–796.
- [137] R. Allan, "Semiconductors drive auto safety and control innovation," *Electronic Design*, pp. 17–20, 2008.
- [138] R. Schubert and G. Wanielik, "A unified bayesian approach for tracking and situation assessment," in *2010 IEEE Intelligent Vehicles Symposium (IV)*, june 2010, pp. 738 –745.
- [139] P. Boyraz, X. Yang, and J. H. L. Hansen, "Computer vision systems for context-aware active vehicle safety and driver assistance," *Digital Signal Processing for In-Vehicle Systems and Safety*, pp. 217–227, 2012.
- [140] J. Arrospe, L. Salgado, and J. Marinas, "Hog-like gradient-based descriptor for visual vehicle detection," in *2012 IEEE Intelligent Vehicles Symposium (IV)*, june 2012, pp. 223 –228.
- [141] S. S. Teoh and T. Brunl, "Symmetry-based monocular vehicle detection system," *Machine Vision and Applications*, vol. 23, no. 5, 2012.
- [142] E. Dickmanns, "The seeing passenger car 'vamos-p'," *Proc. of IEEE Intelligent Vehicles Symposium*, pp. 24–26, 1994.
- [143] B. Aytekin and E. Altug, "Increasing driving safety with a multiple vehicle detection and tracking system using ongoing vehicle shadow information," in *2010 IEEE International Conference on Systems Man and Cybernetics (SMC)*, oct. 2010, pp. 3650 –3656.
- [144] S. Kim, S.-Y. Oh, J. Kang, Y. Ryu, K. Kim, S.-C. Park, and K. Park, "Front and rear vehicle detection and tracking in the day and night times using vision and sonar sensor fusion," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, aug. 2005, pp. 2173 – 2178.
- [145] E. Y. Chung, H. C. Jung, E. Chang, and I. S. Lee, "Vision based for lane change decision aid system," in *The 1st International Forum on Strategic Technology*, oct. 2006, pp. 10 –13.

- [146] W. Liu, X. Wen, B. Duan, H. Yuan, and N. Wang, "Rear vehicle detection and tracking for lane change assist," in *2007 IEEE Intelligent Vehicles Symposium*, june 2007, pp. 252 –257.
- [147] T.-H. Chen, J.-L. Chen, C.-H. Chen, and C.-M. Chang, "Vehicle detection and counting by using headlight information in the dark environment," in *Third International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, vol. 2, nov. 2007, pp. 519 –522.
- [148] W. Zhang, Q. Wu, G. Wang, and X. You, "Tracking and pairing vehicle headlight in night scenes," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 1, pp. 140 –153, march 2012.
- [149] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, june 2005, pp. 886 –893.
- [150] I. Cabani, G. Toulminet, and A. Bensrhair, "Color-based detection of vehicle lights," in *2005 IEEE Intelligent Vehicles Symposium*, june 2005, pp. 278 – 283.
- [151] C. Tzomakas and W. von Seelen, "Vehicle detection in traffic scenes using shadows," IR-INI, Institut fur Nueroinformatik, Ruhr Universitat, Tech. Rep., 1998.
- [152] H. Y. Chang, C. H. Fu, and C. L. Huang, "Real-time vision-based preceding vehicle tracking and recognition," *Proceedings of IEEE International Intelligent Vehicles Symposium*, pp. 514 – 519, June 2005.
- [153] H. Cheng, N. Zheng, and C. Sun, "Boosted gabor features applied to vehicle detection," in *18th International Conference on Pattern Recognition*, vol. 1, 0-0 2006, pp. 662 –666.
- [154] Z. Sun, G. Bebis, and R. Miller, "On-road vehicle detection using evolutionary gabor filter optimization," *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 2, pp. 125 – 137, june 2005.
- [155] —, "Monocular precrash vehicle detection: features and classifiers," *IEEE Transactions on Image Processing*, vol. 15, no. 7, pp. 2019 –2034, july 2006.
- [156] T. Ito and K. Yamada, "Preceding vehicle and road lanes recognition methods for rcas using vision system," *Proceedings of IEEE International Intelligent Vehicles Symposium*, pp. 85 – 90, October 1994.
- [157] T. Kato, Y. Ninomiya, and I. Masaki, "Preceding vehicle recognition based on learning from sample images," *IEEE Transactions on Intelligent Transportation Systems*, vol. 3, no. 4, pp. 252 – 260, December 2002.

- [158] Z. Sun, G. Bebis, and R. Miller, "On-road vehicle detection using gabor filters and support vector machines," in *2002 14th International Conference on Digital Signal Processing*, vol. 2, 2002, pp. 1019 – 1022 vol.2.
- [159] S. Sivaraman and M. M. Trivedi, "Real-time vehicle detection using parts at intersections," in *2012 15th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, sept. 2012, pp. 1519 –1524.
- [160] B. Duan, W. Liu, P. Fu, C. Yang, X. Wen, and H. Yuan, "Real-time on-road vehicle and motorcycle detection using a single camera," in *IEEE International Conference on Industrial Technology*, feb. 2009, pp. 1 –6.
- [161] W. Phatanasrirat and S. Phiphobmongkol, "Motorcycle and license plate detection using fixed-size vertical projection and multi-part mean analysis," in *International Conference on Computer Engineering and Technology*, vol. 2, jan. 2009, pp. 43 –47.
- [162] X. Wen, H. Yuan, C. Song, W. Liu, and H. Zhao, "An algorithm based on svm ensembles for motorcycle recognition," in *IEEE International Conference on Vehicular Electronics and Safety*, dec. 2007, pp. 1 –5.
- [163] J. Chiverton, "Helmet presence classification with motorcycle detection and tracking," *IET Intelligent Transport Systems*, vol. 6, no. 3, pp. 259 –269, september 2012.
- [164] C.-C. Chiu, M.-Y. Ku, and H.-T. Chen, "Motorcycle detection and tracking system with occlusion segmentation," in *Eighth International Workshop on Image Analysis for Multimedia Interactive Services*, june 2007, p. 32.
- [165] Y. Ying and Z. Yuhui, "Technique of measuring leading vehicle distance based on digital image processing theory," in *2010 International Conference on Intelligent Computation Technology and Automation (ICICTA)*, vol. 3, may 2010, pp. 674 –677.
- [166] M.-C. Lu, C.-P. Tsai, M.-C. Chen, Y. Y. Lu, W.-Y. Wang, and C.-C. Hsu, "A practical nighttime vehicle distance alarm system," in *IEEE International Conference on Systems, Man and Cybernetics*, oct. 2008, pp. 3000 –3005.
- [167] Z. Qing-Sen and M. Xie, "Study on the method of measuring the preceding vehicle distance based on trilinear method," in *Second International Conference on Computer Modeling and Simulation*, vol. 2, jan. 2010, pp. 475 –479.
- [168] S.-H. Chen and R.-S. Chen, "Vision-based distance estimation for multiple vehicles using single optical camera," in *2011 Second International Conference on*

Innovations in Bio-inspired Computing and Applications (IBICA), dec. 2011, pp. 9 –12.

- [169] C. Yixin, M. Das, and D. Bajpai, “Vehicle tracking and distance estimation based on multiple image features,” in *Fourth Canadian Conference on Computer and Robot Vision*, may 2007, pp. 371 –378.
- [170] G. P. Stein, O. Mano, and A. Shashua, “Vision-based acc with a single camera: bounds on range and range rate accuracy,” *Proceedings of IEEE Intelligent Vehicles Symposium*, pp. 120 – 125, June 2003.
- [171] N. Blanc, B. Steux, and T. Hinz, “Larasidecam: a fast and robust vision-based blindspot detection system,” in *2007 IEEE Intelligent Vehicles Symposium*, june 2007, pp. 480 –485.
- [172] B.-F. Wu, H.-Y. Huang, C.-J. Chen, Y.-H. Chen, C.-W. Chang, and Y.-L. Chen, “The blind spot warning system for daytime and nighttime,” in *2012 IEEE International Conference on Industrial Technology (ICIT)*, march 2012, pp. 591 –597.
- [173] A. Techmer, “Real time motion analysis for monitoring the rear and lateral road,” in *2004 IEEE Intelligent Vehicles Symposium*, june 2004, pp. 704 – 709.
- [174] M. Tucker, A. Heenan, and A. Buchanan, “Real time embedded sensor fusion for driver assistance,” in *Proceedings of 2005 IEEE Conference on Intelligent Transportation Systems*, sept. 2005, pp. 596 – 601.
- [175] C. Chen and Y. Chen, “Real-time approaching vehicle detection in blind-spot area,” in *12th International IEEE Conference on Intelligent Transportation Systems*, oct. 2009, pp. 1 –6.
- [176] D. Polling, M. Mulder, M. van Paassen, and Q. Chu, “Inferring the driver’s lane change intention using context-based dynamic bayesian networks,” in *Systems, Man and Cybernetics, 2005 IEEE International Conference on*, vol. 1, oct. 2005, pp. 853 – 858 Vol. 1.
- [177] S. Habenicht, H. Winner, S. Bone, F. Sasse, and P. Korzenietz, “A maneuver-based lane change assistance system,” in *2011 IEEE Intelligent Vehicles Symposium (IV)*, june 2011, pp. 375 –380.
- [178] U. Dogan, H. Edelbrunner, and I. Iossifidis, “Towards a driver model: Preliminary study of lane change behavior,” in *11th International IEEE Conference on Intelligent Transportation Systems*, oct. 2008, pp. 931 –937.

- [179] A. Doshi and M. Trivedi, “Investigating the relationships between gaze patterns, dynamic vehicle surround analysis, and driver intentions,” in *2009 IEEE Intelligent Vehicles Symposium*, june 2009, pp. 887 –892.
- [180] M. Tsogas, A. Polychronopoulos, N. Floudas, and A. Amditis, “Situation refinement for vehicle maneuver identification and driver intention prediction,” in *2007 10th International Conference on Information Fusion*, july 2007, pp. 1 –8.
- [181] L. Jin, H. Hou, and Y. Jiang, “Driver intention recognition based on continuous hidden markov model,” in *2011 International Conference on Transportation, Mechanical, and Electrical Engineering (TMEE)*, dec. 2011, pp. 739 –742.
- [182] B. Morris, A. Doshi, and M. Trivedi, “Lane change intent prediction for driver assistance: On-road design and evaluation,” in *2011 IEEE Intelligent Vehicles Symposium (IV)*, june 2011, pp. 895 –901.
- [183] R. Schubert and G. Wanielik, “Empirical evaluation of a unified bayesian object and situation assessment approach for lane change assistance,” in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, oct. 2011, pp. 1471 –1476.
- [184] M. Roelofsen, J. Bie, L. Jin, and B. van Arem, “Assessment of safety levels and an innovative design for the lane change assistant,” in *2010 IEEE Intelligent Vehicles Symposium (IV)*, june 2010, pp. 83 –88.
- [185] R. Schubert, “Evaluating the utility of driving: Toward automated decision making under uncertainty,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 1, pp. 354 –364, march 2012.
- [186] M. Ruder, W. Enkelmann, and R. Garnitz, “Highway lane change assistant,” in *2002 IEEE Intelligent Vehicle Symposium*, vol. 1, june 2002, pp. 240 – 244 vol.1.
- [187] R. Schubert and G. Wanielik, “A unified bayesian approach for object and situation assessment,” *IEEE Intelligent Transportation Systems Magazine*, vol. 3, no. 2, pp. 6 –19, summer 2011.
- [188] T. van Dijck and G. van der Heijden, “Visionsense: an advanced lateral collision warning system,” in *2005 IEEE Intelligent Vehicles Symposium*, june 2005, pp. 296 – 301.
- [189] J. Li-sheng, F. Wen-ping, Z. Ying-nan, Y. Shuang-bin, and H. Hai-jing, “Research on safety lane change model of driver assistant system on highway,” *Proceedings of 2009 IEEE Intelligent Vehicles Symposium*, pp. 1051 – 1056, 2009.

- [190] C. Rodemerk, S. Habenicht, A. Weitzel, H. Winner, and T. Schmitt, "Development of a general criticality criterion for the risk estimation of driving situations and its application to a maneuver-based lane change assistance system," in *2012 IEEE Intelligent Vehicles Symposium (IV)*, june 2012, pp. 264 –269.
- [191] A. Amditis, N. Floudas, U. Kaiser-Dieckhoff, T. Hackbarth, B. Van Den Broek, M. Miglietta, L. Danielson, M. Gemou, and E. Bekiaris, "Integrated vehicle's lateral safety: the lateral safe experience," *IET Intelligent Transport Systems*, vol. 2, no. 1, pp. 15 –26, march 2008.
- [192] J. Feng, J. Ruan, and Y. Li, "Study on intelligent vehicle lane change path planning and control simulation," in *2006 IEEE International Conference on Information Acquisition*, aug. 2006, pp. 683 –688.
- [193] C. Hatipoglu, U. Ozguner, and K. A. Redmill, "Automated lane change controller design," *IEEE Transactions on Intelligent Transportation Systems*, vol. 4, no. 1, pp. 13 – 22, 2003.
- [194] G. Xu, L. Liu, Y. Ou, and Z. Song, "Dynamic modeling of driver control strategy of lane-change behavior and trajectory planning for collision prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 3, pp. 1138 –1155, sept. 2012.
- [195] "Mobileye," <http://www.mobileye.com/all-products/>.
- [196] R. Satzoda, S. Suchitra, and T. Srikanthan, "Robust extraction of lane markings using gradient angle histograms and directional signed edges," in *IEEE Intelligent Vehicles Symposium - IV'2012*, June 2012, pp. 754–759.
- [197] —, "Gradient angle histograms for efficient linear hough transform," in *IEEE International Conference on Image Processing (ICIP)*, Nov 2009, pp. 3273 – 3276.
- [198] W. Rong-ben, Y. Tian-hong, J. Li-sheng, C. Jian-wei, and G. Bai-yuan, "Edge extraction method study based on maximum entropy for linear lane identifying and tracking," in *IEEE Intelligent Vehicles Symposium - IV'2005*, 2005, pp. 849 – 854.
- [199] J. Canny, "A computation approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, pp. 726 – 741, 1987.
- [200] S. Suchitra, R. K. Satzoda, and T. Srikanthan, "Exploiting inherent parallelisms for accelerating linear Hough transform computation," *IEEE Transactions on Image Processing*, vol. 18, no. 10, pp. 2255 – 2264, 2009.

- [201] ———, “Selective extraction of curvilinear edges using gradient angle histograms,” in *International Conference on Communications and Signal Processing*, Feb. 2011, pp. 358–362.
- [202] R. K. Satzoda, S. Suchitra, and T. Srikanthan, “Parallelizing Hough transform computation,” *IEEE Signal Processing Letters*, vol. 15, no. 1, pp. 297 – 300, 2008.
- [203] B. Parhami, *Computer Arithmetic*. Oxford University Press, 2000.
- [204] F. Zhou and P. Kornerup, “A high-speed Hough transform using CORDIC,” *Proceedings of International Conference of VLSI Digital Signal Processing*, pp. 482–487, June 1995.
- [205] P. K. Meher, T. Srikanthan, and J. C. Patra, “Scalable and modular memory-based systolic architectures for Discrete Hartley Transform,” *IEEE Transactions on Circuits and Systems - I: Regular Papers*, vol. 53, no. 5, pp. 1065 – 1077, May 2006.
- [206] S. M. Karabernou and F. Terranti, “Real-time FPGA implementation of Hough Transform using gradient and CORDIC algorithm,” *Journal of Image and Vision Computing*, vol. 23, pp. 1009 – 1017, July 2005.
- [207] C. Espinosa and M. A. Perkowski, “Hierarchical Hough transform based on pyramidal architecture,” *Proc. of Eleventh Annual International Phoenix Conference on Computers and Communications*, pp. 743–750, 1992.
- [208] J. Princen, J. Illingworth, and J. Kittler, “A hierarchical approach to line extraction,” *Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 92–97, 1989.
- [209] S. B. Yacoub and J. M. Jolion, “Hierarchical line extraction,” *IEE Proc. of Vision Image and Signal Processing*, vol. 142, no. 1, pp. 7–14, 1995.
- [210] A. K. Majumdar, “Design of an ASIC for straight line detection in an image,” *Proceedings of Thirteenth International Conference on VLSI Design*, pp. 128 – 133, January 2000.
- [211] C. Jiang and M. Ward, “Shadow identification,” in *1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, jun 1992, pp. 606 –612.
- [212] R. Duda, P. Hart, and D. Stork, *Pattern classification*, ser. Pattern Classification and Scene Analysis: Pattern Classification. Wiley, 2001. [Online]. Available: <http://books.google.com/books?id=YoxQAAAAMAAJ>
- [213] “Fluke 416D Laser Distance Meter,” <http://www.fluke.com/416D>.

- [214] “Computational Vision Archive, Caltech,” <http://www.vision.caltech.edu/html-files/archive.html>, 1999-2001.
- [215] “Datasets for testing by author,” <https://docs.google.com/folder/d/0B6eAY2O8msWGRm5mODBXckVPTXc/edit>, 2012.
- [216] S. T. Police, *2012 Official Handbook on Basic Theory of Driving*, 2012.