Detection of road markings for advanced driver assistance

Suchitra Sathyanarayana

2013

Suchitra Sathyanarayana. (2013). Detection of road markings for advanced driver assistance. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/54995>

<https://doi.org/10.32657/10356/54995>

NANYANG
TECHNOLOGICAL
UNIVERSITY

**DETECTION OF ROAD MARKINGS FOR
ADVANCED DRIVER ASSISTANCE**

SUCHITRA SATHYANARAYANA
SCHOOL OF COMPUTER ENGINEERING

2013

**DETECTION OF ROAD MARKINGS FOR
ADVANCED DRIVER ASSISTANCE**

SUCHITRA SATHYANARAYANA

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

I dedicate this dissertation to my beloved Guru, Paramahansa Sri
Nithyananda, who has enriched my life in ways unfathomable, and who
resides in my heart as an ever-guiding light.

Acknowledgements

I would like to express my heartfelt gratitude to my supervisor, Professor Thambipillai Srikanthan, for his unceasing support, time, valuable guidance and constant encouragement that he offered throughout this thesis. I am greatly indebted to him for the constructive suggestions and ideas, that have immensely contributed to the unfolding of the research work. I also want to thank him for having created an enjoyable research atmosphere and for providing all the advanced facilities at Center for High Performance Embedded Systems (CHiPES).

I would like to offer my special thanks my teammate and husband, Ravi Kumar Satzoda, who has played a vital role at every step of my thesis. The regular team discussions and brainstorming have greatly contributed towards the research work. More importantly, it was his support, encouragement and love that made it possible for me to complete this thesis.

I would like to thank my colleagues, team members, and friends, specially Supriya and Nirmala, for their support in one way or the other towards the completion of the thesis. My gratitude goes out to Mr.Muthukrishnan for helping me with the test drives and generation of dataset for the purpose of evaluation. Thanks to the technician Mr. Jeremiah Chua for the technical support throughout.

I would like to thank my parents, siblings and parents-in-law for their encouragement and support through the period of the thesis. My humble gratitude to my spiritual master, Paramahansa Sri Nithyananda, for being an eternal source of inspiration and inner strength.

Finally, I would like to acknowledge School of Computer Engineering, Nanyang Technological University for providing this opportunity to me for carrying out this research work.

Contents

List of Figures	vii
List of Tables	xv
Abstract	xvii
1 Introduction	1
1.1 Motivation	1
1.2 Objectives	2
1.3 Scope of the thesis	2
1.4 Research Contributions	4
1.4.1 Related Publications	6
1.5 Organization of the Dissertation	7
2 Literature Survey	11
2.1 Introduction	11
2.2 Role of road markings in ADAS	11
2.3 Sensing Technologies for Road Marking Detection	14
2.4 Survey of Algorithms for Road Marking Detection	16
2.4.1 Lane Marking Detection	17
2.4.1.1 Feature Extraction	17
2.4.1.2 Outlier Removal using Lane Models	20
2.4.1.3 Classification of lane boundary markings	27
2.4.2 Arrow Markings Detection	28
2.4.3 Pedestrian Crossing Detection	31
2.5 Summary	32

CONTENTS

3	Basic Linear Road Markings	37
3.1	Introduction	37
3.2	Properties of BLM	38
3.3	Straight Line Edge Map (SLEM) for Feature Extraction	40
3.3.1	Block-based SLEM generation	43
3.3.1.1	Deriving block settings	43
3.4	Deploying SLEM for Lane Marking Feature Extraction	46
3.5	Signed Edges and Hough Transform for Outlier Removal	48
3.6	Dataset for Performance Evaluation	52
3.7	Performance Evaluation	53
3.8	Enhancing Robustness using Temporal History	61
3.9	Overall BLM Detection Algorithm	64
3.10	Distinguishing Solid & Dashed Single Lane Markings	66
3.10.1	Approach 1: Spatial Continuity Analysis	67
3.10.2	Approach 2: Temporal Continuity Analysis	70
3.11	Detecting Horizontal BLM	76
3.12	Summary	78
4	Architecture for BLM Detection	81
4.1	Introduction	81
4.2	Architecture for Detecting and Distinguishing BLM	81
4.3	Accelerating Hough Transform for Increased Throughput	82
4.3.1	Additive Property of Hough Transform	83
4.3.2	Parallel Additive Hough Transform	86
4.4	Computational Cost Analysis	88
4.5	Summary	94
5	Complex Linear Road Markings	95
5.1	Introduction	95
5.2	Challenges in detecting CLM	96
5.3	Adaptive Hough Peak Detection (AHPD)	100
5.3.1	AHPD Algorithm	101
5.3.1.1	Stage 1: Shortlisting Peak Candidates	102
5.3.1.2	Stage 2: Confirming Hough Peaks Using Peak Patterns	111

5.4	Evaluation of AHPD	116
5.4.1	Accuracy Evaluation	116
5.4.2	Computation Cost Analysis	119
5.5	CLM Detection Using AHPD	121
5.5.1	Detecting Double Linear Markings Using AHPD	121
5.5.2	Detecting Zigzag Markings Using AHPD	122
5.6	Results and Discussion	127
5.7	Scale & Distortion Invariance for Linear Marking Detection	130
5.8	Architecture for Detecting BLM and CLM	136
5.9	Summary	138
6	Detecting Arrow Markings	139
6.1	Introduction	139
6.2	Scale & Distortion Invariance for Arrow Marking Detection	141
6.2.1	Limitations of Existing Methods	143
6.3	Scale & Distortion Invariant Shape Signatures for Defining Arrows . . .	145
6.4	Arrow Detection and Classification using Shape Signatures	149
6.4.1	Detecting the Main Arrow Tail	149
6.4.2	Detecting Arrow Heads	152
6.4.3	Arrow Classification based on Sub-components of Arrows	152
6.5	Results & Discussion	154
6.6	Detecting Arrow Markings in Side Lanes	159
6.6.1	Algorithm	160
6.7	Architecture for Arrow Marking Detection	164
6.8	Summary	166
7	Detecting Pedestrian Markings	169
7.1	Introduction	169
7.2	Scale & Distortion Invariant Pedestrian Marking Detection	169
7.3	Scale & Distortion Invariant Shape Signatures for Defining Pedestrian Markings	170
7.4	Pedestrian Marking Detection Using Shape Signatures	172
7.5	Other Markings as Cues for Pedestrian Marking Detection	178
7.6	Results	180

CONTENTS

7.7	Architecture for Pedestrian Marking Detection	184
7.8	Summary	184
8	Conclusions and Future Research	187
8.1	Suggestions for Future Research	190
A	List of road markings in Singapore and their significances	193
	References	197

List of Figures

1.1	Broad classification of road markings that are being considered in this research work (a) Basic linear (b) Complex linear (c) Arrow markings and (d) Pedestrian crossings.	3
3.1	Basic linear road markings.	38
3.2	GAH of different types of features (a) GAH of straight line feature (b) GAH of a curved feature (c) GAH of noise.	40
3.3	GAH of straight and non-straight pixels, (a) Image I (b) Edge map (c) GAH (d) SLEM of I	42
3.4	Effects of varying block size and T_{GA} , (a): Original Image of size 294×512 , (b): Sobel Edge Map with $T_{ \nabla I } = 130$, (c): SLEM with $T_{GA} = 16$, $b_{size} \times b_{size} = 128 \times 128$, (d): SLEM with $T_{GA} = 16$, $b_{size} \times b_{size} = 16 \times 16$, (e): SLEM with $T_{GA} = 16$, $b_{size} \times b_{size} = 64 \times 64$	44
3.5	SLEM of test images; First Column: Test Image; Second Column: Sobel edge map; Third Column: SLEM.	45
3.6	Illustration of SLEM generation for lane markings (a) Near view containing lane markings and shadows, divided into blocks of size 90×128 (b) Image block chosen for illustration (c) Initial edge map with both lane marking edges and the shadow edges (noise), (d) Gradient angle histogram (GAH) for the input edge pixels of block, (e) Edge pixels E_{SL} of straight line edges corresponding to the peak centering -20° . (f) Summary of the underlying steps.	47
3.7	Straight line Hough Transform (a) Collinear points in image plane (b) Intersecting sinusoids in Hough space (c) Implementation as a two-dimensional array of accumulators.	49

LIST OF FIGURES

3.8	Detecting line pair satisfying lane marking conditions using simplified Hough Transform and signed edges.	50
3.9	Illustration of outlier removal process (a) Image block of size 90×128 with lane markings and other edges like tar color change along the direction of the lane (Type C scenario), (b) Straight line edge map E_{SL} obtained using GAH in feature extraction step, (c) Left directional edge map E_{SLl} with the left lane marking edge and the tar color change edge also, (d) Right directional edge map E_{SLr} with the right lane marking edge, (e) Resulting straight lines that are identified as having the lane markings after analysing the Hough spaces.	51
3.10	Map showing routes in Singapore along which test videos were captured.	52
3.11	(a) Test vehicle used for capturing dataset (b) Camera set up inside test vehicle (c) Sample image as captured by the camera (d) Image after cropping.	53
3.12	Near lane detection in complex scenarios: (a) & (b) shadows of trees and other random features, (c) shadows of overhead bridges falling on lane markings (d) presence of vehicle shadows in the direction of lane marking (e) edges due to tar color change edges in the direction of lane markings.	57
3.13	Lane marking detection using proposed method in the presence of edges that are along the direction of lanes towards the vanishing point: (a) & (b) show tar road color changes that create edges along the lanes, (c) & (d) show the presence of vehicle shadows in the direction of lanes. . . .	58
3.14	Lane marking detection in the presence of other road markings.	59
3.15	Examples of misdetections (a) Presence of other road markings that resemble lane marking (b) Very faint and worn out markings (c) Pavement border resembling dashed lane marking.	61
3.16	Example of lanes present/absent in near view; (a) and (b) taken within 1s time window.	63
3.17	Incorporating temporal history into the lane detection process for improved robustness.	63
3.18	Lane position after temporal tracking.	65

LIST OF FIGURES

3.19	Flowchart summarizing the steps in BLM Detection. The arrows between the signed edge map generation and HT blocks are indicated in both directions to convey the feedback from first processing the positive edges, after which HT is performed only for the selected angle range on the negative edges.	66
3.20	Distinguishing lane markings as dashed and solid through two pronged approach (i) Spatial continuity check and (ii) Temporal continuity check.	67
3.21	Generating mask for spatial continuity check (a) Sample image block containing left lane marking (b) Edge map (c) Result of AND-ing the edge map with mask.	68
3.22	Using spatial continuity analysis for distinguishing solid vs dashed lane markings (a) Image divided into blocks.	68
3.23	Using spatial continuity analysis for distinguishing solid vs dashed lane markings.	69
3.24	Using temporal continuity check for differntiating solid vs dashed lane markings.	72
3.25	Flowchart summarizing steps in spatial and temporal continuity analysis for distinguishing lane markings.	75
3.26	Illustration of horizontal marking detection steps.	77
3.27	Results of horizontal marking detection on sample sequence.	78
4.1	Computational blocks in linear marking detection.	82
4.2	Cascading linear marking detection modules for detecting longitudinal & transverse BLMS.	83
4.3	Deriving the additive property of Hough Transform.	84
4.4	Hough Transform computation (a) input edge map, (b) Hough Transform computed using standard equation (3.4).	85
4.5	Illustrating blocks, local origins and relative positions of pixels in blocks.	87
4.6	Comparison of Hough spaces; Hough spaces for Sobel edge maps (first row) and <i>SLEM</i> (second row) for (a) Chimney, (b) Keys, (c) House are shown.	92
5.1	Complex Linear Road Markings.	96
5.2	Issues with conventional Hough peak detection for detecting Zigzag. . .	97

LIST OF FIGURES

5.3	Issues with conventional Hough peak detection for detecting double linear marking.	99
5.4	Butterfly pattern of Hough peak as shown in [156].	101
5.5	Observations to illustrate valid peaks and noise (a) peaks are distinctly above clutter of noise (b) clutter of peaks means noise.	103
5.6	H_θ with peaks selected.	104
5.7	Accumulation counts are divided into bins, each of size $s_3 = 3$. Dotted lines represent the bins and the number of peaks in each bin are listed on the left.	106
5.8	Illustration showing valid and invalid peaks. Both the peaks in all the three cases have the same peak height but their separations are different: (a) local maxima, which is a part of a major peak (b) two valid peaks separated by a distance (c) two valid peaks separated by a valley.	108
5.9	Illustration on the process of splitting P_θ	109
5.10	(a) Hough space $\mathcal{H}$ computed from Hough transform, (b) Reduced Hough space $\mathcal{H}'$ obtained after applying Stage 1 of AHPD.	111
5.11	Zoomed in versions of the Hough spaces in Fig. 5.10 showing the butterfly in $\mathcal{H}$ and butterfly-like pattern in $\mathcal{H}'$	112
5.12	Stage 2.	114
5.13	Sample images chosen (a) synthetic image with closeby edges (b) airplane with lines of varying length (c) bed-lamp with thin features having closeby edges.	116
5.14	Synthetic test image containing closeby lines (a) image (b) Hough space.	116
5.15	Hough peak detection results on airplane image with <code>houghpeaks()</code> and AHPD (a) <code>houghpeaks(NHoodSize = default, NUMPEAKS = 10, Threshold = 0.35*max(H))</code> ; (b) <code>houghpeaks(NHoodSize = 5x5, NUMPEAKS = 10, Threshold = 0.35*max(H))</code> ; (c) <code>houghpeaks(NHoodSize = 5x5, NUMPEAKS = 30, Threshold = 0.35*max(H))</code> ; (d) <code>houghpeaks(NHoodSize = default, NUMPEAKS = default, Threshold = 0.35*max(H))</code> ; (e) AHPD with <code>Threshold = 0.35*max(H)</code>	117

LIST OF FIGURES

5.16	Hough peak detection results on bed-lamp image using (a) houghpeaks(NHoodSize = 5x5, NUMPEAKS = 10, Threshold = 0.35*max(H)), (b) houghpeaks(NHoodSize = 5x5, NUMPEAKS = 30, Threshold = 0.35*max(H)), (c) AHPD.	119
5.17	Illustration to show AHPD successfully detecting closely lines in a double lane marking.	121
5.18	Detecting double lane markings and classification into (1) single solid (2) single dashed (3) double solid and (4) double dashed.	123
5.19	Aerial view of zigzag marking.	123
5.20	Possible zigzag marking scenarios (a) Most common with solid marking preceding zigzag (b) Case with dashed marking at cross road preceding zigzag.	124
5.21	Region of interest for zigzag detection.	127
5.22	Classification results of single/double and solid/dashed.	128
5.23	Images on which Algo. 3 are applied and results summarized in Table 5.2.	129
5.24	Main degrees of freedom of vehicle (a) surging - forward/backward movement and (b) yawing - turning left/right.	132
5.25	Effect of scale and perspective on road markings.	133
5.26	Effect of scale on linear marking detection.	133
5.27	Unchanging appearance of linear markings along the road with scale changes.	135
5.28	Unchanging appearance of linear markings along the road with scale changes.	135
5.29	Effect of viewpoint changes on linear marking detection.	135
5.30	Overall block architecture for detection of simple and complex linear road markings.	136
5.31	Block architecture for complex linear road markings detection.	137
6.1	(a) Potential scenario that would benefit from a ‘microGPS’, (b) Snapshot of some advanced features of Tomtom [1].	139
6.2	Arrow types on roads.	141
6.3	Varying appearance of road marking with changing viewpoints.	141
6.4	Changing scales of arrow marking with vehicle motion.	142

LIST OF FIGURES

6.5	Illustration of classic correlation.	144
6.6	Variations in Hough spaces generated by same arrow marking from different viewpoints: (a) shows Hough space of arrow in Fig. 6.3(b), and (b) shows Hough space of arrow in Fig. 6.3(c).	145
6.7	(a) Sample forward arrow (b) E_l (c) E_r	147
6.8	Left-slant (solid) and right-slant (dashed) lines are indicated on different arrow parts: (a) Main arrow tail, (b) Forward facing arrow head, (c) Right arrow head, and (d) Left arrow head.	148
6.9	(a)-(d): Top row shows the candidate line pairs for main arrow tail of a simple forward arrow; Bottom row shows the axis for each of the line pairs.	150
6.10	Sample road scene showing: (a) axis of main arrow tail passing through the vanishing point, (b) axis of side arrow head being a horizontal. . . .	151
6.11	Figure showing how the axis of main arrow tail is computed.	152
6.12	Block diagram showing the proposed method to detect different arrow markings.	153
6.13	Comparison between AHPD and MATLAB <code>houghpeaks()</code> on arrow image (a) Hough space of edge map shown in (b), (c) lines detected using <code>houghpeaks()</code> (d) lines detected using AHPD (e) AHPD result with an alternate visualization.	154
6.14	Different types of arrow markings are detected using the proposed method.	156
6.15	Scale and distortion invariance of proposed method.	156
6.16	Output of proposed arrow detection and classification on an image sequence; Outputs (i) “No Arrow” when no arrow seen in near-view (ii) “Arrow Detected” when arrow is detected, but not classified yet (iii) arrow type when arrow is close enough to perform classification.	158
6.17	Forward-left arrow in host lane versus neighboring lane (a) faint marking and compressed features (b) prominent features and less distortion. . . .	160
6.18	Relationship between arrows in host and side lanes.	162
6.19	Deducing arrow type based on edge pixel count in arrow head masks. . . .	163
6.20	Flowchart summarizing steps in arrow detection in side lanes.	165
6.21	Block diagram summarising computational modules for arrow detection and classification.	166

7.1	View of a pedestrian crossing (a) as seen by a pedestrian (b) as seen by a driver.	170
7.2	Scale and distortion effects on pedestrian marking detection (a) and (c) are captured at varying scales, while (a), (b) and (d) are captured with varying camera viewpoints.	171
7.3	(a) Aerial view and (b) Perspective view as seen from on-board camera, based on which signature of pedestrian marking is defined.	171
7.4	Image sequence illustrating the various components of pedestrian marking signature that are perceived at various scales (a) Horizontal is the most prominent when vehicle is furthest away (b) Both horizontal and stripes are seen when vehicle moves closer to pedestrian marking (c) Stripes are the most prominent features when vehicle is at the pedestrian marking.	172
7.5	Properties displayed by edges of stripes (S_E).	174
7.6	Lettered markings that can be potentially mistaken for a pedestrian marking.	176
7.7	Flowchart describing steps for detecting pedestrian marking.	178
7.8	Component parts of a pedestrian crossing region (A) Zigzag marking (B) Horizontal linear marking (C) Pedestrian Crossing.	179
7.9	Cascading computational modules for detecting pedestrian markings. . .	180
7.10	Pedestrian marking detection on a video sequence.	183
7.11	Block architecture summarizing pedestrian marking detection.	184

LIST OF FIGURES

List of Tables

3.1	Description of test videos	54
3.2	Results of detecting basic linear markings along the road	56
3.3	Results of temporal continuity check to distinguish lane markings	73
4.1	Reduction in total edge pixel count using SLEM	89
4.2	Comparison of total computation time	93
5.1	Computation cost comparison of AHPD with [156]	121
5.2	Results of Zigzag Detection	131
6.1	Location of Right and Left Slant Lines for Each Arrow Part	149
6.2	Arrow Classification using Presence/Absence of Arrow Parts	153
6.3	Results of Arrow Detection	155
6.4	Possible Arrow Combinations	161
6.5	Classification of Arrows in Neighboring Lanes using Edge Counts in Masks	164
7.1	Detection of Stripes in Pedestrian Markings	181
A.1	Road Markings Along the Road and their Significance	193

Abstract

Automatic detection of road markings will enhance the capabilities of Advanced Driver Assistance Systems (ADAS) as road markings denote vital information pertaining to traffic safety and navigation. However, little work has been done in the area of vision-based detection of road markings in general, and existing literature was found to be largely confined to lane marking detection. In this thesis, algorithms and architectures have been proposed to identify road markings that are categorized into (a) basic linear markings (BLM) (b) complex linear markings (CLM) (c) arrow markings and (d) pedestrian markings.

The identification of BLM such as dashed and solid lane markings has been tackled first. An efficient feature extraction process based on gradient angle histograms was introduced for shortlisting potential lane marking candidates in a block-based manner, resulting in customized edge maps (called Straight Line Edge Map or SLEM). A study on the relationship between the block-size and the quality of SLEM was conducted to identify appropriate block settings leading to successful extraction of lane markings despite noisy edge pixels such as tree shadows. The outlier removal step involves decomposing the SLEM into positive and negative edges based on intensity transition characteristics, which are then subjected to the Hough Transform (HT). The thickness criterion was also introduced during the detection of HT peaks to eliminate artifacts resembling lane markings. The proposed GAH based preprocessing approach is shown to reduce the number of HT computations by more than 50%, due to a notable reduction in the number of edge pixels and angle range for HT. It is shown that the proposed lane marking detection technique yields high detection accuracy, ranging from 98% to 99%, upon validation on an extensive dataset with more than 6,000 image frames representing different illumination, weather and complex road conditions.

In order to distinguish solid and dashed single lane markings, a two-pronged technique based on continuity analysis of the lane markings in the spatial and temporal domains was proposed. It was shown that this method successfully distinguishes dashed and solid lane markings for all the test sequences considered in the dataset. The proposed BLM detection module can be readily configured to detect horizontal BLMs such as stop lines. A generic version of the lane marking detection algorithm was realized to detect BLMs for a given thickness.

A novel parallel method for HT computations called Additive Hough transform (AHT) was proposed to drastically collapse the complexity of HT computations by replacing the trigonometric operations with simple additions. It relies on dividing the image into uniform blocks, and processing them in parallel, by exploiting the proposed additive property of HT. A study on how block size affects the compute efficiency was conducted to show that AHT reduces the total computation time by at least k^2 times as compared to existing HT architectures for a $k \times k$ grid, translating to a speed up of at least 67 times against existing methods for an 8×8 grid. It was demonstrated that the block size in AHT and SLEM can be adjusted to achieve the required speedup and improve the quality of edge map, respectively.

Detecting CLMs, such as double linear markings and zigzag markings, is more challenging than detecting BLM, particularly due to inherent limitations of the conventional Hough Peak detection. This motivated the development of the novel Adaptive Hough Peak Detection (AHPD) algorithm that adaptively identifies valid peaks in a noisy Hough space. The method relies on an iterative climbing process to generate a reduced Hough space, after which shape properties of peaks are exploited to isolate valid peaks from local minima. This not only eliminates redundant peak detections, but also detects lines in close proximity. The AHPD is also aided by the signed edges for separating lines in close proximity. Unlike conventional methods, AHPD is fully automatic and does not necessitate manual setting of the threshold and non-maximal suppression window size. This also makes it possible to automatically detect uneven lengths of lines as in the case of zigzag markings more deterministically. Evaluation of the AHPD on the 101 CMU object database confirms that it can successfully detect the major line features of interest, including lines of close proximity and lines of varying lengths amidst noise. Evaluation using 640 road images consisting of 120 zigzag images and 120 double linear lane markings shows that CLMs can be detected with 98% accuracy.

Detection of road arrow markings is complex mainly due to varied appearances of the same marking under different scales and perspectives. A scale-invariant technique, based on shape signatures, is proposed for the efficient detection and classification of arrow markings. It relies on decomposing the arrow markings into its constituent parts by representing each part with a unique shape signature using a combination of GAH, signed edges and HT. The shape signatures associated with an arrow marking are not only invariant across scales and distortion effects, but are also easy to identify by employing methods proposed for BLM and CLM. This has led to the elimination of exhaustive matching operations associated with conventional template matching approaches. The method was evaluated on a database of over 200 arrow markings and it was shown to yield a detection and classification accuracy of 97% and 98% respectively. An efficient method to identify arrows in neighboring lanes was also proposed by relying on position and type of arrow detected in the host lane. It was shown to withstand the notable distortions and compression of the arrow marking features in the neighboring lanes.

An algorithm to detect pedestrian markings was proposed next by incorporating techniques for detecting scale and distortion invariant shape signatures that are associated with the stripes of the pedestrian marking. Other road markings in the vicinity of the pedestrian crossing have also been employed to act as cues to predict an upcoming pedestrian marking so as to further enhance the robustness of the detection process. A unified engine to detect BLMs, CLMs, arrow and pedestrian markings is presented using the core modules such as GAH, signed edge generator, AHT and AHPD. Finally, the major contributions in this thesis have paved the way for propelling further research in this emerging area of interest and for realizing high performance ADAS at low cost.

Glossary

ADAS – Advanced Driver Assistance Systems
AHPD – Adaptive Hough Peak Detection
AHT – Additive Hough Transform
BLM – Basic Linear Road Markings
CLM – Complex Linear Road Markings
CMOS – Complementary Metal-oxide Semiconductor
EDF – Edge Distribution Function
FAST – Features from Accelerated Segment Test
FIFO – First In First Out
FN – False Negative
FP – False Positive
GAH – Gradient Angle Histogram
GPS – Global Positioning System
HT – Hough Transform
HOG – Histogram of Oriented Gradients
IPM – Inverse Perspective Mapping
KNN – K-Nearest Neighbor
LANA – Lane Finding in Another Domain
LHT – Linear Hough Transform
MSER – Maximally Stable Extremal Regions
NCC – Normalized Cross Correlation
PHT – Probabilistic Hough Transform
PPHT – Progressive Probabilistic Hough Transform
RANSAC – RANdom SAmple Consensus
RoI – Region of Interest
RHT – Randomized Hough Transform
SIFT – Scale-invariant Feature Transform
SIMD – Single Instruction Multiple Data
SLEM – Straight Line Edge Map
SVM – Support Vector Machine
TP – True Positive
TN – True Negative
WPM – Warp Perspective Mapping

1

Introduction

1.1 Motivation

Road markings, traffic signs and traffic lights are integral components of a typical urban road scenario. They serve to constantly feed the driver with supplementary information related to road safety, navigation and traffic regulations, to enable informed decision making while driving. Whilst many techniques exist in the market for detecting traffic signs and traffic lights, some of which are even deployed in vehicles, little has been done in the area of detecting and classifying road markings. Information contained in road markings is many times unique i.e. not conveyed by traffic signs or lights. Hence it is crucial to rely on road markings to enforce road safety. For example, while speed limits are captured in both traffic signs and road markings, valid traffic directions in lanes are conveyed only by road arrow markings, and not represented in traffic signs. Similarly, the type of lane marking, such as dashed, solid, single, double, zigzag contain exclusive information related to lane keeping, lane change, parking etc. Detecting road markings can therefore directly aid in a range of Advanced Driver Assistance Systems (ADAS). Infact, this has been a topic of recent research in literature and the intelligent vehicles community acknowledges the benefits of detecting various kinds of road markings. Detecting type of lane marking, for instance, can aid in lane change assistance systems. Detecting the type of arrow markings can enhance GPS-based navigation systems, by directing vehicles to the appropriate lane in a timely manner, in order to facilitate turnings. Detecting pedestrian markings and alerting the driver about its presence well in advance can significantly improve safety on roads. In order

1. Introduction

to detect these road markings, vision sensors or cameras are most popular. This is because active sensors such as RADAR, LIDAR etc. mainly rely on depth information, which in this case is not a suitable parameter to separate road markings from the flat road surface. Cameras on the other hand capture the road scene as seen by the driver, and hence ideal for gathering visual cues like road markings. Moreover, with dropping camera costs and the non-intrusive nature of cameras, vision-based techniques for road marking detection have gathered much popularity over recent years.

1.2 Objectives

The objective of this research work is to propose, develop and validate novel vision-based techniques to detect and distinguish road markings, in an urban traffic scenario. For this purpose, the set of major road markings in Singapore roads, described in Sec. 1.3, are considered. Further, the detection of road markings needs to be real-time, as it is mainly intended to assist the driver towards informed decision making while driving. Therefore, this research aims to develop computationally efficient techniques that can lend well for real-time capable applications. The techniques should also cater to a range of complex cases that are typically encountered in a real driving scenario, such as varying weather conditions like rain, overcast skies, sunny days with strong shadows, night lighting and presence of other noisy artifacts such as vehicles, shadows from trees, vehicles, overhead bridges etc. Hence, achieving robustness in the proposed techniques for detecting road markings is a key objective.

1.3 Scope of the thesis

Road markings can be grouped into various categories based on criteria such as continuity (solid or dashed), number (single, double), position (along the road i.e. ‘longitudinal’ or across the road i.e. ‘transverse’), and color (white, yellow or red).

In this thesis, the following categories of road markings are considered for detection and classification, based on their shapes and underlying complexities.

- **Basic linear markings (BLM)**: referring to the class of road markings defined by a single straight line of a certain thickness, painted either along (longitudinal) or across (transverse) the road, either solid or dashed in nature.

Figure 1.1: Broad classification of road markings that are being considered in this research work (a) Basic linear (b) Complex linear (c) Arrow markings and (d) Pedestrian crossings.

- **Complex linear markings (CLM):** referring to road markings that comprise of multiple linear segments. These could either be double straight line markings, in the longitudinal or transverse directions, or zigzag markings that run along lane boundaries in the longitudinal directions.
- **Arrow markings:** arrows on lanes indicating allowable directions of traffic flow, which are of the following types: (a) Simple Forward Arrow, (b) Simple Right Arrow, (c) Simple Left Arrow, (d) Left/Right Arrow, (e) Forward/Left Arrow, (f) Forward/Right Arrow and (g) Forward/Left/Right Arrow.
- **Pedestrian markings:** also termed as a zebra crossing, consisting of a set of parallel lines oriented along the lanes and painted across the road, indicating a zone where pedestrians are given priority to cross the road, and vehicles are expected to wait.

1. Introduction

These markings and their categories are summarized in Fig. 1.1.

1.4 Research Contributions

The major contributions of this research work are:

- A thorough survey was carried out on existing related work and their limitations were identified.
- A computationally efficient technique to detect single lane marking robustly in a wide range of illumination, weather and shadow conditions has been proposed. As part of this, a novel block-based line feature extraction step has been proposed using gradient angle histograms (GAH). Further, an outlier removal method has been proposed based on signed edges, Hough Transform (HT) and use of priori information such as thickness of lane marking. It is shown that the lane marking detection method has many attractive features both in terms of computational savings, as well as its capacity to eliminate unwanted features. The proposed method has been evaluated exhaustively on more than 6,000 image frames and it is shown that the method yields a high accuracy rate of over 98%. A comprehensive study on effect of changing block size on quality of feature extraction has been undertaken, and the approach for appropriate block settings has been established. Also, techniques to distinguish solid lane markings from dashed are proposed using temporal and spatial continuity analysis. A reusable computation module to detect BLMs has been developed, and it has been deployed for detecting horizontal line markings.
- An architecture for BLM detection has been proposed. In order to circumvent the computational bottlenecks posed by the HT computations, a novel technique to parallelise the HT has been proposed, based on the additive property of HT that has been identified. It has been shown that the proposed additive HT (AHT) yields an acceleration of at least 67 times against state-of-the-art Hough based line detection techniques for an 8×8 grid. The computational advantage of the proposed techniques for BLM detection has been presented in detail.

- In order to overcome the challenges of detecting CLM using conventional Hough peak detection methods, a novel adaptive Hough peak detection (AHPD) algorithm has been developed. It is shown that this method relies on an iterative climbing process and shape information of the Hough peaks to adaptively set threshold values above noise levels. The proposed AHPD effectively detects lines of close proximity and lines of varying length, without any manual intervention. This algorithm was evaluated on a general object database from CMU and is shown to successfully detect lines of interest. Algorithms to detect zigzag markings and double lane markings using the proposed AHPD method are developed. Further, effect of scale and distortion on the detection and classification of all linear markings have been studied.
- Techniques to detect arrow markings and classify them using unique shape signatures are proposed. One of the key features of the proposed approach is that the detection is invariant to changes in scale and distortion due to perspective or viewpoint changes. An evaluation of the method has been performed on a dataset containing more than 250 arrow images. Methods to detect arrow markings in neighboring lanes is also proposed by relying on information from host lanes, thereby overcoming the challenges of arrow features being more distorted and compressed in side lanes as compared to host lanes.
- Algorithm to detect pedestrian markings using unique shape signatures is proposed, which is scale and distortion invariant. Also, a framework is proposed wherein cues from other markings that act as indicators for an upcoming pedestrian marking are used effectively to predict the pedestrian marking. A unified computation framework is developed that integrates the various computational blocks proposed for detecting road markings in this thesis.
- A dataset of road marking images including various kinds of lane markings, horizontal markings, arrows and pedestrian crossings, was created and is made available online for research community at [4]. The dataset comprises of image sequences and images taken under varying road scenarios like presence of shadows, illumination changes, different times of the day etc. The repository contains over 7,000 test images.

1. Introduction

1.4.1 Related Publications

The following papers were published in international conferences and journals, in which some of the proposed research in this thesis is presented.

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.

To Be Submitted

1. S. Suchitra, R. K. Satzoda, and T. Srikanthan , “Automated Hough Peak Detection”, *IEEE Transactions on Image Processing*.
2. R. K. Satzoda, S. Suchitra and T. Srikanthan , “Local-illumination Invariant Vehicle Detection Techniques”, *IEEE Intelligent Vehicles Symposium - IV’2014*, 2014.
3. R. K. Satzoda, S. Suchitra and T. Srikanthan , “Multiple Lane Detection with Varying Edge Prominence”, *IEEE Intelligent Vehicles Symposium - IV’2014*, 2014.

1.5 Organization of the Dissertation

This thesis is organized into the following 8 chapters:

1. Chapter 1 introduces the thesis and contains background, motivation, objectives and major research contributions.

1. Introduction

2. Chapter 2 provides a review of the existing techniques and methodologies in the area of detection and classification of road markings for Advanced Driver Assistance Systems (ADAS). Starting with an overview of sensing technologies in ADAS, the feasibility of the various sensors for road marking detection is reviewed. Next, a comprehensive review of various existing techniques for detecting road markings such as lane markings, arrow markings and pedestrian crossings is undertaken. Their strengths and weaknesses are also outlined. Lastly, the underlying computational blocks commonly encountered for the purpose of road markings extraction are discussed.
3. In Chapter 3, computationally efficient techniques for detection of basic linear road markings (BLM) were proposed, starting with single lane markings. Firstly, an efficient edge based feature extraction step is proposed to generate edge maps containing only straight line features of interest. Next, an outlier removal step is proposed using Hough Transform, edge type and priori information, in order to prune out all those features that do not belong to lane markings. The evaluation of the proposed method of lane marking extraction is then presented on over 6,000 image frames, captured in various complex scenarios such as tree shadows and presence of other artifacts resembling lane markings. A temporal history analysis was then introduced to further improve robustness of the lane marking detection method, and to also generate the lane RoI in cases where the near view does not contain lane markings. Next, techniques to distinguish between solid and dashed lane markings are proposed using spatial and temporal continuity analysis. The BLM detect module is then deployed to detect horizontal linear markings.
4. In Chapter 4, an architecture for detecting BLMs and distinguishing them is presented, bringing together the various computational blocks that were proposed in the previous chapter. The Hough Transform (HT) is identified to be one of the major computationally demanding steps, thus setting the motivation to accelerate it. A novel parallel HT algorithm, based on the additive property of HT is proposed that leads to orders of magnitude speedup in the HT computations. A detailed discussion on the computational savings that are achieved using the proposed BLM detection method is presented.

5. In Chapter 5, techniques for detecting complex linear markings (CLM), including double linear markings and zigzags, are presented. Firstly, the challenges in detecting CLM are addressed, arising primarily due to the limitations of the conventional Hough peak detection. Next, an adaptive Hough peak detection (AHPD) method is proposed that is able to detect lines of close proximity and lines of varying lengths, without any redundant peak detection in an automated manner. The BLM detection module proposed in Chapter 3 is now improved using the AHPD, which is then deployed to detect double linear markings. The technique for detection of double linear markings is then integrated with the continuity analysis techniques proposed previously, in order to classify lane markings as single-solid, single-dashed, double-solid or double-dashed. The AHPD is then deployed to detect zigzag markings. Results for detecting double linear markings and zigzag markings are presented. Finally, a computational architecture that consolidates detection of all linear markings is presented, by cascading the core modules.
6. Chapter 6 starts with introducing the benefits of detecting arrow markings for advanced navigation. Then, the challenges in detecting road arrow markings are addressed, mainly covering the issue of scale and distortion invariance. Limitations of existing methods to tackle this problem are also outlined. Next, novel techniques to detect and classify arrows in road scenes are presented using scale invariant shape signatures. These are based on decomposition of the arrows into their component parts and the use of signed edge maps to uniquely define each arrow type. The validation of the method on the arrows dataset is then performed, and the misdetections are analysed. Next, detection of arrows in side lanes are addressed by relying on host lane arrows as reference. Results of arrow detection and classification in side lanes is presented next.
7. Chapter 7 presents techniques to detect pedestrian markings. Firstly, pedestrian markings as seen by the driver are described using fundamental edge features. Unique properties satisfied by the lines representing the pedestrian marking are identified, which form the basis of the pedestrian marking detection algorithm presented in this chapter. Next, the use of other road markings as indicators to predict pedestrian markings are explored, in order to make the detection more

1. Introduction

robust. The evaluation of the method is then presented. A discussion on scale-distortion invariance has been included. Finally, a unified block architecture integrating the various road marking detection modules has been presented.

8. Chapter 8 presents the conclusions of the research in relation to the objectives set in Chapter 1. Finally, recommendations for future work are presented.

2

Literature Survey

2.1 Introduction

The purpose of this chapter is to provide a review of the existing techniques and methodologies in the area of detection and classification of road markings for Advanced Driver Assistance Systems (ADAS). Starting with an overview of sensing technologies in ADAS, the feasibility of the various sensors for road marking detection is reviewed. Next, a comprehensive review of various existing techniques for detecting road markings such as lane markings, arrow markings and pedestrian crossings is undertaken. Their strengths and weaknesses are also outlined. Lastly, the underlying computational blocks commonly encountered for the purpose of road markings extraction are discussed.

2.2 Role of road markings in ADAS

Advanced Driver Assistance Systems (ADAS) have been predicted to be the future of vehicle control for driver safety and assistance [2] and has been one of the highest-growth segments in vehicle electronics [3]. News analysis from Frost and Sullivan [5] estimates that the market for ADAS will cross euro 143.0 million by 2015, triggered primarily by growing demand for safety and automation. In alignment with this projection, the recent past has witnessed an increasing number of automobile industries incorporating more and more ADAS features into their products. Existing ADAS market has a product mix of safety systems as well as comfort and convenience systems

2. Literature Survey

[6]. These include speed alert, lane support/blind spot detection, automated safe following, pedestrian detection, vision enhancement, driver attention monitoring etc. A closer look at any of these ADAS systems reveals that they typically involve sensing and extracting valuable information from the road scene, evaluating that information, and communicating to the driver about the road scene based on the evaluation outcome [7].

Road markings painted on the roads are one such critical information meant for traffic safety, navigation and regulation [8, 9, 10]. They are of various types, including lane markings, dashed and solid, stop lines, pedestrian crossings, arrows, zigzag markings etc [11]. Road markings are used to channel vehicle and pedestrian traffic, to define parking, nonparking, giveaway and loading areas [12]. They play an important role in preventing accidents by limiting pedestrian and/or vehicle conflicts and defining paths through hazardous areas. A recent study conducted by Texas Transportation Institute [13] in 2012 highlights the importance of having prominent road markings for improving traffic safety. The study reports that introducing prominent lane markings resulted in a dramatic reduction in total crashes by upto 30%, and fatal plus injury crashes by upto 38%. The study further reports that the installation of new chevron signs (a kind of road marking usually installed on curved roads) reduced fatal crashes by 16%. Road markings also help to segregate and direct traffic safely and efficiently in a variety of road situations, from high-speed trunk roads and high volume urban environments to unlit rural roads [12].

Since road markings contain valuable information related to road safety and navigation, several ADAS applications extract and analyze the different kinds of road markings in the road scene. This has led to research in detection of lane markings [14, 15, 16, 17, 18, 19], classification of lane markings [20, 21, 22], detection and classification of arrows [9, 17, 23, 24, 25, 26], detection of stop lines [27], pedestrian crossings [9, 28, 29], worded markings such as STOP [17] etc. One of the most researched areas within this domain is the detection of lane markings, which has been shown to be the basis for a wide range of ADAS applications such as lane departure warning [30, 31], automatic vehicle control [32, 33], driver alertness monitoring [34], vehicle collision avoidance [35, 36], lane change assistance [21, 37, 38, 39] etc. Not only that, classifying whether the lane markings are solid or dashed have potential applications in lane change assistance [21]. The authors in [40] express that recognizing from a certain

distance, the painted road objects, knowing the number of lanes and their type, can help the driver assistance system take the correct decision in changing or remaining on the same lane. This forms the motivation of their work in extracting information about the road ahead, treating each painted sign as an individual object, instead of detecting lanes as a whole. Popescu et al. [20] in their recent work on differentiating lane markings, point out that double or continuous line boundaries separate the directions of traffic; discontinuous boundaries separate lanes in the same direction, and merge-type markings (densely discontinuous markings) separate the road from road-side parking area, and are also present in intersections as direction guides.

Arrow markings indicate valid traffic flow directions in the lanes, and help to channel the traffic smoothly, thereby mitigating chaotic traffic or queues at junctions. They permit road users to drive on multiple lanes side by side in a more secure way since they define the possible driving directions [24]. Similarly, arrow markings have been identified to be useful in understanding the driving environment [17, 23], and to regulate the driving directions at upcoming intersections or junctions [9, 24]. Arrow markings detection can even be used for advanced road map generation [41] or even for more accurate autonomous guidance [42]. The need to detect arrow markings is further strengthened by the fact that the exclusive information provided by them, unlike information such as speed limits, which is also provided by traffic signs [17]. They advise road users on their choice of the appropriate lane at an early stage [24]. Thus, detecting the different arrow markings is another very valuable step towards building advanced driver assistance systems that enable safer and less cluttered roads, by assisting the driver to navigate into the correct lanes.

However, there is very limited work on detecting arrows that is available in literature [9, 17, 23, 24, 25, 26], and this problem of detecting and classifying road arrow markings has been recognized as an important problem only in recent years. Another critical road marking information, which if detected early can tremendously add to road safety, is pedestrian crossing [10]. According to a recent survey in the U.S [43], nearly six pedestrian accidents were occurring at marked pedestrian crossings for every one mishap in unmarked crosswalks. It would therefore be of value to have an ADAS that alerts the driver of any upcoming pedestrian crossings automatically, to reinforce the safety checks while navigating through pedestrian areas [9, 28, 29] and thereby improving road safety. However, most existing work on literature on detecting pedestrian crossings cater to

2. Literature Survey

assisting the blind [28, 29, 44], and these approaches cannot be directly deployed for ADAS applications.

Other road markings include zigzag lines, stop lines, different types of road boundaries like double lines, dashed lines etc. [11]. There is not much work done on developing systems that can detect such road markings, though they are equally important for road safety. For example, white zigzag lines contain information about approaching pedestrian crossings [11]. Hence detecting them can aid in giving an advance warning to the driver to watch out and drive more carefully. This can directly aid in ADAS systems on pedestrian and obstacle detection, especially in designated crossings. Similarly detecting stop line ensures that the driver slows down and stops, and this can avoid accidents with traffic in the orthogonal direction [27]. Detecting a double or single yellow zigzag marking aids the driver to ensure he abides by the regulations indicated by them, regarding no-stopping and hence regulate traffic flow, in those road stretches [11].

Therefore, the relevance of detecting road markings to benefit a number of ADAS applications has been clearly established in literature. We now present a survey of the different techniques and methodologies that are currently employed in detecting the various road markings, after first reviewing the underlying sensor technologies.

2.3 Sensing Technologies for Road Marking Detection

ADAS applications rely on a wide range of sensors such as monocular and stereo cameras, long and short range radars, sonars, lidars, infrared, ultrasound etc. to enable cars to monitor their surroundings [45]. The choice of sensors depends on the functional requirements of the application, and the sensors chosen largely define the capabilities of the system. Broadly, ADAS sensors can be broadly classified into two categories - active and passive sensors, depending on the kind of sensing involved [46]. Active sensors, such as radars [47, 48, 49, 50, 51], lidars [52, 53, 54, 55, 56] and sonars [57] are based on reflection of electromagnetic waves. The reflected waves are studied to gauge the depth of the obstructions, and therefore such sensors are ideal for detecting 3-D objects. For example, in [47, 49, 50, 58, 59, 60], vehicle detection and tracking is done using radar sensors. Similar use of radars for detecting guardrails can be found in [48, 51]. In [56], lidar is used as a forward object sensor to recognise the environment

2.3 Sensing Technologies for Road Marking Detection

of the road. Lidars are used for vehicle detection in [52, 53, 54, 55, 56, 61] and for pedestrian detection in [54]. Active sensors are also known to work well in adverse weather conditions such as fog, poor lighting etc [46].

However, very few attempts have been made to detect road markings using active sensor technologies like radars and lidars [62, 63]. This is because active sensors work well in applications that involve detection of 3-D obstacles that can reflect the waves from the active sensors, unlike flat road markings painted on the road surface, that lack 3-D depth information. In [62], Schubert et al. investigated the feasibility of estimating the course of lanes using an automotive radar sensor, only. For that purpose, a data fusion approach was proposed which processes stationary radar detections from the road border as well as tracked vehicles. However, the authors point out that this system does not work if no guard rails are present for a long time, relies on prior knowledge of the lane widths and number of lanes, and assumes that vehicles are always driving in the middle of their lanes. In [63], Lindner et al. present an approach to detect lane markings using an active light detection and ranging device (lidar). A polar lane detector grid is used to combine the distance and intensity measurement channel of the lidar sensor. However, through such approaches, only the high reflective lane marks can be reliably detected, thus indicating the need to depend on vision-based sensing for a robust detection of road markings. According to [32], laser and radar sensors are useful in rural areas for helping resolve road boundaries but they fail on roads with multiple lanes. McCall et al. in [32] also discuss how a combination of both radar and vision sensors could be useful for lane detection. A multi-sensor fusion approach for lane detection has been presented in [64], where radar based techniques are proposed to detect the pavements, which are inturn used as indicators to detect the lane markings using vision sensors. However, the use of depth sensors alone have been identified as not being feasible for detecting road markings [62, 63].

The second category of sensors called passive sensors have been shown to be more suitable to detect road markings [32, 46, 65]. They rely on a non-intrusive approach and capture the scene without the use of electromagnetic waves, as in the case of active sensing. Examples of passive sensors are infra-red cameras, vision-based sensors or cameras, stereo cameras etc. Use of vision based cameras for road marking detection has been widely studied [15, 16, 17, 23, 28, 66, 67, 68, 69, 70, 71]. In [18, 66, 72, 73] etc., lane markings are detected and tracked using a single forward looking camera.

2. Literature Survey

In [36], lane markings are detected using a stereo vision system. In [23], monocular cameras are used for detection and recognition of road arrow markings. In [20, 22, 74], camera sensors are used for classifying lane boundary markings into various categories. [9, 28, 29] are attempts to detect pedestrian crossings using a monocular camera sensor and [27] used cameras for detecting stop lines at intersections. An integrated vision-based approach to detect various kinds of road markings has been proposed in [17], and other camera-based methods to detect and classify different kinds of road markings can be found in [9, 75].

In summary, it was seen that monocular vision modality i.e. cameras are the most frequently used sensors for detection and classification of road markings [65], because of their ability to capture the whole scene ahead, as seen by the human eye. Road markings are designed keeping the human visual system in mind, hence using a camera to get the same visual cues for a computational equivalent system makes great sense [65]. Further, from an evolutionary viewpoint, there is an increasing trend towards the use of cameras sensors for ADAS applications [5, 32, 46], owing to decreasing costs of CMOS cameras as compared to expensive active sensors [46], and also their non-intrusive nature. According to [32], vision sensors can be used in a wide variety of ways as compared to active sensors, and this versatility of vision sensors is an added reason for their popularity. Below, we survey the existing vision-based algorithms towards road marking detection.

2.4 Survey of Algorithms for Road Marking Detection

One of the main observations from existing literature on this subject is that, road marking detection appears to be largely confined to detecting lane markings. Infact in many references, the word ‘road marking’ has been used to convey ‘lane marking’, and many of the other markings, such as zigzag markings, double line markings, arrow markings etc. have been ignored. Quoting from recent literature [9]: “Only a few of systems incorporate road marking recognition. This explains why this problem was drastically less studied than lane markings detection. Indeed, the detection and recognition of all kinds of markings should provide future ADAS with a more comprehensive knowledge of the vehicles environment”. In another recent publication [76], it is mentioned that “To our knowledge, very few researches have concerned other kinds of special road

2.4 Survey of Algorithms for Road Marking Detection

markings (e.g. pictograms, text, arrows, zebras)”. Although lane markings are the most common markings and form an important subset of road markings, one cannot ignore the importance of other kinds of road markings. This research work will address the detection and classification of various kinds of road markings. Tao et al. in [17] mention - “While there is extensive work in computer vision focused on detection of traffic signs, the corresponding literature for road marking detection is rather sparse. Indeed, even large databases of road markings for evaluation are hard to come by, with a single exception [22], which however does not contain labels for the types of road markings”.

Existing work on lane markings detection is summarized below, followed by the limited available work on detection of other markings such as arrows, pedestrian crossings etc.

2.4.1 Lane Marking Detection

Lane detection has been one of the areas that has caught tremendous attention from researchers in the past decade, mainly because of its relevance in a number of ADAS applications such as lane departure warning [18, 31, 77], driver-attention monitoring [34, 78, 79] or automated cruise control [35, 80, 81]. One of the most crucial steps in lane detection is the **extraction of lane markings** [32, 65, 82]. Lane marking detection methods broadly consist of two main computational stages (a) feature extraction and (b) outlier removal, which are performed in accordance with a chosen lane model [32]. The initial candidate set of features are selected in the edge detection step, and are further filtered in the outlier removal step, to retain only those satisfying the lane model. Sometimes, a preprocessing step is also explicitly considered prior to the feature extraction stage [65, 82]. The lane model chosen influences the steps in the feature extraction stage and outlier removal stage.

2.4.1.1 Feature Extraction

Lane feature extraction can be either edge-based [14, 16, 19, 31, 70, 83, 84, 85, 86, 87, 88, 89], frequency-based [66, 66, 90, 91, 92, 93] or intensity/color-based [15, 86, 87, 94, 95].

2. Literature Survey

Edge-based lane feature extraction: Edge and gradient information are some of one of the most commonly used metrics for lane feature extraction. An edge may be regarded as a boundary between two dissimilar regions in an image [96], or a discontinuity in image intensities [97]. Discontinuities in image brightness are likely to correspond to discontinuities in depth, discontinuities in surface orientation, changes in material properties and variations in scene illumination [97]. Ideally, applying an edge detector on a road image with prominent lane markings leads to a set of candidate points representing the boundaries of the lane markings. One of the the common approaches of edge detection, by thresholding intensity gradients using the Sobel kernel [97] has been used for extracting lane edge features in [16, 84, 87, 88]. In [85], a modified algorithm based on the global edge detector [98] is used, after a 1-D Gaussian smoothing pre-processing step, to extract lane features. The global edge detector is based on the difference among the pixels in a 3×3 mask. In another of the edge-based lane extraction approaches by Otsuka et al. [14], directional edge based techniques are used for detecting straight lane markers and raised pavement markers. The algorithm selects only those edge points that are pointed towards the vanishing point, by first computing edge direction using near-neighbor elements. The authors in [70], make use of intensity variations along a particular direction using multiscale analysis in a hierarchical manner. A dynamic edge detection threshold is also proposed using this multiscale analysis. In [89], fusion of edge information with intensities is performed to separate lane boundaries from the road surface, using a priori knowledge that the lane marks painted in one of three colors of white, yellow, and even blue are brighter than the road surface. In [87], a lane feature extraction algorithm is proposed that combines intensity, edge and width constraints. Points that are above a certain intensity threshold and edge threshold, also satisfying other criteria such as width are chosen as potential lane marking candidates.

In general, edge based techniques work well with prominent lane edges but they can pose to be an issue in cases of highly cluttered shadows from trees obscuring lane markings or in cases of low contrast of road markings [32]. Low contrast of markings could happen in scenarios where illumination is poor, either due to cloudy weather condition or evening/night times, or where the markings are worn away due to passage of time [31]. Sometimes, the road markings exhibit strong edges, but are surrounded by other features that also exhibit edges, such as tree shadows, shadows and edges of vehicles, shadows from overhead fixtures, and other line features formed due to change

2.4 Survey of Algorithms for Road Marking Detection

of tar color etc. In such cases, it is challenging to extract out just the edges of the road markings [16]. In [14], in order to pick up edges of low contrast, the threshold is set to low values, as a result of which high number of noisy edge pixels are also picked up. The authors claim to deal with the noise by selecting only those edge points which are in the direction of vanishing point. However, this approach of noise elimination is a local pixel-level operation and as a result, it can be expected that noise from shadows and other unwanted features which also have similar edge directions will also be picked up. In another edge-based approach by P. Linder et al. [83], features are extracted at multiple levels for geometric lane feature extraction. Edges generated using the Canny edge detector are used to construct line segments, parallel lines, curves and then detect lane markings. As pointed out by the authors, the effectiveness of this method is dependent on the image and the Canny edge map generated. It is a well known fact that while Canny edge detector gives good results for faint edges, it has a tendency to clutter the edge map with noise heavily [97]. Hence it is expected that the accuracy of lane marking detection from this method suffers in the presence of shadows, noise and other lane-like features [83]. Lee et al. [31] proposed an edge-based lane extraction method for the application of lane departure detection. The method employs histograms of edge magnitudes with respect to edge orientation angle, referred to as edge distribution function (EDF), in a recursive manner. The EDF enables a correlation between the edge-related information and the lane-related information. Examining the EDF by the shape parameters of the local maxima and the symmetry axis results in identifying whether a change in the traveling direction of a vehicle has occurred. However, noise from pavements, edges of long vehicles etc., which have edges also running in a similar direction as lane markings, can affect the detection of peaks in EDF.

Frequency-based lane feature extraction: The other kind of lane feature extraction methods is based on operations in the frequency domain. Frequency-based techniques such as the Lane-finding in ANother domain (LANA) [90, 91, 92] have been shown to be effective in dealing with extraneous edges. The method is based on a set of frequency domain features that capture relevant information concerning the strength and orientation of spatial edges. The frequency domain features are combined with a deformable template prior, in order to detect the lane markers of interest. However,

2. Literature Survey

even this may be confused by complex shadowing [32]. Coskun et al. [93] use Gabor filters to narrow down lane markings after first performing a warp perspective mapping (WPM) [9, 23, 40] to get a top view. Zhou et al. [66] employ Gabor filters on image intensities to initially determine the vanishing point, followed by Canny edge detector along with Hough Transform to detect lane edges in the near view. In [66], Gabor Filter texture analysis is adopted to estimate orientation in each pixel. Then, the orientations in those pixels are used to determine lanes that are pointed at a vanishing point.

Intensity-based lane feature extraction: Lane feature extraction based on image intensity and color information is the next class of approaches. In a color-based method proposed in [94], lane markings are considered to be high contrast color features against the road surface. They propose to use image morphology based lane marking extraction. In [15], the input images are processed by a horizontal bisector method to find mid-points of potential lane markings, and these are taken as the basic data points for line detection. In [95] intensity (brightness) information is used after a Gaussian smoothing preprocessing step. On the filtered image, a peak finding algorithm is applied wherein a range of values that increase first and then decrease without any internal ripples are selected as lane candidates. In another intensity-based feature extraction method, a template matching algorithm is employed [99], wherein a Cross Correlation (NCC) [100] operation is performed with a template that resembles a common lane marker on the road.

2.4.1.2 Outlier Removal using Lane Models

Once features are extracted, lane models are used to select only those features that fit into the model, and reject the outliers [32, 65]. Common lane models used are straight line model in image domain, typically used in conjunction with the Hough Transform [8, 16, 72, 73, 77, 84, 86, 89, 99, 101, 102] or RANSAC [15, 103], deformable contours [104, 105], parallel lines in inverse perspective warping [9, 23, 40] and piecewise constant curvatures [73, 106].

Straight line model and Hough Transform: One of the most common lane models is the linear model, where lane markings are considered to be conforming to straight lines at predictable angle ranges, converging into the vanishing point [73, 82, 107]. The

2.4 Survey of Algorithms for Road Marking Detection

Linear Hough Transform [108] (HT) is one of the most widely used methods to select the straight lane edges and remove outliers [8, 72, 86, 99, 102], and is based on the parametric representation of lines passing through a point in the image plane. In [109], the Hough transform is applied on oblique image edges, after enhancing the edges using a median filter, to select the lane features, and reject outliers. In [89], after the candidate edge pixels are extracted using a boundary pixel extractor with six constraints based on edge and intensity information, the Hough Transform is applied to detect the lanes. In this approach, the region of interest is divided into four sub-regions to increase robustness, and the lines in each of these sub-regions are considered separately for the HT processing. In [110], lane marking features are first extracted from images captured by two downward side cameras by edge pair detection. Hough Transform is then applied on the candidate points to get the lane positions. In [107], Hough Transform is used for detection and tracking lane contours. Geometric constraints about lane contours being parallel and lying on a plane ground are imposed, to simplify the peak detection step of Hough Transform. In [8], a lane marking tracking method using Hough Transform and Kalman Filtering is presented. In [84] and [16], Hough Transform is applied in block based fashion and these outputs at the block level are analysed in a hierarchical manner in [84] to extract prominent lane features in the road scene. Randomized Hough transform [111], which is an extension of Hough transform, has also been used for the purpose of lane marking detection [72, 99]. In a Randomized HT, a set of n pixels are randomly selected from the edge image for determining the n parameters of the curve. In [99], a template matching based [100] feature extraction is first performed to locate potential lane marking points. This output is binarized, and the Randomized Hough Transform is applied on the binary image to locate a collection of straight lines representing the lane marking positions. In [72], the Randomized Hough Transform is applied on the edge map generated by thresholding the Sobel gradients. The threshold for edge detection is dynamically selected by performing a statistical analysis on the sampled pixel intensity values using a histogram of the gradient magnitudes.

One of the main advantages of using the Hough Transform for lane detection is its robustness to partial occlusions and discontinuities in edges [72]. However, one of the limitations of directly applying the linear Hough Transform model on the entire image is that it works well only for straight lane cases, and would yield a weak detection precision when applied to curved lane scenarios [82]. To overcome this problem, many

2. Literature Survey

researchers have proposed to break the road region into subregions and treat them separately [66, 73], considering that lane behavior can be modelled as straight lines in certain regions. In [66], the image is divided into two regions - near view and far view, and the Hough Transform is applied on the edge map of the near view, created using the Canny edge detector, to determine lane edges in the near view. In [73], the image is divided into near and far regions and a linear approximation of the curved lanes is done. Hough Transform is applied in the near region after a *k-means* clustering to reduce errors. For detecting lane markings that are further away in the far view, a piecewise linear approach with ‘step walking’ has been employed [73]. Another work that divides the image into near and far regions is presented in [102], where Hough transform is applied in the near view and linear parabolic fitting in the far view. The underlying basis for this is that, even curved roads exhibit a linear geometry in the near view, hence it is safe to assume a linear model in the near view. In [106], the authors divide the lower half of the image contour into five sections of decreasing width from the bottom to the top, and then apply the Hough transform separately in each section. Meuter et al. [88] uses a fast radial symmetry algorithm [112], which is an extension to the Hough Transform for the outlier removal step. The radial symmetry algorithm takes the gradient direction into account to determine the reference point in the parameter space. For each pixel that contains a strong gradient, a line perpendicular to the gradient direction is calculated. The intersection point of this perpendicular line with the voteline is used as voting target [88].

Another issue with application of HT is the high computational costs involved, as the HT computation is intensive in trigonometric operations [113]. Conventionally, it treats all edge pixels in the edge map equally and processes all of them, resulting in high computation complexity. A number of techniques have been proposed to optimize the HT computation [111, 114, 115, 117, 118, 119, 120, 121]. One of the ways to reduce the complexity of HT computations is by reducing the number of edge pixels on which Equation (3.4) is applied to. Randomized HT (RHT) [111] and probabilistic HT (PHT) [119] compute the LHT on a reduced set of edgels by either random or probabilistic selection of edgels respectively. A further improvement to PHT is progressive PHT (PPHT) [120], where once a peak in the Hough space crosses a threshold, i.e. when a line is detected, the voting process for that line is terminated. That line is traced and all edgels supporting it are marked as ‘voted’ even if they are yet to vote and removed from

the set of edgels whose voting is pending. However, the above methods do not keep in mind the shape of the feature of interest (in this case, linear features), and the edge pixels are sampled randomly throughout the image. This not only leads to a reduction in the number of required edge pixels actually corresponding to features of interest, but also does not eliminate noisy pixels belonging to features not of interest. Gradient angle of edge pixels has been used to limit the number of HT computations [118, 121]. In O’Gorman’s HT [118], the HT is applied on the coordinates of every edgel for the gradient angle only and not for all the angles in the Hough space. Gradient information was used in [121] to further reduce the complexity of PPHT, wherein gradient angles are used to determine the direction of edges to trace the lines faster in PPHT. O’Gorman’s HT [118] reduces the number of angles for Hough voting.

Another straight line model is employed in [94], wherein a line fitting is applied after performing a thinning operation based on image morphology. The whole lane is reproduced using Markov modeling and the proposed lane segment linking algorithm. However, since this approach relies on image contrast, and its efficiency in cases of faint markings is questionable, where contrast is not significant. In summary, the straight line model is a reliable model under the assumption that the vehicle is in a zone of straight lanes, or for detecting lane markings in the near view, where the markings show predictable properties [73].

Higher degree curve fitting: Curved lanes and lanes of more complex shapes compared to straight lines, can be modelled with curves of higher orders, such as parabolas [102], clothoids [88] and deformable templates [68, 104, 122] based on chi-curves [122], splines [93, 104]. In [102], the lanes in far view are modelled using a parabolic curve fitting. [88] uses an approximation of the clothoid model, wherein the lane located on a flat ground-plane is represented as a second or third order polynomial. When lanes of more arbitrary shapes need to be considered, then deformable templates are shown to be suitable [82]. Deformable contours, also referred to as snakes, are curves defined within an image domain which can move under the influence of internal forces from the curve itself and external forces from the image data [105]. Once internal and external forces have been defined, the snake can detect the desired object boundaries (or other object features) within an image [104]. Kluge et al. [92] were amongst the first to use a deformable template model of lane structure to locate lane boundaries.

2. Literature Survey

The Metropolis algorithm is used to maximize a function which evaluates how well the image gradient data supports a given set of template deformation parameters. It has been pointed out in [123] that the deformable template based approach in [92] is time consuming because of the large amount of processing done on all pixels. More recently, Wang et al. [104] proposed a lane detection and tracking algorithm using deformable contours, specifically B-snakes. They propose the use of B-snakes for lane detection using the argument that, compared to other deformable models, B-snakes can describe much wider range of lane shapes because of their local controllability. For example, it can describe a more complex road shape, such as ‘S’ or a sharp corner turn, just by increasing the number of control points. In [68], an active contour based line model is proposed that uses linked control-lines on sub-regions of the image. Candidate lines are first extracted from the sub-regions using the Hough Transform and the most prominent line is found by the active contour based model that minimizes the snake energy [68]. In [93], a second derivative approach of Composite Bezier spline fitting method calculates the exact coordinates of the control points, and a score function based on pixel matching was used for determining lanes in curved roads.

The use of deformable templates has been shown to be effective in detecting lane marking edges in challenging scenarios [92], such as those with shadows, road skid marks etc., that contribute to clutter in the image. Also, straight line models, deformable models can detect more complex curves of the lanes, as they have a certain degree of freedom in the evolution compared to parametric models [82]. However, the underlying computations involved are more complex as well. And, deformable templates are pointed out to be computationally intensive, involve costly optimization steps [124] and are time consuming [123].

Inverse perspective mapping: One of the other road models used for the purpose of lane markings detection is to transform the image into its inverse perspective space through a process called Inverse Perspective Mapping (IPM) [125], and then identify parallel lines in this space to detect lanes [9, 36, 40, 70, 86, 93]. The angle of view under which a scene is acquired and the distance of the objects from the camera (namely the perspective effect) contribute to associate a different information content to each pixel of an image [126]. This perspective effect poses a limitation in application of a uniform filters to capture these markings in the whole image, and this becomes

2.4 Survey of Algorithms for Road Marking Detection

specially a constraint for Single Instruction Multiple Data (SIMD) implementations, which consider the image as a mere collection of pixels, independent of their semantic content [36]. To cope with such problems, [36] applied a geometrical transform called Inverse Perspective Mapping (IPM), proposed originally by Mallot et al. [125], using which the perspective effect from the acquired image can be removed, and the image can be remapped into a new 2-dimensional domain where the information content is homogeneously distributed among all pixels, thereby enabling processing in the SIMD paradigm.

Most of the times, IPM is done prior to the lane feature extraction step [93], while sometimes it is performed at the end, after lane detection. In [93], an inverse perspective mapping is first performed, to get the top view of the image, prior to apply Gabor filters and curve fitting for outlier removal. [36] performs IPM first, followed by extraction of lane features marked by bright lines with uniform width in the transformed domain. In [86], the IPM is applied after Hough Peak detection is performed, in order to get more properties of the lane marking, such as position and rotation. In [70], a stereo-vision based approach using IPM is applied for lane marking detection. First, IPM is applied on the stereo image pair to remove the perspective effect, and the mapped images are subject to a multi-scale lane feature detection step. In [9], lane markings are identified after first performing an IPM, by detecting objects as connected components and analysis their shape. In [40], a template matching based method for feature extraction is done on the image, after first removing the perspective effect using IPM.

One of the criticisms that the IPM faces, however, is that this transform involves a number of assumptions, such as the flatness of the road, which in reality result in losses of accuracy and in an eventual degeneration of the performance [127]. For example, as stated in [36], the authors used the assumption of a flat road and other apriori knowledge, such as the camera position, orientation etc. to work out the new pixel position of each pixel in the image. Another issue with the inverse perspective mapping is that the underlying mathematical equations involve an extensive usage of trigonometric operations and division operations. Considering that this mapping needs to be performed for every pixel on the image, the amount of processing required for each frame is significantly high. For example, the method in [70] is shown to achieve a performance of 5 frames per second when run on a Xeon 2.6 GHz processor.

2. Literature Survey

RANSAC: RANSAC, short for RANdom SAmple Consensus [128], is another approach that has been employed to filter out non-lane features and select those conforming to lane properties [15, 103]. RANSAC is an iterative method to estimate parameters of a mathematical model from a set of observed data which contains outliers. It works on the assumption that data consist of inliers and outliers, where inliers refer to data whose distribution can be modelled, while outliers refer to data that do not fit into the model. Outliers can result either due to incorrect interpretation of data, or due to incorrect measurements, or due to noise [128]. In [15], the input images are first processed by a horizontal bisector method to find mid-points of potential lane markings. With these as basic data points, RANSAC is applied to detect line segments that correspond to lanes. In [103], lane marking detection is first performed using artificial neural networks and support vector machine, by taking 9×3 window samples of intensity values, following which RANSAC is applied to discard outliers and detect lane boundaries.

One of the main advantages of the RANSAC is its robustness in estimating the model, even when the number of outliers present in the data set are significantly high [128]. However, one of the issues with RANSAC is that there is no upper bound on the time it takes to compute these parameters [134]. Limiting the number of iterations does not guarantee a satisfactory or optimal result. Since increasing the number of iterations betters the accuracy, usually RANSAC is run on a high number of iterations to attempt reasonably good results. The implication of this is ofcourse the high computational costs that are involved in the process. The RANSAC-based method in [15] was implemented on Pentium 4 processors, and as such, it is not feasible to directly run such an algorithm on constraint-driven embedded platforms. Another disadvantage of RANSAC is that it requires the setting of problem-specific thresholds [134]. Also, RANSAC can only estimate one model for a particular data set. As for any one-model approach when two (or more) model instances exist, RANSAC may fail to find either one. In such cases, where more than one more instance is present, the Hough transform is shown to be an alternative robust estimation technique [134].

Temporal tracking: In applications such as lane departure warning, driver attention monitoring etc., where the lane parameters need to be tracked over time, the lane detection outputs are subject to a temporal tracking [8, 32, 70, 88, 129], in order

2.4 Survey of Algorithms for Road Marking Detection

to remove outliers across frames. Two of the most common tracking approaches are Kalman filtering [8, 88] and particle filtering [70, 129]. The tracking is done usually in a closed-loop feedback with the feature extraction, so that the tracked lane position defines an a priori estimate of the location and orientation of the extracted features [32]. In this step, tracking algorithms are used to predict the lane position using the history information of previously detected lane markings [21, 32]. By predicting the possible position of lanes in the current frame, the computation cost is reduced, as each frame is not processed independent of its history. Next, the predicted value is used to verify the measured lane position in the current frame. This improves the robustness, as outliers across time frames can be easily located and eliminated.

2.4.1.3 Classification of lane boundary markings

While most work on lane detection treat the entire lane as a single object defined by various geometrical models, few researchers have approaches this problem differently, by treating each road marking as a separate object and classifying them [40]. Attempts to classify lane boundary markings such as solid, dashed, circular reflectors etc. have been made in [20, 22, 32, 40, 42, 74].

A frequency analysis method to differentiate lane boundary markings as solid or dashed or merge was first proposed in [22], using a Fourier analysis. Another frequency analysis based approach was proposed in [74] by Gavrilovic et al. Here, a method to detect repeating markings was proposed, using median filtering, Discrete Fourier Transform and RANSAC, and this method can be applied to classify the lane boundary markings as solid or dashed. In another work on classification of lane markings by Stefan et al. [42], equally spaced scan lines projected in the image space were used. A scanline is defined by them a straight line orthogonal to the painted road marking and it is represented in 3d-world coordinates in the vehicle coordinate system. First, the lines are sampled using scanlines, and each scanline is classified as 0 or 1, in order to extract the type of line it represents. Then, using the successors and predecessors, scanlines are concatenated and outliers are removed. Finally, the series of scanlines is analyzed and the type of line (none, solid or dashed) is determined. Recently, another method based on scanlines to differentiate dashed and solid lane markings was presented by Voichita et al. [20]. Frequency analysis of the gray level profile of the feature boundaries is used, and the method assumes that the lane geometry is already estimated using a particle

2. Literature Survey

filter based lane tracker [129]. The binary signal encoding the relation between the average grayscale values of the markings and the asphalt is the primary feature of their boundary classification algorithm. Temporal integration and filtering is performed on the features extracted and the shape of the output signal is analysed using a period histogram, to classify the markings as dashed or solid. In order to differentiate painted lane markings from circular reflectors bounding the lanes, McCall et al. in [32] proposed the use of steerable filters based on second derivatives of two-dimensional Gaussians. A steerable filter [130] is an orientation-selective convolution kernel used for image enhancement and feature extraction that can be expressed via a linear combination of a small set of rotated versions of itself. Steerable filters have many desirable qualities that make them suitable for extraction and classification of road markings. Firstly, they can be created to be separable, which can help to accelerate the processing [32]. Secondly, a finite number of rotation angles for a specific steerable filter are needed to form a basis set of all angles of that steerable filter [130]. This allows one to see the response of a filter at a given angle, and therefore, to tune the filter to specific lane angles or look at all angles at once. This property is exploited in [32], considering circular reflectors have high responses in all directions, while line markings have high responses in a single direction. However, none of the methods above address the issue of computational costs and robustness in complex scenarios.

2.4.2 Arrow Markings Detection

Compared to the substantial work that has been done and reported on detection of lane markings using on-board cameras, there is very limited work on detecting arrows that is available in literature, as pointed out in [9, 24]. Also, this problem of detecting and classifying road arrow markings has been recognized as an important problem only in recent years, considering that the limited published work in this topic is confined to the last five years. Existing arrow detection techniques employ algorithms like inverse perspective mapping (IPM) [9, 17, 23, 40, 131] of input images, connected components [9, 24], wavelets [23], curve/spline fitting models [24] and geometrical template matching [17, 40, 42].

Chira et al. [40] use a graphical programming language called LabVIEW for detection, measurement and classification of painted objects on the road, including arrow

2.4 Survey of Algorithms for Road Marking Detection

markings. Here, the acquired image is transformed using the inverse perspective mapping, then a geometric pattern matching using templates is performed for locating lane markings and arrow markings. In [9] by Philippe et al., an algorithm was proposed to identify various road markings, including arrows, which is based on the following processing steps: marking pixel extraction, detection using connected components before inverse perspective mapping, and recognition based on pattern matching. Template matching is either performed with single patterns as in the case of arrows or repetitive rectangular patterns, as in the case of pedestrian crossings. In another method on detection of road arrow markings by Nan et al. [23], an inverse perspective was first done to remove the perspective effect, an improved Haar wavelet feature extraction is performed to extract potential features, after which recognition of arrows is performed using a multi-class Support Vector Machine (SVM). In another work on road markings extraction [132], road area is first segmented out to reduce the search space, features are extracted using brightness criteria, using a line-by-line scan of the image, and then road markings such as arrows are extracted using polygonal representation of the markings. However, this method is too slow to use in real-time [17]. Maier et al. [24] presented a general geometric approach to arrow markings detection using curve based prototype fitting. The region of interest is first narrowed down using preliminary knowledge of lane positions, to reduce the search space. Connected components of high contrast points are extracted as potential arrow features, and these extracted features are analyzed using prototype fitting models like splines etc. Classification of arrows is done using measures like Hausdorff distance. In another arrow detection method proposed by Vacek et al. [42], first, the region within the image where the arrow marking is expected, is determined using lane positions. A segmentation of this region is done in order to extract the arrow. The bounding box of the extracted arrow is then used for template matching with known arrows. However, this method uses the horizontal extent of markings at various distances to classify them, requires a lot of hand-coding for each type of road marking and does not scale well to a large number of marking types [17]. In another pattern recognition based approach to detect arrow markings and rectilinear markings by Rebut et al. [131], mathematical morphology is employed on the image transformed using IPM, taking into account geometrical characteristics such as size. Hough Transform is used for extracting white lines corresponding to rectilinear objects, Fourier descriptors are used for describing the boundaries of an object,

2. Literature Survey

and KNN classifier [133] recognizes the object from a training base built with simulated data.

One of the recent methods on detection and classification of various road markings, including arrow markings, was proposed by Tao et al. [17]. According this approach, the image is first subject to inverse perspective mapping [125], to reduce the perspective distortions. As with other IPM-based lane detection methods enlisted in 2.4.1, the ground is assumed to be flat, and hence not yielding perfectly parallel lane lines in the birds-eye view, when the vehicle is on hills or has pitch and roll movements. Next, to detect the region of interest, a Maximally Stable Extremal Regions (MSERs) algorithm [135] is applied on the rectified images. An MSER is connected region that is extremal in the sense that the pixels inside the regions are always brighter or darker than the pixels on the boundary [135]. Next, a set of feature points is extracted from the regions of interest using the FAST corner detector, originally proposed by Rosten et al. [136], which has been reported to be faster than both SIFT and Harris corner detectors [17]. Following this step, a histogram of oriented gradients (HOG) descriptor [137] is computed for each point of interest, for a fixed number of scales and orientations. Next a template matching algorithm is applied in two stages, first putative matching pairs of POIs based on their feature vectors are found, and second, the results are refined through a structural matching algorithm that matches the 2D geometry of the points of interest within the road marking. Although the method in [17] is meant to be a general framework to detect various kinds of road markings, arrow markings form one of the major datasets that this method was applied on.

One of the issues with template matching based approaches to arrow detection [17, 40, 42] is the high computational complexity, as with any template matching method. High computational costs associated with template matching is well recognized in literature, and several techniques to reduce the underlying computational costs of template matching have been proposed [100, 138, 139, 140, 141, 142]. It has been reported in [113] that the popular template matching methods such as Normalized Cross Correlation operation does not meet speed requirements for many applications. Further, this cost is significantly higher when the scale of the arrow markings and viewpoint of the camera are unknown. Lowe et al. [143] point out that template matching based approaches are shown to be effective in known environments, where parameters such as illumination, pose of the object are known to be fixed. However, in environments where

rotation, scale, illumination and pose are variable parameters, template matching turns out to be computationally infeasible [143]. For a scale-invariant template matching, the whole template matching process needs to be repeated for templates of various scales, resulting in a significant number of underlying computations.

2.4.3 Pedestrian Crossing Detection

There are a handful of techniques [17, 74, 75] that address the problem of detecting pedestrian crossings for ADAS applications. In [74], a method for the detection and recognition of repeating markings such as hatched areas, pedestrian-crossing, chevrons and give-way markings is proposed. Several filters such as mean filter, median filter, opening filters were explored for feature extraction, and it was concluded that the median filter gives the best feature extraction results. This result is extended to extract larger markings (hatched markings, zebra-crossing, arrows). The features are then subject to filtering in the frequency domain using Discrete Fourier Transform. Following this, a connected components characterization is performed, followed by a spatial filtering and RANSAC to classify the various road markings. Moez et al. [75] proposed classification of pedestrian crossings, continuous or dashed lane markings, left, right and central road edge markings. The approach involves three levels of abstraction from the pixel to the object level through connected-components, and at each level, Markov Random Field models the a priori knowledge about object intrinsic features and object interactions. The method proposed by Tao et al. [17], that is based on template matching on the IPM-transformed image, was discussed as an approach to classify various arrow markings. The same method, has been reported to have been applied on a wider set of road markings such as pedestrian crossings, speed limit signs and worded markings like ‘STOP’.

There are a few other techniques on vision-based pedestrian crossing detection [28, 29, 44] that have been developed to cater to assist the blind. As part of an initiative to assist the partially sighted, Stephen et al. [28] detect pedestrian crossings using intensity variation information to isolate potential regions, Hough Transform for detecting parallel lines that form the crossing, along with RANSAC to remove outliers. Also, to differentiate pedestrian crossings from staircases, which also display a similar group of parallel edges, pose information using homography search, and vanishing points criteria are employed. In another attempt to aid the navigation of the blind,

2. Literature Survey

Uddin et al. in [29] use bipolarity features based on intensity distribution of white and black pixels, to first identify the region of pedestrian crossing. Then, the direction of the crossing is inferred based on optimization function that is derived based on the Fourier Transform of the intensity map. Only frontal crossing are picked up, rejecting the sided ones. Lastly, the crossing length is estimated using a camera projection model. A slight variation of this method has been presented by the same author in [44]. However, these techniques cannot be directly applied for an ADAS application to assist the driver, mainly because the views of the pedestrian marking are orthogonal to each other in the two applications.

2.5 Summary

After a thorough survey on existing literature on road marking detection, the following three observations were made:

1. Cameras are the most frequently used sensors for detection and classification of road markings [65], because of their ability to capture the whole scene ahead, as seen by the human eye. Depth information, as captured by active sensors, cannot be used as a criteria to separate road markings from the flat road surface, as in the case of obstacles. Road markings are designed keeping the human visual system in mind, hence using a camera to get the same visual cues for a computational equivalent system is most suitable [65]. Further, from an evolutionary viewpoint, there is an increasing trend towards the use of cameras sensors for ADAS applications [5, 32, 46], owing to decreasing costs of CMOS cameras as compared to expensive active sensors [46], and also their non-intrusive nature. According to [32], vision sensors can be used in a wide variety of ways as compared to active sensors, and this versatility of vision sensors is an added reason for their popularity.
2. Detection and classification of road markings is of tremendous value to a host of ADAS applications. Detecting lane markings has been shown to be the basis for a wide range of ADAS applications such as lane departure warning [30, 31], automatic vehicle control [32, 33], driver alertness monitoring [34], vehicle collision avoidance [35, 36], lane change assistance [21, 37, 38, 39] etc. Classifying whether

the lane markings are solid or dashed have a role to play in lane change assistance [21] and helps to separate the directions of traffic, and road from road-side parking area [20]. Extracting arrow markings have been shown to be useful for assisted navigation [17, 23], to regulate the driving directions at upcoming intersections or junctions [9, 24], advanced road map generation [41] or accurate autonomous navigation [42]. Detecting pedestrian markings have been shown to be useful for improving safety on roads [17, 74, 75].

3. However, there is very limited literature available today that tackles the problem of detecting and classifying various kinds of road markings, such as arrow markings, zigzag road markings on road boundaries, pedestrian crossings, lane markings of different types such as solid, dashed, single, double etc., as pointed out in [9, 17, 76]. While there is extensive work in computer vision focused on detection of traffic signs, the corresponding literature for road marking detection is rather sparse [17]. Even databases of road markings for evaluation are hard to come by [17]. Work in this area is confined to lane marking detection, which has been extensively researched in the past decade. Infact in many references, the word ‘road marking’ has been used to convey ‘lane marking’, and many of the other markings, such as zigzag markings, double line markings etc. have not been considered.
4. Inspite of the extensive literature on lane marking detection, robust extraction of lane markings in complex scenarios, such as presence of shadows and similar artifacts, is not a solved problem, and continues to be researched upon [70, 103]. Edge-based feature extraction techniques pose to be an issue in cases of highly cluttered shadows from trees obscuring lane markings or in cases of low contrast of road markings [32]. Low contrast of markings could happen in scenarios where illumination is poor, either due to cloudy weather condition or evening/night times, or where the markings are worn away due to passage of time [31]. Sometimes, the road markings exhibit strong edges, but are surrounded by other features that also exhibit edges, such as tree shadows, shadows and edges of vehicles, shadows from overhead fixtures, and other line features formed due to change of tar color etc. In such cases, it is challenging to extract out just the edges of the road markings [16]. In [14], an attempt is made to detect lane markings of very low contrast,

2. Literature Survey

but this method also faces the problem of high noise as a result. In another recent method [83], the effectiveness depends on the Canny edge map, which can get cluttered in the presence of noisy artifacts. Hence it is expected that the accuracy of lane marking detection from this method suffers in the presence of shadows, noise and other lane-like features [83]

5. Limited research addresses issues of reducing computational complexity of the algorithms employed, as a step towards real-time solutions. One of the most popular outlier removal methods for lane marking extraction, the Hough Transform (HT) [8, 72, 86, 99, 102], involves high computational costs, as its equations are intensive in trigonometric operations [113]. Also, conventionally, HT treats all edge pixels in the edge map equally and processes all of them, and this further contributes to the high computational complexity. Although a number of techniques have been proposed to optimize the HT computation in general [111, 114, 115, 117, 118, 119, 120, 121], few methods specifically address the context of optimizing HT for lane marking detection. Similar issues exist with outlier removal methods like RANSAC, which although known to be robust, is considered computationally very expensive [134]. Similarly, the commonly used Inverse Perspective Mapping (IPM) employs extensive use of trigonometric and division operations, and considering that this mapping needs to be performed for every pixel on the image, the amount of processing required for each frame is significantly high. Likewise, deformable templates are pointed out to be computationally intensive, involve costly optimization steps [124] and are time consuming [123].
6. Scale-invariance and its associated computational complexity seems to be a major issue for detecting markings that are shapes, such as arrows. Most of the existing arrow detection templates are based on template matching [17, 40, 42], and it has been recognized in literature that template matching has high computational costs involved [100, 138, 139, 140, 141, 142]. It has been reported in [113] that the popular template matching methods such as Normalized Cross Correlation operation does not meet speed requirements for many applications. Further, this cost is significantly higher when the scale of the arrow markings and viewpoint of the camera are unknown. Considering that arrow markings

can be of varying scales and can be captured from multiple viewpoints based on the position of the vehicle with respect to the lane, the arrow marking detection needs to be scale-invariant and viewpoint-invariant. For such a scale/viewpoint invariant matching, the whole template matching process needs to be repeated for templates of various scales and viewpoints, resulting in a significant number of underlying computations. Template matching based approaches are shown to be effective only in known environments, where parameters are known to be fixed [143].

In the following chapters of this thesis, the various categories of markings as described in Chapter 1, are tackled one by one - (a) basic linear markings (b) complex linear markings (c) arrow markings and (d) pedestrian markings. In the next chapter, basic linear markings (BLM) are considered. Techniques for detecting and classifying BLM are proposed, after which their computational complexity and robustness in various challenging scenarios are evaluated.

2. Literature Survey

3

Basic Linear Road Markings

3.1 Introduction

Basic linear road markings (BLM) refer to the class of road markings defined by a single straight line, painted either along (longitudinal) or across (transverse) the road, and can be either solid or dashed. The set of basic linear markings is illustrated in Fig. 3.1. The abbreviation of ‘BLM’ will be used throughout this thesis to refer to these basic linear markings.

The most common BLM are lane markings. As summarized at the end of the literature survey in Sec. 2.5 of Chapter 2, existing lane marking detection methods suffer from robustness issues in complex scenarios, and researchers have still been addressing this problem in recent years. Those techniques that aim to achieve robust detection usually have high computational costs. In this chapter, computationally efficient techniques are proposed, to detect lane markings in varying lighting, weather and shadow conditions, and in the presence of other similar artifacts. First, the underlying properties of the various BLM are studied. Based on these properties, a novel feature extraction step is proposed, that employs histograms of gradient angles (GAH) to generate customized edge maps containing straight lines (referred to as Straight Line Edge Map or SLEM). The computational benefits of using SLEM for feature extraction are demonstrated for lane feature extraction. Next, an efficient outlier removal method is presented that selects only pixels bounding the lane markings and rejects noisy artifacts, based on signed edges and Linear Hough Transform (HT). The integrated method using SLEM, signed edges and HT is then evaluated on an exhaustive

3. Basic Linear Road Markings

Figure 3.1: Basic linear road markings.

data set containing lane markings. A temporal history based analysis is then proposed to improve the detection accuracy. Next, techniques to distinguish between dashed and solid lane markings are proposed, followed by a method to detect horizontal markings.

For the rest of the thesis, the following terms and definitions are used:

- ‘Host vehicle’: The vehicle with a camera sensor mounted in it, with reference to which the road scene is defined. Video captured by the camera on the host vehicle serves as input for extracting road markings.
- ‘Host lane’: Refers to the lane in which the host vehicle is present at any given instance of time.

3.2 Properties of BLM

In order to detect and classify basic linear road markings, distinct properties exhibited by them that distinguish them from shadows, overhead fixtures, railings etc., are first observed and listed below from P1 to P5:

- *P1 - Straight edges*: Basic linear road markings, as the name suggests, are made of linear boundaries. Hence, linear edges form the primary features of interest for detecting them.
- *P2 - Predictable locations*: These markings occur at predictable locations in the image plane, and have predictable angles of orientation. For example lane markings bounding the host lane appear as slanting lines on either side of the host lane, pointing towards the vanishing point, while horizontal lines like stop lines appear across the host lane.
- *P3 - Opposite signed edge pair*: Each BLM is bound by a pair of linear edges of opposite nature - left edge, where the intensity transition is from the dark road surface to the relatively light colored lane marking, and right edge, where the intensity transition is from the light colored lane marking, back to the dark road surface.
- *P4 - Thickness criteria*: The thickness of the BLMs is in a predictable range, especially in the near view, where the distortions related to perspective effects are more controlled. This translates to a predictable distance between left and right edges in these markings.
- *P5 - Temporal continuity*: The position of the markings is expected to be consistent across consecutive frames in the video sequence, as it is assumed that there are no abrupt movements of the host vehicle in an urban road scenario. This is based on the well-established assumption criteria in literature [32] that the vehicle dynamics on urban roads are predictable and changes in the scene are gradual across time.

The above properties are used in a systematic manner to propose efficient BLM detection techniques. Firstly, the fact that BLMs are bound by straight lines (property *P1*) is used for proposing an efficient feature extractor, described below in Sec. 3.3.

3.3 Straight Line Edge Map (SLEM) for Feature Extraction

Straight lines form the primary feature set of BLM (property $P1$), and the feature extraction step for BLM detection involves extracting these line features efficiently. A novel approach for detecting linear edge features from any given image, and generating an edge map containing only straight edges (called Straight Line Edge Map or SLEM) is proposed in this section. The generation of SLEM is based on the property that edge pixels lying on a straight line exhibit similar gradient angles. In the proposed approach, gradient angle histograms (GAH) are constructed using gradient angles of all edge pixels in the image, i.e. pixels with gradient magnitudes higher than a threshold. The GAH is analyzed to select the pixels that qualify as part of straight lines, in the following manner. It is observed that edge pixels that are part of the same linear edge have similar gradient angles, and hence a straight line edge is reflected as a peak in the GAH. On the other hand, pixels on curved edges and noise would contain a wide range of gradient angles, resulting in the values getting dispersed across the bins in the GAH. In other words, it is very likely that a peak in the GAH would indicate the presence of a line in the image. In most cases, higher the peak in the GAH, longer is the line in the image (An exception to this is when multiple small parallel lines lead to a higher peak in the GAH compared to a single longer line, as illustrated in Sec. 3.3.1.1). Thus, by shortlisting frequently occurring gradient angles using the GAH and by selecting only those points from the original edge map that have these selected gradient angles, a modified edge map containing mainly the straight lines in the image can be generated.

Figure 3.2: GAH of different types of features (a) GAH of straight line feature (b) GAH of a curved feature (c) GAH of noise.

Fig. 3.2 shows the GAH for different feature types. It can be seen from Fig. 3.2 (a)

3.3 Straight Line Edge Map (SLEM) for Feature Extraction

Algorithm 1 Algorithm for SLEM generation

```

1: Input:  $m \times n$  sized image  $I$ 
2: Initialize GAH
3: for every  $p(x, y)$  in  $I$  do
4:   compute  $|\nabla I|$  based on (3.1)
5:   if  $|\nabla I| \geq T_{|\nabla I|}$  then
6:     compute  $\nabla_\theta I$  based on (3.2)
7:      $E := 1$ 
8:     Update  $GAH(\nabla_\theta I)$ 
9:   else
10:     $E := 0$ 
11:   end if
12: end for
13: apply  $T_{GA}$  on GAH to generate set  $G_f$ 
14: for every  $E(x, y)$  with  $E(x, y) = 1$  do
15:   if  $\nabla_\theta I(x, y) \in G_f$  then
16:     $SLEME_{SL}(x, y) := 1$ 
17:   else
18:     $SLEME_{SL}(x, y) := 0$ 
19:   end if
20: end for
21: Output:  $m \times n$  sized SLEM

```

that the signature of a straight line is given by high values confined to a small number of bins. A curve of similar length as the straight line (Fig. 3.2(b)) shows that the values in the GAH are spread out over a wider angle range and lower in height, as compared to that of the straight line. Noisy patterns, such as leaves of trees (Fig. 3.2 (c)) yield GAHs that are spread over the entire angle range. This is because of the randomness in gradient angle distribution, where the probability of all gradient angles occurring is similar. Also, the bin heights are much lower.

Algorithm 1 describes SLEM generation for an $m \times n$ sized image I . A gradient angle histogram (GAH) having bins from 0 to 180° , with a resolution of 1° is initialized. For every pixel $p(x, y)$ in I , the horizontal and vertical Sobel kernels, S_x and S_y [97] are applied to obtain signed horizontal and vertical gradient values $\nabla_y I$ and $\nabla_x I$, respectively. These are used to compute the gradient magnitudes $|\nabla I|$ of all the pixels in the image I using:

$$|\nabla I(x, y)| = \sqrt{|\nabla_x I(x, y)|^2 + |\nabla_y I(x, y)|^2} \quad (3.1)$$

where (x, y) are the indices of a pixel in I .

A pixel whose gradient magnitude is greater than a threshold $T_{|\nabla I|}$ is considered as an edge pixel. An initial edge map E is generated based on the pixels that satisfy this

3. Basic Linear Road Markings

threshold, by setting $E(x, y)$ to 1. Threshold $T_{|\nabla I|}$ was heuristically set to be 10% of the maximum gradient value for that image block. Given that the road markings are the most prominent features in the image, this threshold setting was seen to yield good edge maps E , which forms the basis for further processing to generate *SLEM*. Next, the gradient angle $\nabla_{\theta} I(x, y)$ is computed for each edge pixel in E using the equation shown in 3.2 below, following which the bin count corresponding to that particular $\nabla_{\theta} I(x, y)$ in the GAH is incremented.

$$\nabla_{\theta} I(x, y) = \tan^{-1} \left(\frac{\nabla_y I(x, y)}{\nabla_x I(x, y)} \right) \quad (3.2)$$

After reading through all the pixels in I , the GAH is thresholded using a threshold $T_{\nabla_{\theta} I}$ resulting in a set of frequently occurring gradient angles G_f . The straight line edge map SLEM is generated by choosing only those edge pixels from E whose gradient angle is a member of the set G_f . In other words, SLEM is a subset of the Sobel edge map E , containing mainly the straight edges in the image.

The proposed SLEM generation algorithm is illustrated on an image in Fig. 3.3 (a) whose original Sobel edge map is shown in (b). Its GAH that is computed for all points in the edge map is shown in (c). Fig. 3.3 (d) shows the SLEM generated by choosing only those edge pixels that have the selected gradient angles upon thresholding the GAH. It can be seen that this modified edge map has the straight line only and has filtered out the circle.

Figure 3.3: GAH of straight and non-straight pixels, (a) Image I (b) Edge map (c) GAH (d) SLEM of I .

3.3 Straight Line Edge Map (SLEM) for Feature Extraction

3.3.1 Block-based SLEM generation

The SLEM generation algorithm shown above is for an image I . However, using a single SLEM over the entire image was found to yield a large number of edge pixels that are not a part of the straight line features. This is because there is a loss of spatial information in a GAH, because of which its constituent peaks can be attributed to multiple features. To minimize the loss of spatial properties and to facilitate a more controlled analysis of the features, the image is divided into blocks of uniform dimensions, and the Algorithm 1 is applied separately to each block. The peaks can be more conclusively attributed to straight line segments within the blocks, leading to a more spatially localized operation. Further, the block level processing allows sensitivity to local changes within the image, such as illumination, brightness of marking etc. Also, block-based approach has potential benefits of parallel processing, whereby the SLEM of each block can be computed independent of the other.

The image of size $m \times n$ is divided into regular sub-image blocks using a $k \times l$ grid. A padding is done such that m and n are divisible by k and l respectively to facilitate division of the image into regular blocks. GAHs are computed for each block independently resulting in their corresponding SLEMs. The SLEMs of all blocks are concatenated to form the SLEM of the image.

3.3.1.1 Deriving block settings

The block based approach allows the SLEM generation to cater to varying lengths of lines of interest. In this section, the effect of size of the block defined by $b_{size} \times b_{size}$ and the length of the lines of interest, on the quality of SLEM, is studied. Quality of SLEM is defined by two factors: the presence of the lines of interest and absence of non-line edge pixels, treated as noise. In order to set the block size, L_{min} is defined as the minimum number of points that are required on any line in the SLEM. Any line that is smaller than L_{min} is to be discarded as noise and hence it is not to be reflected in the SLEM. As it was shown earlier, the selection of points for the SLEM is done based on the thresholded GAH. So, T_{GA} , the threshold to the GAH, is set to L_{min} .

The block size should be set such that the contribution of valid lines to the peaks in the GAH should be maximized and that of noise should be minimized. Having a very large block size compared to T_{GA} , i.e, $b_{size} \gg T_{GA}$ would allow noise and non-straight

3. Basic Linear Road Markings

line edges to be picked up. This is because large blocks would include a large number of edge features of varying gradient properties, thereby causing the entire GAH to be offset by a certain value. Thresholding this kind of histogram using a relatively small value would allow most of the edge points to pass through, resulting in an SLEM that will not be very different from the original Sobel edge map.

Figure 3.4: Effects of varying block size and T_{GA} , (a): Original Image of size 294×512 , (b): Sobel Edge Map with $T_{|\nabla I|} = 130$, (c): SLEM with $T_{GA} = 16$, $b_{size} \times b_{size} = 128 \times 128$, (d): SLEM with $T_{GA} = 16$, $b_{size} \times b_{size} = 16 \times 16$, (e): SLEM with $T_{GA} = 16$, $b_{size} \times b_{size} = 64 \times 64$.

On the other hand, if b_{size} becomes comparable or smaller than T_{GA} , then noise is removed effectively but the lines of interest may not be picked up. This is because the lines of interest can span across multiple blocks and each block may not have enough edge pixels to satisfy the high threshold values. These two extreme scenarios are illustrated on an image shown in Fig. 3.4 (a). The Sobel edge map is shown in Fig. 3.4 (b). Let us set $L_{min} = T_{GA} = 16$. Now, if a relatively large block size is chosen, say $b_{size} \times b_{size} = 128 \times 128$, the resultant SLEM is not very different from the Sobel edge map as shown in Fig. 3.4 (c). If the block size is to equal T_{GA} , i.e, $b_{size} = 16$, then we see that most of the lines of interest are not picked up (Fig. 3.4 (d)). For example, the main boundaries of the building are missing in the SLEM, which is not desirable. It is also interesting to note that, the smaller vertical pillar edges are picked up while the longer horizontal boundaries are not. This is because the smaller vertical lines are

3.3 Straight Line Edge Map (SLEM) for Feature Extraction

parallel, which have the same gradient angle resulting in a higher peak in the GAH.

It was found from experimental validation that a good compromise between the two extremes is to set the block size to be around 4 to 6 times the value of T_{GA} . The rationale for choosing this range of values for the block size was that, for these settings, not only were the lines of interest found to be prominent enough to be selected by the threshold, but the effect of non-straight line edges on the peaks was also kept under control. Fig. 3.4 (e) shows the SLEM generated for $b_{size} = 4 \times T_{GA} = 64$. It can be seen that the SLEM that is generated for these settings has cleaned the edge map of visible noise and also picked up collinear edge pixels falling on lines of interest. For example, the tree and circular lamp boundaries on the left of the image with no significant straight lines have been removed. A rectangular box on the left bottom of the SLEM is picked up, which was missed out for smaller block settings.

Figure 3.5: SLEM of test images; First Column: Test Image; Second Column: Sobel edge map; Third Column: SLEM.

Fig. 3.5 shows the SLEM generated for three test images: *Chimney*, *Keys* and *House* with block settings as described above, i.e. $b_{size} = 4 \times T_{GA}$. It is evident from all three images that the SLEM filters out visible non-straight line features. For example, the noisy edge pixels due to the foliage of the tree in *Chimney* is cleared in the SLEM. Similarly the elliptical lobes are removed in the SLEM of *Keys*. Additionally, depending

3. Basic Linear Road Markings

on the block size settings, small lines that are not of interest are also eliminated. This can be seen in the SLEM for *House* image.

3.4 Deploying SLEM for Lane Marking Feature Extraction

In this section, the above algorithm for SLEM generation is applied to images of road scenes containing lane markings. It will be shown that SLEM is an efficient step towards identifying potential lane marking candidates.

In order to set the region of interest (RoI) for detecting lane markings, only the patch of road surface immediately in front of the host vehicle, referred to as ‘near view’ is considered, which corresponds to the bottom portion of the image. The main rationale behind this is that lane markings are most clearly represented in the near view, considering their proximity to the host vehicle. Moreover, the lane model can be approximated to a straight lane model in the near view. This approach of relying on the *near view* of the host vehicle for lane detection using a straight lane model has been employed in [14, 31, 66, 73, 102].

Considering that the SLEM generation process is a block-based approach, the near view is divided into regular blocks B , and the SLEM for each image block B_i is independently computed, as per Algorithm 1.

Next, the angle range which defines the orientation of the lane markings in the near view (Property $P2$) is priori information that is known. So, the search for the peak in the GAH is restricted to the pre-defined angle ranges given by $\Delta_L = [\theta_{Ll}, \theta_{Lu}]$ and $\Delta_R = [\theta_{Rl}, \theta_{Ru}]$, where Δ_L and Δ_R are the angle ranges for left and right lanes respectively. Only those edge pixels from the image blocks, which give a peak in the expected angle ranges in GAH are selected. This will help to not only eliminate randomly distributed edge pixels such as those formed due to shadows, but also remove other linear features that are not aligned in the same direction as the lane.

Consider an image block B_i in the RoI. If a peak is detected at an angle θ_p in GAH_{B_i} such that θ_p is within Δ_L or Δ_R , the peak height n_{θ_p} gives the number of edge pixels having the gradient angle θ_p . The edge pixels in the neighboring bins of GAH, i.e., $\theta_p \pm \delta$ are considered to accommodate slight variations in the gradient angles of edge pixels along a line. If n_{θ_p} is greater than a minimum number of edge pixels,

3.4 Deploying SLEM for Lane Marking Feature Extraction

n_{th} , that should satisfy a straight line condition, we consider those edge pixels in the selected bins in GAH and create the straight line edge map E_{SL} . The same is repeated for all blocks in the region of interest.

Figure 3.6: Illustration of SLEM generation for lane markings (a) Near view containing lane markings and shadows, divided into blocks of size 90×128 (b) Image block chosen for illustration (c) Initial edge map with both lane marking edges and the shadow edges (noise), (d) Gradient angle histogram (GAH) for the input edge pixels of block, (e) Edge pixels E_{SL} of straight line edges corresponding to the peak centering -20° . (f) Summary of the underlying steps.

Fig. 3.6 illustrates the above processing steps on an image block (Fig. 3.6(b)) with a right lane marking. Fig. 3.6(c) is the initial edge map formed by thresholding the gradient magnitudes. It can be seen that this edge block has both lane marking edges and some noisy edge pixels that are not in the direction of the lanes. When the GAH is computed for these edge pixels, as shown in Fig. 3.6(d), two peaks are formed, one centering around $\theta_{p1} = 0^\circ$ and the second one at $\theta_{p2} = -20^\circ$. From the priori knowledge, it is known that the left lanes will occur in an angle range of $\Delta_R = [-40^\circ, -15^\circ]$. With this information at hand, the first peak centering at 0° is rejected and the second peak is selected because $\theta_{p2} \in \Delta_R$. Fig. 3.6(d) plots the edge pixels that have a gradient angle which form the second peak in GAH at 35° . The

3. Basic Linear Road Markings

noisy edge pixels that are not a part lane edges are eliminated and the edge pixels that appear to be a part of lane markings only are picked up.

3.5 Signed Edges and Hough Transform for Outlier Removal

The edge pixels contained in SLEM correspond to straight line features in the image. However, all these lines are not necessarily those of lane markings. Straight lines from vehicle edges, shadows, road markings, pavements and other artifacts are also picked up in the SLEM. In order to promote only lane features and eliminate non-lane features, we use the third and fourth properties (*P3* and *P4*) of lane markings, i.e., a lane marking has a left edge formed by dark to light transition, and a right edge formed by light to dark transition. Also, these edges of opposite nature are separated by a predictable distance apart. This distance measure is referred to as the ‘thickness criteria’, as it is governed by the predictable thickness of the lane marking in the near view.

To select line-pairs where the left and right lines capture edges of opposite intensity transitions, we first spilt the edge pixels in straight line edge map E_{SL} into two groups using the equations below:

$$\begin{aligned} E_{SLl} &= \{E_{SLl}(x, y) : \nabla_x I(x, y) \geq 0\} \\ E_{SLr} &= \{E_{SLr}(x, y) : \nabla_x I(x, y) < 0\} \end{aligned} \quad (3.3)$$

where E_{SLl} has the edge pixels that show dark to light intensity transition, and the edge pixels in E_{SLr} show light to dark intensity transition. These edge maps are called as *directional edge maps*. According to properties *P3* and *P4* of a lane marking, we should get two straight line edges L_l and L_r , which transit from dark→light and light→dark respectively, and are separated by a small distance $\Delta\rho_{lr}$ (‘thickness criteria’ in *P4*). This distance corresponds to the expected thickness of the lane marking in the near view of the camera, and can be considered as a priori data. The number of pixels this corresponds to can be computed as a one-time calibration process.

Thus, given the two edge maps E_{SLl} and E_{SLr} , the next step is to check whether E_{SLl} has a line that runs in parallel to another line in E_{SLr} , where these two lines are separated by $\Delta\rho_{lr}$. In order to do this, we employ the Hough Transform (HT) [108]. The HT is conventionally applied on a given edge map to locate lines of interest and its computation is based on the parametric representation of lines passing through a point

3.5 Signed Edges and Hough Transform for Outlier Removal

in the image plane indexed by $x - y$ coordinates. A point $P(x, y)$ in the image plane is defined by a set of intersecting lines at $P(x, y)$. Each line passing through $P(x, y)$ can be represented by its parametric representation using

$$\rho = x \cos\theta + y \sin\theta \quad (3.4)$$

where (x, y) is the pixel coordinate in the image plane, ρ is the length of the perpendicular from the origin to the line and θ is the angle made by the perpendicular with the x -axis. Every line that passes through $P(x, y)$ is represented by a unique $\rho - \theta$ combination. These $\rho - \theta$ values are used to create a two dimensional parametric space called Hough space where a point in image space is represented by a sinusoidal curve. The Hough Transform can be used to detect collinear points by studying the Hough space as shown in Fig. 3.7. Points, $P_1(x, y)$, $P_2(x, y)$ and $P_3(x, y)$, lying on the same line (Fig. 3.7(a)) share the same $\rho - \theta$ combination, (ρ_l, θ_l) , which implies that the sinusoidal curves of the collinear points intersect in the Hough space (Fig. 3.7(b)). It is implemented using a two-dimensional $\rho - \theta$ accumulator array as shown in Fig. 3.7(c). For every edge pixel coordinate, the ρ values are computed for angles from θ_{min} to θ_{max} and the bins corresponding these $\rho - \theta$ combinations are incremented. After processing the entire edge map, the array is scanned for accumulators that give high values or peaks. A peak in the array represents intersection in the parametric space and hence a straight line in the edge map.

Figure 3.7: Straight line Hough Transform (a) Collinear points in image plane (b) Intersecting sinusoids in Hough space (c) Implementation as a two-dimensional array of accumulators.

Now, in order to check for the presence of the left and right lane marking edges in E_{SLl} and E_{SLr} respectively, the Hough transform for E_{SLl} is first computed to give

3. Basic Linear Road Markings

a Hough space HS_l . The presence of peaks are then checked in the predefined angle range of HS_l . The predefined angle range, that we search in HS_l is given by the angle range that was previously obtained from the peaks in GAH. Let us say, we get a peak at (ρ_l, θ_l) in HS_l . We then compute the HT of E_{SLr} for a small angle range around θ_l resulting in Hough space HS_r . This is done because, the lanes edges in E_{SLl} and E_{SLr} are expected to be parallel edges, i.e., they must be subtending similar angles in Hough space. If we get a peak at (ρ_r, θ_r) in HS_r , we find:

$$\Delta\rho = |\rho_l - \rho_r| \quad \Delta\theta = |\theta_l - \theta_r| \quad (3.5)$$

The straight line edges in E_{SLl} and E_{SLr} are confirmed as lane markings if,

$$(\Delta\theta < \delta) \text{ AND } (\Delta\rho_{lr}^l < \Delta\rho < \Delta\rho_{lr}^u) \quad (3.6)$$

where δ is a small angle range. $\Delta\theta < \delta$ implies that the edges belong to parallel straight line edges. In addition, if $\Delta\rho_{lr}^l < \Delta\rho < \Delta\rho_{lr}^u$ is also satisfied, then the linear parallel edges, one in E_{SLl} and the other in E_{SLr} , are separated by the distance governed by the thickness of the markings. This satisfies properties $P3$ and $P4$ of lane markings. For the purpose of validation, the thickness measure $\Delta\rho_{lr}^l$ and $\Delta\rho_{lr}^u$ are set to 5 and 10 pixels respectively.

Figure 3.8: Detecting line pair satisfying lane marking conditions using simplified Hough Transform and signed edges.

At the end of this step, if the above conditions are met for line pair $l - r$, then their positions and angles (ρ_l, θ_l) and (ρ_r, θ_r) are output, where l and r refer to the left and right counterparts of the line pair. If no such line pair is found satisfying all the

3.5 Signed Edges and Hough Transform for Outlier Removal

Figure 3.9: Illustration of outlier removal process (a) Image block of size 90×128 with lane markings and other edges like tar color change along the direction of the lane (Type C scenario), (b) Straight line edge map E_{SL} obtained using GAH in feature extraction step, (c) Left directional edge map E_{SLl} with the left lane marking edge and the tar color change edge also, (d) Right directional edge map E_{SLr} with the right lane marking edge, (e) Resulting straight lines that are identified as having the lane markings after analysing the Hough spaces.

above conditions, then it is concluded that the edge pixels received from the block do not represent a lane marking. Fig. 3.8 summarizes these computation steps.

The above mentioned process is illustrated using the example shown in Fig. 3.9. The selected block shown in Fig. 3.9(a) gives the SLEM E_{SL} shown in Fig. 3.9(b) after the feature extraction step. It can be seen that the SLEM processing has eliminated all edges except those straight line edges that are in the direction of the left lane. In Fig. 3.9(b), there are both valid lane marking boundaries and an edge that does not belong to a valid lane boundary, i.e., the edge formed by tar color change. When E_{SL} is split into the left and right directional edge maps E_{SLl} and E_{SLr} , we get the edges shown in Fig. 3.9(c) and (d) respectively. It can be seen that E_{SLl} has two edges corresponding to dark→light intensity transitions, one from the lane marking and other from the dark tar color to light tar color. However, E_{SLr} has one edge corresponding to the light→dark transition of the lane marking.

The lane marking needs to now be identified from these edges using HT. The parameters $\Delta\rho$ and $\Delta\theta$ were initialized to be 10 and 5° respectively based on heuristics. On applying HT to E_{SLl} , we get the following peaks in its Hough space HS_l represented

3. Basic Linear Road Markings

by their (ρ, θ) values: $(502, 40^\circ)$ and $(515, 49^\circ)$. HT is now applied to the right edge map E_{SLr} for the angle ranges surrounding 40° and 49° . The peak detected in the Hough space HS_r is given by $(506, 43^\circ)$. It can be seen that the (ρ, θ) pairs - $(502, 40^\circ)$ in HS_l and $(506, 43^\circ)$ in HS_r satisfies the Δ_ρ and Δ_θ conditions in equation (3.5). Therefore, the edges of the lane markings are picked up while the other edge in the direction of the lane is rejected as shown in Fig. 3.9(e).

3.6 Dataset for Performance Evaluation

Figure 3.10: Map showing routes in Singapore along which test videos were captured.

In order to evaluate exhaustively the proposed algorithms in this thesis, video sequences were captured on major roads and highways in Singapore, representing varying illumination conditions of day time, late evening, night, midnight and tunnel lighting. The different various weather conditions considered were clear skies, cloudy, drizzle and rain. Scenarios with bright and faint shadows of trees, shadows of overhead bridges and vehicles and presence of other road markings were also considered. Fig. 3.10 shows the map of Singapore indicating the routes on which the test videos were captured. Fig. 3.11 (a) shows a snapshot of the test vehicle that was used for capturing the dataset, and Fig. 3.11 (b) shows the test camera that was used. The specifications of the camera are as follows: It is a USB car camera, having a wide angle lens of 120° field of view and a focal length of 6 mm. It has various video resolution options, out of which HD resolution with height $\times$ width being 720×1280 was chosen. The bottom 180 rows of pixels in the image capture the bumper of the car at all times (Fig. 3.11 (c)), which

were cropped out, and only the cropped images are considered for further processing. A sample of a cropped image in Fig. 3.11 (d). The test images are extracted from video sequences at the real-time frame rate of 30 frames per second, resulting in a total of more than 6,000 test images.

Figure 3.11: (a) Test vehicle used for capturing dataset (b) Camera set up inside test vehicle (c) Sample image as captured by the camera (d) Image after cropping.

A dataset containing 20 real image sequences was created from the video sequences for evaluating the proposed BLM based lane detection method. The list of test images sequences, numbered from 1 to 20, along with the keywords that describe them, is presented in Table 3.1. Each video stretch is tagged with keywords describing the road scenario in those frames, such as ‘shadows of trees’, ‘night scenario’, ‘vehicle in tunnel’, ‘drizzle’, ‘artifacts due to changing tar color’ etc.

3.7 Performance Evaluation

The validation of all algorithms proposed in this thesis has been carried out using MATLAB version 7.12 (R2011a). However, it must be noted that MATLAB was used only for the functional validation of the methods, and the MATLAB implementations are not indicative of metrics such as time complexity. This is because the techniques

3. Basic Linear Road Markings

Table 3.1: Description of test videos

No.	N_f	Description keywords
1	143	tree shadows, day time, partially cloudy
2	288	changing tar color, dull weather, low illumination, day time
3	265	tunnels, artificial lighting
4	301	daytime, bright contrast of road markings, dark road
5	301	faded markings, light road color, day time, no shadows
6	299	evening, tar color changes
7	281	midnight road scenario
8	361	overcast skies, low lighting, light drizzle
9	361	changing tar color, late evening
10	228	night time with streetlights and headlights on
11	278	presence of other markings like arrows
12	301	drizzle, wipers on
13	260	drizzle, wipers on
14	601	rain, wipers on, evening lighting, cloudy, curved road
15	291	daytime, bright contrast of road markings, dark road
16	299	after sunset, very low lighting, presence of arrows
17	357	evening, tar color changes
18	301	wet roads, evening light
19	479	overcast skies, low lighting, light drizzle
20	451	sunny, thick shadows from trees and overhead bridge, glare from windshield, curved road

proposed in this thesis exploit hardware acceleration techniques like parallelism, which are not captured by MATLAB. Comparative analysis is therefore done by comparing the underlying computational complexities.

In this section, the evaluation results of the proposed method are presented using the SLEM based feature extraction, followed by outlier removal using signed edges and HT , to detect near lane markings.

Ground truth for each test frame was manually generated by capturing the coordinates of the lane markings in the near view, using the function `getpts()` in MATLAB. Each of the test images was subject to the proposed lane detection algorithm. The detected lane marking output, given by the (ρ, θ) values generated at the end of the outlier removal step (illustrated in Fig 3.9), was plotted back on the original image. These results were evaluated against the ground truth data by visual inspection, and the various error metrics are derived as below:

- True Positives (TP): TP is defined as an image sample wherein a lane marking is visible in the near view, and the algorithm also detects the lane marking in the same location i.e, the difference in the (ρ, θ) values of the ground truth and the

experimental result is acceptable.

- False Positives (FP): FP refers to those image samples where there is no lane marking in the near view, but the algorithm detects the presence of it. FP is also used to refer to cases where the marking is detected, but inaccurately.
- True Negatives (TN): TN refers to the samples where there is no lane marking present in the near view, and the system also does not detect any lane markings
- False Negatives (FN): FN is defined as an image sample where there is a valid lane marking present in the near view, but the system misses it.
- Detection Accuracy (Acc): The total detection accuracy is defined by the percentage of total number of true detections on a scale of 0 to 1, and given by the equation below:

$$acc = (TP + TN)/N_f \quad (3.7)$$

where N_f is the total number of frames in the video sequence considered

The algorithm was applied on each of the 20 image sequences captured in various scenarios, as summarized in Table 3.1. In order to set the block size for SLEM generation, we consider the minimum length of line features of lane markings in RoI. This is taken to be 14 pixels based on heuristics, and anything smaller than this is regarded as noise. In other words the T_{GA} for the SLEM generation step is taken to be 14. In Sec. 3.3.1.1, it was shown that a good block size with respect to a given T_{GA} is given by a value around 4 to 6 times T_{GA} . For the image sequences considered with image size is 720×1280 , a block size of 90×128 satisfies the criteria for T_{GA} and also divides the image into blocks without any padding. Also, this block size resulted in SLEMs with lane edges in most cases, as seen on visual inspection of these SLEMs. The $\Delta\rho_{lr}^l$ and $\Delta\rho_{lr}^u$ values for the thickness criteria are set to 5 and 15 pixels respectively for lane markings. For the special case when thicker lane markings are to be detected, these values are reset to 15 and 30 respectively. Angle range for left and right lane markings for GAH and HT computations were set to be $(20^\circ \text{ to } 70^\circ)$ and $(-20^\circ \text{ to } -70^\circ)$ respectively. The bottom 180 rows of bumper area corresponds to two rows of blocks,

3. Basic Linear Road Markings

which are cropped out. The bottom-most row after cropping is taken to be the near view RoI.

The results of the proposed algorithm are summarized in Table 3.2, where the TP , FN , TN , FP and acc are generated for the left and the right lane markings separately. Subscripts $_l$ and $_r$ are used for denoting the left and right lane markings of the host lane. The sequence index and the number of frames in each sequence (N) are also indicated.

Table 3.2: Results of detecting basic linear markings along the road

No.	N_f	TP_l	FN_l	TN_l	FP_l	acc_l	TP_r	FN_r	TN_r	FP_r	acc_r
1	143	120	2	20	1	0.979	124	3	15	1	0.972
2	288	231	0	47	10	0.965	166	7	115	0	0.976
3	265	239	2	24	0	0.992	265	0	0	0	1.000
4	301	274	3	24	0	0.990	295	1	0	5	0.980
5	301	268	4	29	0	0.987	298	1	0	2	0.990
6	299	277	2	20	0	0.993	290	3	0	6	0.970
7	281	276	3	0	2	0.982	269	2	9	1	0.989
8	361	359	1	0	1	0.994	349	4	2	6	0.972
9	361	283	1	77	0	0.997	270	2	89	0	0.994
10	228	217	2	8	1	0.987	224	2	0	2	0.982
11	278	237	1	35	5	0.978	196	0	82	0	1.000
12	301	254	1	46	0	0.997	262	2	37	0	0.993
13	260	234	2	24	0	0.992	226	1	32	1	0.992
14	601	471	5	122	3	0.987	467	3	127	4	0.988
15	291	279	2	6	4	0.979	278	4	9	0	0.986
16	299	280	3	16	0	0.990	289	3	0	7	0.967
17	357	309	6	42	0	0.983	349	4	0	4	0.976
18	301	263	3	35	0	0.990	272	5	18	6	0.963
19	479	336	2	138	3	0.989	468	4	0	7	0.977
20	451	316	7	128	0	0.984	441	10	0	0	0.978
Avg						0.987					0.982

It can be seen that the algorithm yields a high average detection accuracy of 98.7% and 98.2 %, in the left and right lane markings respectively. Also, in almost all sequences the number of TP is significantly high with respect to the total number of positives in the sample set. This metric, conventionally referred to as ‘True Positive Rate’ or TPR is higher than 0.97 in all sequences. The system performs well in complex scenarios, such as shadows, and presence of other similar artifacts. Fig. 3.12 illustrates a sample set of images with near lanes identified by the proposed method in the different complex scenarios. The proposed method is able to identify the near lane markings in the presence of randomly scattered shadows of trees, with and without vehicles (Fig.

3.12(a)&(b)), uniform shadows of overhead bridges (Fig. 3.12(c)), shadows of vehicles that are closely following the lane markings (Fig. 3.12(d)) and tar color changes (Fig. 3.12(e)). Each of these scenarios is discussed below.

Figure 3.12: Near lane detection in complex scenarios: (a) & (b) shadows of trees and other random features, (c) shadows of overhead bridges falling on lane markings (d) presence of vehicle shadows in the direction of lane marking (e) edges due to tar color change edges in the direction of lane markings.

3. Basic Linear Road Markings

Shadows from trees are random in their distribution and the use of SLEM selects only those edges pixels that conform to straight lines. This eliminates most of the edge pixels forming tree shadows. Sometimes shadows from trees do give horizontal edges. Again, these are filtered out by setting the appropriate angle range in GAH and by checking for peaks only in those angle ranges. In the case of shadows from overhead bridges and other fixtures such as lamp posts, the edges are usually not along the lane. Setting the priori angle range in GAH can filter out line edges that are present at any other angular orientation, other than that of the lane markings.

Figure 3.13: Lane marking detection using proposed method in the presence of edges that are along the direction of lanes towards the vanishing point: (a) & (b) show tar road color changes that create edges along the lanes, (c) & (d) show the presence of vehicle shadows in the direction of lanes.

Edges from vehicles are usually linear and also aligned towards the vanishing point, just like lane markings. If these vehicle edges fall in the expected position of lane markings, then they tend to create features resembling lane markings. Similarly, when the tar color of the road changes uniformly along the lane, it leads to straight line edges in the direction/position of lane markings. It is more difficult to distinguish these features from lane markings as they show similar properties in terms of their orientation and position, as compared to randomly distributed edge pixels like tree shadows. However, the use of signed edges with Hough Transform and the thickness criteria successfully eliminates these edges from being considered as lane markings. Some of these cases are shown in Fig. 3.13.

Figure 3.14: Lane marking detection in the presence of other road markings.

The effectiveness of the proposed method is also shown in extracting lane markings in the presence of other road markings. Fig. 3.14 shows two of these cases. The different kinds of arrows in Fig. 3.14 are rejected and the lane markings only are detected in the near view. The arrow markings are not picked up by the proposed method based, as they are not found in the expected RoI, do not have parallel edges and do not satisfy the ‘thickness criteria’. In Fig. 3.14(b), it can be seen that the proposed method rejects the chevron markings beside the right lane and selects the right lane marking only.

The system gives a very high accuracy of $acc_l = 0.992$ and $acc_r = 1.0$ for sequence #3, where the vehicle enters a tunnel. This is because the bright artificial lighting enables a very high contrast between the lane markings and the road, which is also

3. Basic Linear Road Markings

relatively dark in this case. This high contrast, coupled with the fact that there is no noise from other artifacts, unlike the case in other sequences captured outdoors, this particular sequence yields the best detection accuracy.

Next, we need to also evaluate the system performance in cases where no lane markings are present. Sequences which have solid lane markings, almost always show a lane marking in the near view, and hence have few number of negatives in the sample data to start with. However, sequences #2, #9 and #11 contain considerable stretches where the vehicle moves through zones of no lane markings, such as intersections. It is seen that the number of TN detected almost equals the total number of negative samples (given by $TN + FP$). This therefore yields a high TNR or ‘True Negative Rate’ (given by the ratio of the number of true negative detections with respect to total number of negatives in the test sequence). The negatives in sequences such as #1, #3, #4, #5 etc. happen because of the periodic absence of lane markings in the near view, at regular intervals due to the dashed lane markings. Again, the system successfully detects no lane markings in these cases. It will be shown later that this phenomena of periodic presence and absence of lane markings in the near view in the case of dashed lane markings, as against a continuous presence of the markings in the case of solid lane markings, is exploited later in Sec. 3.10 to distinguish between the two types of lane markings.

There was a small percentage of mis-detections that were observed. Some of the examples of mis-detections are shown in Fig. 3.15. In Fig. 3.15 (a), an image from sequence #8, has a SLOW marking painted in front, and the vertical stroke of the painted letter ‘L’ resembles a lane marking, in terms of satisfying most properties of lane markings. This gets picked up as the lane marking instead of the actual lane marking that is present left of this, leading to a false positive. In another scenario shown in Fig. 3.15 (b), the road contains badly worn out lane markings, thus resulting in a poor edge map. The overcast sky and low overall lighting further contributes to the markings appearing faint, thus making it difficult for the edges to be detected, thereby resulting in a false negative. In 3.15 (c), a pavement border occurs in the expected lane marking position on the right side of the host lane. Also, there is no other lane marking on this side. The painted pavement border resembles a dashed lane marking and since the system cannot locate any other marking on this side of the lane, mistakes this to

3.8 Enhancing Robustness using Temporal History

be the lane marking. It will be shown in Sec. 3.8 that most of these misdetections can be rectified by a temporal analysis of the frames.

Figure 3.15: Examples of misdetections (a) Presence of other road markings that resemble lane marking (b) Very faint and worn out markings (c) Pavement border resembling dashed lane marking.

3.8 Enhancing Robustness using Temporal History

One of the observations with the road scene videos is that it exhibits a continuity across time frames (property $P5$ of lane markings, Sec. 3.2). It is therefore expected that the positions of lane markings do not undergo a drastic change across two consecutive frames. Analysis of detection outputs of past frames can help in two ways:

1. Validating position of lane marking detected and removal of outliers: The lane position detected in the current frame is expected to be consistent with the position in previous frames. An extra check on the current detection output to validate that it is consistent with history information helps to further improve the robustness of the proposed method. Lane position outliers can be isolated and rectified using the temporal continuity analysis. For example, as shown in Fig. 3.15, it is possible that other features that resemble a lane marking, that

3. Basic Linear Road Markings

are also present in the near view are sometimes confused for the lane marking. However, this happens usually only at certain frames when the confusing artifacts enter the near view and exhibit similar position and orientation as that of valid lane markings. Such cases are considered to be outliers and can be easily spotted using temporal history information of the lane position, by performing a continuity check. Infact, we will show here that most of the misdetections discussed above in Sec. 3.7 can be rectified by isolating outliers across time frames.

2. Bridging gaps in detecting lane RoI: The region bound by the host lane is used as region of interest (RoI) for detecting other markings, such as horizontal BLM, as will be shown in subsequent chapters. Hence, it is important to have the lane position at all times, even if markings are briefly absent in the near view for a few frames. It was explained above in Sec. 3.7 that sequences like #1, #3, #4, #5 have a high number of negatives in the near view, simply because of the dashed lane markings appearing and disappearing periodically in the blocks of interest that define the near view. In the absence of visible markings in the near view, history information serves as an indicator to infer the RoI. To illustrate this point, consider the example shown in Fig. 3.16, of a dashed lane marking. Frames (a), (b), (c), (d) are taken from the same video sequence with increasing time stamps, captured within 1s time window. The near view (after cropping the bumper, for better visualization) is marked by a black dashed line in the bottom region of the frames. In (a) and (b), the lane marking on the left side of the host lane is almost fully absent in the near view. It starts to enter the near view in frame (c) and has a significant presence in the nearview in (d). Since we do not expect drastic changes in the lane positions between consecutive frames in a real video sequence, temporal analysis can be used to infer missing information in frames such as (a) and (b). In other words, history information can be used for inferring information, in cases where the current frame does not carry sufficient data.

Flowchart in Fig. 3.18 shows the steps for using temporal history information for achieving both the above mentioned objectives. Lane detection algorithm outputs the presence/absence of lane marking in the near view and also the position of the lane marking in terms of θ and ρ given by the Hough Transform. If k is the index of the current frame, then L_k is a binary variable used to denote whether the lane detection

3.8 Enhancing Robustness using Temporal History

Figure 3.16: Example of lanes present/absent in near view; (a) and (b) taken within 1s time window.

Figure 3.17: Incorporating temporal history into the lane detection process for improved robustness.

algorithm is able to detect a lane marking or not in the near view, and is assigned the value 1 or 0 respectively. At any given stage, the lane detection outputs (L_k, ρ_k, θ_k) of the past N frames are stored in a first-in first-out (FIFO) buffer. At the time of initialization, temporal algorithm waits for the first N frames to be processed and the FIFO buffer to be full, having a minimum number of 1's in the L array.

3. Basic Linear Road Markings

If the lane marking is found in the near view, then its position in terms of (ρ, θ) values is checked against the history, by considering the past N frames. Error functions ρ_Δ and θ_Δ are computed by subtracting the current ρ, θ values from the average of the ρ, θ values of the past N frames. If these differences exceed the maximum errors η_ρ and η_θ , then the detection output is considered as an outlier. This is based on the premise that there is continuity in the lane marking positions across frames, and any changes will be gradual. If an outlier is spotted, then the lane position for that frame is replaced by the ρ, θ values generated from history. The average ρ, θ values is computed for the previous N frames from history. If L_k is 0, where no lane marking is detected in the near view, then the host lane RoI is inferred as the ρ, θ values of the previous frame ($N - 1^{th}$ frame). If the number of zeros in L within the past N frames, which is the number of times the lane marking was not detected in the recent history, exceeds a value Z_{max} , then it is inferred that the vehicle has entered a zone of no lane markings and the FIFO buffer is reset. N was heuristically set to a value of 7.

Fig. 3.18 shows an example of the output ρ and θ values after the proposed temporal tracking is applied, on the left and right lane markings separately. The sequence considered has dashed markings on the left side and continuous markings on the right. The red graph indicates the (ρ, θ) values prior to tracking, and the green graph shows the (ρ, θ) values after tracking. It can be seen that the left side shows periodic dips prior to tracking, due to the dashed marking resulting in negatives in the near view. These gaps are now filled by the tracking, and the lane position is output at all times. Not only that, in the right side, where the marking is solid, there are a few false negatives or outliers. These are now fixed due to the temporal tracking.

3.9 Overall BLM Detection Algorithm

This section first summarizes the steps described in the above sections for the detection of lane markings, and then generalizes the approach for detecting any BLM.

The inputs to the algorithm are:

1. Region of interest, RoI
2. Expected angle range, $\Delta\theta$
3. Expected thickness Δd

Figure 3.18: Lane position after temporal tracking.

As explained in Sec. 3.4, the RoI was taken to be the near view comprising of the last row of blocks in the test images. The left half of the near view was considered for detecting left lane markings and right half for the right lane marking. Angle range for left and right lane markings were set to be $(20^\circ \text{ to } 70^\circ)$ and $(-20^\circ \text{ to } -70^\circ)$ respectively. The expected thickness for lane markings was taken to be in the range of 5 to 15 pixels.

Firstly, the blocks in the RoI are subject to SLEM generation using gradient angle histograms, as per algorithm 1. The resultant edge maps contain mainly straight line edges and thus capture the lane marking features. To eliminate lines that do not belong to lane markings, only those line pairs are chosen which exhibit edges of opposite nature, and separated by a known distance apart. To check for these efficiently, first, the positive edges are sent into HT. Only for the positions where peaks are detected in the positive edge map, the vicinity is searched for presence of peaks in the negative edge map. The output of this is the peak positions of the line pairs detected. The (ρ, θ) of these peaks give the position and angular orientation of the lane markings.

The above set of steps can be generalized to detect any BLM, by adjusting the region of interest, expected angle range and expected thickness to appropriate values. Flowchart in Fig. 3.19 summarizes these steps for detecting any BLM. Based on this, we formulate a generic function for detecting BLM using the proposed method, as described below in eq. (3.8):

$$S = \text{blmDetect}(\Delta\theta, I_{ROI}, \Delta d) \quad (3.8)$$

3. Basic Linear Road Markings

Figure 3.19: Flowchart summarizing the steps in BLM Detection. The arrows between the signed edge map generation and HT blocks are indicated in both directions to convey the feedback from first processing the positive edges, after which HT is performed only for the selected angle range on the negative edges.

where S corresponds to set of m valid line pairs detected, given by

$\{ \{ (\rho_{l1}, \theta_{l1}), (\rho_{r1}, \theta_{r1}) \}, \{ (\rho_{l2}, \theta_{l2}), (\rho_{r2}, \theta_{r2}) \} \dots \{ (\rho_{lm}, \theta_{lm}) \} \}$. Input parameters such as RoI, expected thickness of BLM and expected angle range can be set based on the marking to be detected. The linear feature extraction step using SLEM, and the outlier removal step using HT and signed edges are applied for these parameters in similar manner as was shown for lane markings. It is shown in subsequent sections and chapters how `blmDetect()` is used as a basic function for detecting various other kinds of markings.

3.10 Distinguishing Solid & Dashed Single Lane Markings

The algorithm proposed for detecting BLM can be extended to distinguish them as solid or dashed. A dashed lane marking is visibly characterized by gaps occurring at

3.10 Distinguishing Solid & Dashed Single Lane Markings

regular intervals, as we move along the marking, in the image. A solid lane marking on the other hand would indicate a continuous presence when we traverse along the marking. This spatial discontinuity of the dashed marking in a still image, translates into a temporal discontinuity in a video sequence as well. That is, if one were to be viewing the lane marking through a fixed window, across time frames, then the presence and absence of the lane marking would be seen in an alternating fashion. On the other hand, a solid marking shows continuity in the marking, with no gaps, for that stretch.

Figure 3.20: Distinguishing lane markings as dashed and solid through two pronged approach (i) Spatial continuity check and (ii) Temporal continuity check.

The proposed algorithm uses a two-pronged approach for distinguishing between a dashed and solid linear marking. One is a spatial continuity check that is applied within an image frame, and the other is a temporal continuity check that is applied across time frames. Having both the spatial and temporal checks together strengthens the accuracy of the proposed approach.

3.10.1 Approach 1: Spatial Continuity Analysis

From the previously explained algorithm to detect basic linear markings along the road (Sec. 3.9), the position of the lane marking and its angular orientation are determined. Now, in order to classify this marking as dashed or solid, the block in which the lane marking is detected is selected, along with its neighboring blocks on either side horizontally, in the bottom most row of blocks. We explain the algorithm for the left lane and the same can be extended for the right lane also. Let the left and right edges of the left host lane determined by `blmDetect` be given as $L_L^l(\rho_L^l, \theta_L^l)$ and $L_L^r(\rho_L^r, \theta_L^r)$ respectively. Let the block in which the lane was found be $B(i, j)$, where i is the last row (i.e. $i = 8$). Therefore, in order to detect the spatial continuity check, we consider blocks $B(i, j - 1)$ to $B(i, j + 1)$. As an example, in the image shown in Fig. 3.22 (a), the left lane marking is detected at $j = 3$; therefore, blocks 2,3 and 4 are considered

3. Basic Linear Road Markings

for processing the left lane marking. Similarly, blocks 5, 6 and 7 are considered for the right host lane. The reason why these adjacent blocks are considered is because the lane marking is expected to extend into these blocks. Therefore, checking for the presence of the lane features in these blocks will determine the type of the lane marking.

Figure 3.21: Generating mask for spatial continuity check (a) Sample image block containing left lane marking (b) Edge map (c) Result of AND-ing the edge map with mask.

Figure 3.22: Using spatial continuity analysis for distinguishing solid vs dashed lane markings (a) Image divided into blocks.

Next, the positions of the left and right edges of the lane marking is used to generate a mask M_L , such that it contains the lane marking. This mask is generated so that we can probe the edges along the lane marking only and eliminate all other edges. We consider a small allowance to the positions of the left and right edges to capture the lane markings within the mask. This is illustrated in Fig. 3.21. Fig. 3.21 (a) shows an image block with the lane marking, (b) shows the edge map of the marking and (c) shows the mask that needs to be considered that can encapsulate the lane marking.

3.10 Distinguishing Solid & Dashed Single Lane Markings

Figure 3.23: Using spatial continuity analysis for distinguishing solid vs dashed lane markings.

Given $L_L^l(\rho_L^l, \theta_L^l)$ and $L_L^r(\rho_L^r, \theta_L^r)$, the mask in block $B(i, j)$ is generated by checking if the coordinates (x, y) in $B(i, j)$ satisfy the following:

$$(\rho_L^l - \delta_\rho < (x \cos \theta_L^l + y \sin \theta_L^l)) \wedge ((x \cos \theta_L^r + y \sin \theta_L^r) < \rho_L^r - \delta_\rho) \quad (3.9)$$

In the above equation, δ_ρ is the small allowance that is set, to capture the lane marking within the mask. All coordinates (x, y) s in $B(i, j)$ that satisfy the above equation will result in M_L . In other words, all coordinates that are located to the right of the left edge and left of the right edge, satisfy the above equation resulting in M_L . This mask is then used to selectively get the lane edges along the left lane. Given the straight line edge map (SLEM) for block $B(i, j)$, represented by $E_{SL(i, j)}$, the selective edge map along the lane $E_{ML(i, j)}$ is obtained as:

$$E_{ML(i, j)} = E_{SL(i, j)} \odot M_L \quad (3.10)$$

where $\odot$ represents elementwise multiplication. The same procedure is repeated for all the three blocks, i.e. $B(i, j - 1)$ to $B(i, j + 1)$ and the E_{MLs} of the three blocks are concatenated in the same order to get the overall E_{ML} .

We generate a vertical projection P_L of E_{ML} by summing all edge pixels along the horizontal direction. We traverse row by row in the edge map contained within the mask region and the sum of edge pixels is computed for each row and these values are

3. Basic Linear Road Markings

projected into a linear array, which is referred as projection array P_L . The projection arrays are shown in Fig. 3.22 (b) and (c) for the left and right lane markings respectively. The projection array has 90 values corresponding to the block height of 90, where each row yields an entry in the array.

A continuous string of 0's in this array indicates a gap in the marking, and hence a dashed marking. In the illustration in Fig. 3.22, which is a case of dashed markings in both left and right sides of the host lane, the projection arrays are shown in Fig. 3.22 (b) and (c). Continuous strings of zeros, of lengths more than 30 and 40 are seen for the left and right lane markings respectively, clearly indicating the presence of dashed markings on either side. An example of a host lane with solid markings on one side and dashed markings on the other is shown in Fig. 3.23. The left side of the host lane contains a dashed marking, which is reflected by a stretch of more than 20 zeros in its corresponding projection array. On the other hand, the right side shows a solid marking, which is reflected by the absence of any zero-stretches in the projection array. The stray zeros that are present due to breaks in the edges and are discarded as outliers. Similarly, if there are stray edge pixels in the gap regions, they are ignored as outliers. The steps for spatial continuity check are summarized as a flowchart in Fig. 3.25 after the following section on temporal continuity check.

3.10.2 Approach 2: Temporal Continuity Analysis

The above spatial continuity analysis can be used to distinguish the lane markings as dashed or solid in a static frame, and this method can be applied even if the vehicle is stationary. However, for moving vehicles, temporal history information can serve as an additional cue to confirm the finding, through a continuity analysis of the markings across time frames is performed. This is based on the observation that a solid marking not only exhibits spatial continuity, but also a temporal continuity. On the other hand, a dashed marking exhibits gaps in the spatial domain (as shown in Sec. 3.10.1), as well as across time frames, as explained below.

In order to perform temporal continuity check, a small slice of the image from the bottom most row of blocks (near view), 10 pixels in height is chosen as the window for viewing the lane markings. The presence or absence of lane markings in this image slice is checked in every image frame, by applying a mask around the lane position that is output as (ρ, θ) by the proposed lane marking detection method. The same mask that

3.10 Distinguishing Solid & Dashed Single Lane Markings

was generated for the spatial continuity check in the previous approach is used. The lowermost 10 pixel window from bottom is used to check if there are any edge pixels that are captured in the mask. The total number of edge pixels in this window N_L and N_R are determined for the left and right host lanes respectively. If this count is greater than a given threshold, a temporal variable $R(k)$ is updated, where k represents the time instant. In other words, $R(k)$ is used to keep track of whether or not the lane marking was detected in the image slice in the k -th frame. The R values of the previous M frames in the history are checked. When a minimum number of zeros (marked by Z_{min}) is found in this array, it indicates a gap in the marking, and hence presence of a dashed marking. Also, when the number of zeros exceeds a certain threshold value Z_{max} , it means the vehicle has entered a zone with no lane markings (as also shown previously in Sec. 3.8). The temporal continuity check is summarized in the equation below:

$$Z_{max} > M - \left[\sum_{i=0}^M R(k-i) \right] > Z_{min} \quad (3.11)$$

where $M = f(FR, V)$, FR being the frame rate per second and V being the velocity of the vehicle in meters per second. The number of frames M that are stored and checked for temporal continuity depends on the frame rate and the velocity of the vehicle. For a stationary vehicle or a very slow moving vehicle, only spatial continuity check is employed. The temporal continuity check is triggered only if the vehicle is moving with a minimum speed of 10m/s. The speed of the vehicle in the test videos considered for this experimentation was in the range of 10m/s to 25m/s. Given that the frame rate in the camera set up used was 30 frames per second, and given that the gaps in the dashed markings are around 2m to 3m long on non-expressways in Singapore, a maximum of around 10 frames is required to fully capture the gap. M is set to 20 and Z_{max} is set to 10. Setting M to 20 frames ensures sufficient buffer to capture the gap along with the marking.

It is to be noted that Z_{max} is a parameter that can be adjusted based on the knowledge of the inter-dash spacing (which is in turn affected by whether the vehicle is on an expressway or non-expressway), and the speed range of the vehicle. The test videos that were considered for validation of the temporal continuity analysis were mainly taken from non-expressways, where we find more road markings than expressways. Hence, Z_{max} was set to 10 (given inter-dash spacing of about 3m and vehicle speed in the

3. Basic Linear Road Markings

range of 10 to 25 m/s). However, if the system were to be tried on expressways, then Z_{max} would need to be set to a higher value to allow detection of relatively sparsely spaced dashed markings, without falsely detecting road with no markings. One way to automatically adjust the Z_{max} value is based on the length of continuous strings of ‘0’s and ‘1’s in the R array, which would indicate how sparsely or closely the dashes are present in the lane marking.

Fig. 3.24 shows an illustration of the R array for lane markings on either side of the host lane.

Figure 3.24: Using temporal continuity check for differentiating solid vs dashed lane markings.

Performance evaluation: The proposed approach to distinguish lane markings using temporal continuity check is applied on the 20 test sequences in the dataset described in Sec. 3.6. Table 3.3 shows the results of the temporal continuity analysis for some of the sequences in the dataset. A sample image indicating the nature of the markings in that sequence is shown. Also, the temporal arrays for the left and right lane markings are shown in two separate columns of the table, along with the classification outcome marked as ‘Dashed’ or ‘Solid’. It can be seen that the arrays

3.10 Distinguishing Solid & Dashed Single Lane Markings

conclusively show an alternating waveform pattern in the case of dashed markings, but is a constant waveform set to 1 in the case of solid markings.

Further, dashed markings can have varying gaps between every two dashes to indicate the degree of allowance given to vehicles to change lanes. Closely spaced dashed markings indicate a more restricted lane change, and sparse dashes indicate a free movement of vehicles between lanes. The temporal array generated through this analysis can also throw light on whether the dashed markings are closely or sparsely spaced. The troughs in the waveform correspond to the gaps, and hence the length of troughs in the alternating waveform is studied more closely. It was observed that the length of the trough is dependent on two parameters: (i) the speed of the vehicle and (ii) the length of gaps between the dashes. For the same lane stretch, if the vehicle speed is increased, then it can be expected that the frequency of the waveform increases, thereby decreasing the trough length, and vice versa. Also, for a given speed, if the troughs are longer, then it can be inferred that the gaps are relatively larger. This can be seen in image sequence # 9 in Table 3.3. The lane marking on the left host lane are closely spaced, while the markings on the right are sparse. This is reflected in the corresponding waveforms. Since the vehicle speed is fixed for the analysis of the left and right lane markings, the relative gaps can be easily inferred. In order to infer the exact length of gaps, this analysis can be extended further to an automated model that incorporates the actual vehicle speed as an input parameter.

Table 3.3: Results of temporal continuity check to distinguish lane markings

#	Preview	$Temp_L$	$Temp_R$
1		Dashed	Dashed
2		Dashed	Solid

3. Basic Linear Road Markings

3		Dashed	Solid
4		Dashed	Solid
5		Dashed	Solid
6		Solid	Dashed
7		Dashed	Dashed
8		Dashed	Dashed

The two approaches of spatial and temporal continuity analysis for distinguishing lane markings between solid and dashed, are summarized in a flowchart in Fig. 3.25.

3.10 Distinguishing Solid & Dashed Single Lane Markings

Figure 3.25: Flowchart summarizing steps in spatial and temporal continuity analysis for distinguishing lane markings.

It was observed that the main computationally complex operation in the proposed techniques is the mask generation which involves repeated computation of the HT equation as shown in equation (3.9). However, it can be seen from the flowchart in Fig. 3.25 that both the temporal and spatial continuity check modules use the same mask. Therefore, the mask need not be generated separately for each of the modules. But, computing equation (3.9) repeatedly even once, can be computationally complex. We will show in the forthcoming sections, how this complexity is reduced to simple additions and comparisons only, using the proposed computation technique in Sec. 4.3.

3. Basic Linear Road Markings

3.11 Detecting Horizontal BLM

In order to detect horizontal BLM in the host lane, we consider the host lane region in the near view as the RoI. This RoI is derived by the left and right lane markings that were detected in the lane marking detection step. Using the positions of left and right lanes given by (ρ_L, θ_L) and (ρ_R, θ_R) respectively, all those pixels that fall within the host lane, in the near view, are selected as RoI. This process is similar to the selection of edges within the mask that was used earlier for the spatial and temporal continuity check. All edge pixels whose coordinates satisfy the following will be analyzed for horizontal BLM detection:

$$(\rho_L < (x \cos \theta_L + y \sin \theta_L)) \wedge ((x \cos \theta_R + y \sin \theta_R) < \rho_R) \quad (3.12)$$

As mentioned previously, the computation complexity of solving this inequality is addressed by the proposed computation technique in Sec. 4.3. This is applied for all the blocks in the bottom two rows, which is considered as the near-view, as anything above that is not distinguishable by the human eye.

The edge pixels in the RoI are then subjected to BLM detection again but with different input parameters to the `blmDetect` module. Since we are interested in horizontal BLMs now, the angle range is set to close to 0° . This is illustrated in Fig. 3.26. First, the left and right lane markings bounding the host lane are detected, by performing a BLM detection on RoI_A and RoI_B respectively (Fig. 3.26(b)). RoI_A and RoI_B are the RoIs defined for detecting lane markings, as discussed in the previous chapter. Then, the left and right lane marking positions are used to determine RoI_C (Fig. 3.26(c)) for detecting horizontal BLMs, by employing the equation (3.12) shown above. BLM detection is now applied on RoI_C with an angle range of $0^\circ \pm \Delta$, as we are looking for horizontal markings. In this case $\Delta\theta$ is set to 10. Hence, the SLEM generation, Hough computation and Hough peak detection are now restricted to the angle range of -10° to $+10^\circ$. The result of the `blmDetect` module on RoI_C with the above settings will result in the horizontal marking as shown in Fig. 3.27(d).

This entire process of detecting the horizontal BLMs is summarized below in Algorithm 2. It can be seen from the algorithm that the same BLM module is repeatedly used but with different inputs to detect the different markings that are needed to identify

Algorithm 2 Algorithm for horizontal BLM detection

- 1: $S_L = blmDetect(ROI_A, 20^\circ to 70^\circ, 10 to 15)$
 - 2: $S_R = blmDetect(ROI_B, -20^\circ to -70^\circ, 10 to 15)$
 - 3: $ROI_C = f(S_L, S_R)$
 - 4: $S_H = blmDetect(ROI_C, -10^\circ to 10^\circ, 5 to 15)$
-

Figure 3.26: Illustration of horizontal marking detection steps.

horizontal BLMs. Therefore, the proposed method allows for reusability of resources to detect horizontal markings.

In order to establish whether the horizontal marking detected is dashed or solid, a spatial continuity check is applied, in a similar manner as demonstrated for lane markings in Sec. 3.10.1. The only difference is that here, the mask is horizontal in the region where the marking is detected. It is to be noted that the temporal continuity check that was described in Sec. 3.10.2 for longitudinal markings, is not relevant in the case of horizontal markings. The reason is that the longitudinal markings run along the road, and hence their presence or absence in the near view can be an indicator regarding the type of longitudinal marking, when studied across frames. However, this is not the case with horizontal lines.

Performance evaluation: The proposed method for horizontal BLM detection was applied on the image dataset described earlier in Sec. 3.6. 25 image sequences containing horizontal markings and another 25 of them without horizontal marking were chosen to evaluate the proposed algorithm. It is seen that the algorithm successfully

3. Basic Linear Road Markings

detects the horizontal lines for the image sets considered. One such sequence is shown in Fig. 3.27. The frames are arranged from left to right and top to bottom. The result of the detection is indicated on the top left corner in every frame. It can be seen from the first three images in the first row in Fig. 3.27 that when the horizontal marking is in the far-view (i.e. not inside the last two rows), the algorithm does not identify the horizontal BLM. From the fourth frame onwards, the horizontal line enters the RoI meant for horizontal BLM detection. It can be seen that the algorithm is able to identify the BLM accurately in the rest of the frames. Computationally, the same BLM module is reused for the different BLMs that are needed to detect the horizontal BLMs. The computations required for detecting the RoIs using the equation (3.12) are simplified in the forthcoming sections.

Figure 3.27: Results of horizontal marking detection on sample sequence.

3.12 Summary

In this chapter, computationally efficient techniques for detection and classification of BLM were proposed. Firstly, an efficient edge based feature extraction step using histograms of gradient angles was proposed, that generates customized edge maps containing only straight line edges, which are our primary features of interest. Next an

outlier removal step, using the Hough Transform, signs of edges and priori information about lane markings is formulated, that helps to prune out noise and qualify only the lane marking edges. This method of lane marking extraction was evaluated on over 6,000 image frames, captured in various complex scenarios such as tree shadows and presence of other artifacts resembling lane markings. The algorithm was shown to give a high average detection accuracy of over 98% for the 20 image sequences considered. A temporal tracking method was then proposed to further improve robustness of the lane marking detection method, and to also generate the lane RoI in cases where the near view does not contain lane markings. A generic repeatable function called `blmdetect()` was defined to detect all BLMs in a given region of interest. Next, techniques to distinguish between solid and dashed lane markings were presented using a spatial and temporal continuity analysis. This was shown to yield an accuracy of 100% on the validation set. Finally, the `blmdetect()` module was extended to detect horizontal line markings, which was shown to successfully detect the horizontal markings in all the 25 videos considered for validation.

In the next chapter, an architecture for basic lane marking detection is presented.

3. Basic Linear Road Markings

4

Architecture for BLM Detection

4.1 Introduction

In the previous chapter, robust techniques were proposed to detect BLMs and distinguish them in complex road scenarios. In this chapter, an architecture for BLM detection is proposed, bringing together the various computational blocks that were proposed in the previous chapter. Further, the most compute-intensive block is identified to be the Hough Transform (HT) module, thus setting the motivation to accelerate it. Next, in order to overcome the computational bottleneck of HT, a novel parallel HT architecture (called Additive Hough Transform or AHT) is proposed based on the additive property of HT that was identified. It will be shown that the AHT leads to significant reduction in the complexity of computation as it replaces trigonometric computations with mere additions, leading to acceleration of the feature detection process. Finally, a detailed discussion on the computational savings that are achieved using the proposed BLM detection method is presented.

4.2 Architecture for Detecting and Distinguishing BLM

The computational blocks involved in the entire process of BLM detection are outlined in Fig. 4.1. The initial inputs, final outputs and intermediate outputs are indicated in boxes with dashed outlines. The computational blocks are shown in gray boxes. It may be recalled from Chapter 3 that the first step in the detection of BLM is the extraction of linear features. This was done by constructing gradient angle histograms (GAH) to generate edge maps containing primarily straight line edges, called Straight

4. Architecture for BLM Detection

Figure 4.1: Computational blocks in linear marking detection.

Line Edge Maps (SLEM). Next, in order to promote straight lines that belong to the BLM and filter outliers, properties of BLM are used, such as expected RoI, angle of orientation and expected thickness of BLM. It is seen that all these are fed in as inputs into the BLM Detect Module. Using these priori data, the Hough Transform is applied on the left and right edge maps, after which a peak detection is performed on the Hough space. The peaks in the Hough space satisfying the thickness criteria are output as the lane marking parameters (ρ_l, θ_l) and (ρ_r, θ_r) . The function `blmDetect()` is a generic function defined earlier in Chapter 3 to detect BLMs, that captures the inputs and outputs of the entire BLM detection process. Fig. 4.2 shows how the proposed `blmDetect()` module is cascaded in different ways alongwith the temporal and spatial continuity checks to detect and distinguish the different BLMs.

4.3 Accelerating Hough Transform for Increased Throughput

It has been established in literature that Hough Transform (HT) is one of the computationally intensive modules in the context of lane marking extraction [162], [163]. Speeding HT computation will improve the overall throughput of the system. In this section, a novel parallel Hough Transform computation method is proposed, which drastically collapses the total computation time as compared to the conventional HT

4.3 Accelerating Hough Transform for Increased Throughput

Figure 4.2: Cascading linear marking detection modules for detecting longitudinal & transverse BLMs.

approach. The Hough Transform is explored to derive an additive property. The additive property is illustrated on a sample image and it is further exploited to develop a parallel grid-based Hough Transform computation method called Additive Hough Transform (AHT). We discuss the inherent parallelism of AHT for hardware realization and the hardware costs are analyzed. The proposed AHT is compared against existing Hough Transform computation methods for total computation time using one of the benchmark images.

4.3.1 Additive Property of Hough Transform

The additive property of the Hough transform is defined by the following theorem. *Theorem 1:* The Hough Transform (HT) of point A with respect to the image origin ‘O’ is equal to the sum of the HT of A with respect to another point B and HT of B with respect to O , i.e.

$$HT(A, O) = HT(A, B) + HT(B, O) \quad (4.1)$$

4. Architecture for BLM Detection

where $HT(X, Y)$ represents the Hough Transform of point X with respect to point Y .

Proof: Considering a line l passing through a point A , the perpendicular from origin O onto l is $\rho_O(A, \theta)$, where θ is the angle made by the perpendicular with the x-axis (Fig. 4.3).

Now, let us consider another point B in the image. The perpendicular projection onto l from B is $\rho_B(A, \theta)$, as shown in Fig. 4.3. Also, for a line m passing through B such that $m \parallel l$, $\rho_O(B, \theta)$ is the perpendicular projection onto m from O . Using the properties of parallel lines, it can be seen that the angle remains the same for all the three projections, i.e. $\rho_O(A, \theta)$, $\rho_B(A, \theta)$ and $\rho_O(B, \theta)$.

If B is considered as a local origin, $\rho_B(A, \theta)$ is the HT of A with respect to B for a given angle θ , i.e.

$$\rho_B(A, \theta) = HT(A, B). \quad (4.2)$$

$\rho_O(B, \theta)$ is the HT of B with respect to the origin O for the same angle θ as shown by the following equation.

$$\rho_O(B, \theta) = HT(B, O) \quad (4.3)$$

Figure 4.3: Deriving the additive property of Hough Transform.

From Fig. 4.3, it can be seen that

$$\rho_O(A, \theta) = \rho_B(A, \theta) + \rho_O(B, \theta) \quad (4.4)$$

In (4.4), $\rho_O(A, \theta)$ is the HT of A with respect to O for the angle θ . Using (4.2) and (4.3), (4.4) can be rewritten as,

$$HT(A, O) = HT(A, B) + HT(B, O) \quad (4.5)$$

4.3 Accelerating Hough Transform for Increased Throughput

This means that the Hough Transform of a point with respect to a global origin can be broken into two parts - the HT of that point with regards to a local origin and the HT of the local origin with respect to the global origin. This property can be also be proven using coordinate geometry and trigonometric relations, using equations similar to (1).

In order to verify the additive property and show its equivalence to the conventional HT computation, it is applied on a test image. In order to apply this property on an image, the edge map of size $m \times m$ is first divided into k^2 blocks using a $k \times k$ grid, each block comprising $(\frac{m}{k})^2$ pixels. We consider m to be divisible by k but the proposed method can be extended for other cases also. The origin of the edge map is taken to be the leftmost and the bottommost pixel. Similarly, in every block, the leftmost and the bottommost pixel is taken as the *local block origin*. The Hough Transforms of pixel positions with respect to local block centers are referred to as ‘local HT’ and those with respect to the origin of the edge map as ‘global HT’. This is illustrated in the test image shown in Fig. 4.4.

Figure 4.4: Hough Transform computation (a) input edge map, (b) Hough Transform computed using standard equation (3.4).

Let us consider an 8×8 edge map in Fig. 4.4(a). Pixel O , with coordinate position $(0,0)$ is taken to be the global origin of the image. The shaded pixels refer to the edge pixels: $A(3,1)$, $B(4,2)$, $C(5,3)$, $D(6,4)$, $E(6,5)$ and $F(6,6)$ with respect to O . The Hough Transform of this edge map, as computed using the standard equation in (3.4) is shown in Fig. 4.4(a). The angles are taken from 0 to π in steps of $\frac{\pi}{8}$. It can be seen from Fig. 4.4(a) that the sinusoids intersect at two points in the Hough space at

4. Architecture for BLM Detection

$\theta = \frac{3\pi}{4}$ and $\theta = \pi$ or 0. These intersections represent the straight lines - $ABCD$ and DEF , respectively.

Now, let us consider a 2×2 grid symmetrically dividing the edge map into four blocks of 4×4 pixels each, as indicated by the bold lines in Fig. 4.4(a). The local origins of each of these blocks are given by the leftmost and bottommost pixels at $(0, 0)$, $(4, 0)$, $(0, 4)$ and $(4, 4)$. For every edge pixel, we now compute the local HT based on their positions relative to the local origins. For example, C has a coordinate position of $(1, 3)$ with respect to its local origin, $(4, 0)$, and its local HT is calculated based on this.

In Fig. 4.4(b), we illustrate the Hough space generated using the additive property. The first column in Fig. 4.4(b) shows the local HTs of each of the edge pixels. The second column shows the plot of the global HT of the local origins corresponding to each edge pixel. The curves in the third column are obtained by adding the plots in the first and second columns, i.e. the summation of the local HT of edge pixel and global HT of local origins. These resultant plots of every edge pixel are combined together to obtain the Hough Transform of the entire edge map. It can be seen from the combined plot that it matches with the reference Hough Transform plot in Fig. 4.4(a), illustrating the validity of this approach. Intersections at $\theta = \frac{3\pi}{4}$ and π are seen which represent the same straight lines - $ABCD$ and DEF .

4.3.2 Parallel Additive Hough Transform

The additive property of Hough Transform is employed to propose a block based computation technique - Additive Hough Transform (AHT), which can compute the HT of an image in a parallel manner. Unlike conventional methods which parallelize the computation of HT for multiple angles only, the proposed method also parallelizes the HT computation for multiple edge pixels in the image.

In order to do this, we first divide the input edge map into smaller images, referred to as ‘blocks’, using a grid and then process each of these blocks independently in parallel by computing the local and global HTs as shown below.

Computing Local HT:

Consider a 2×2 grid to divide an 8×8 sized edge map into 4 blocks, as shown in Fig.

4.3 Accelerating Hough Transform for Increased Throughput

Figure 4.5: Illustrating blocks, local origins and relative positions of pixels in blocks.

4.5. Blocks $b(0,1)$ and $b(1,0)$ have their local block origins as $O_{0,1}$ and $O_{1,0}$ respectively. Let us take a point P in block $b(0,1)$ and another point P' in block $b(1,0)$ with the same relative positions with respect to their local origins. Due to the same relative positions, the HT of P with respect to $O_{0,1}$ is the same as the HT of P' with respect to $O_{1,0}$. Similar arguments can be extended to any other pair of points with the same relative position, like Q in $b(0,1)$ and Q' in $b(1,0)$. This means that for computing the local HT of points in all the blocks of the grid, it is enough to process just a *single* block.

Computing Global HT:

With the local HTs computed, the global HTs of these points can be computed using the additive property in Theorem 1. In other words, the local HTs of the points that were just obtained can be added to the global HTs of their corresponding local origins resulting in the global HTs of the points. The global HTs of the local block origins can be precomputed as the coordinates are known beforehand. To summarize, for any pixel position P in the edge map lying in block $b(i,j)$, we can say that:

$$HT(P, O) = HT(P, O_{b(i,j)}) + HT(O_{b(i,j)}, O) \quad (4.6)$$

where $O_{b(i,j)}$ is the local origin of block $b(i,j)$.

AHT allows for parallel computation of Hough Transform in hardware. As explained earlier, the edge map is divided into k^2 blocks. The global HTs of the local origins of these blocks are precomputed. The local HTs of $\left(\frac{m}{k}\right)^2$ pixel locations in a *single* block are also precomputed and stored. An array of k^2 processing elements (PEs) is designed where each PE comprises a simple two-operand adder for each of the k^2 blocks. Let

4. Architecture for BLM Detection

(x, y) be a coordinate with respect to the local block origin $O_{b(i,j)}$ where i and j are used to index the blocks in the grid and $0 \leq \{i, j\} < k$. The range of x and y is limited by the size of the block resulting in $0 \leq \{x, y\} < \frac{m}{k}$. k^2 pixels, one from each block, located at the same coordinate position (x, y) with respect to their local origins are accessed from the edge map in parallel. As shown previously for the same relative position (x, y) , all the k^2 pixels have the same local HT value, $HT(P(x, y), O_{b(i,j)})$. This value is accessed from the memory and added to k^2 precomputed global HTs of the local origins of the k^2 blocks, i.e. $HT(O_{b(i,j)}, O)$, using the two-operand adders in the PE array. Thus, at the end of each cycle, the Hough Transform of k^2 pixels is computed simultaneously. The following expression summarizes parallel AHT computation.

$$HT\left(P\left(x + i\left(\frac{m}{k}\right), y + j\left(\frac{m}{k}\right)\right), O\right)_{i,j=0,1,\dots,k-1} = HT(P(x, y), O_{b(i,j)}) + HT(O_{b(i,j)}, O)$$

The computational efficiency of the proposed AHT technique in terms computation complexity and timing is discussed in the next section. The use of AHT to reduce computation cost of finding the masks is also discussed.

4.4 Computational Cost Analysis

The step-by-step use of *SLEM*, signed edges and HT together for detection of BLM lead to a substantial reduction in the computational complexity, as explained below.

(a) Reduced Number of Edge Pixels for HT using SLEM: One of the main advantages of using the proposed *SLEM* for feature extraction is that it drastically reduces the number of edge pixels that are sent for Hough Transform, as compared to performing a conventional edge detection, thereby leading to significant computational savings. Table 4.1 compares the number of edge pixels in the proposed *SLEM* against the the conventional Sobel edge map for the three images that were considered earlier. It can be seen that the *SLEM* for *Chimney* image has about 85% lesser number of points for HT computation as compared to its Sobel counterpart. From Fig. 3.5, this image visibly has the maximum amount of unwanted non-straight line edge pixels like the foliage of the tree and the texture of the wall, which are eliminated leading to large savings. *House* contains mainly straight lines and hence the savings is not as high as in *Chimney*, although it still leads to a substantial reduction of voting points by 53 %.

This shows that the reduction in edge points in the *SLEM* is more evident in those images that contain more noise and curved edges. These are unwanted features for the straight line Hough transform and are removed using the proposed gradient angle histograms.

Table 4.1: Reduction in total edge pixel count using *SLEM*

Image	Sobel	<i>SLEM</i>	% Savings
Chimney	19321	2805	85 %
Keys	7247	2137	70 %
House	7950	3735	53 %

(b) Reduced Angle Space for HT using *SLEM* and Priori Information: The region of interest and expected orientation of the BLM as captured by the camera in the host vehicle can be estimated, depending on the types of BLM. This priori knowledge helps to reduce computations involved. For example, knowing the range of possible angles of line edges that bound lane markings, greatly reduces the computations of HT. Not having the knowledge of the angles of the lines means that HT needs to be repeated for each of the angles in the angle set. The proposed method further narrows down the angle range drastically for the HT computations through the use of gradient-angle histograms *GAH* in *SLEM*. This is because the location of the peaks in the *GAH* indicates the angle of orientation of the potential lines in the image. Once the *SLEM* and signed edge map are generated, the next step of processing using HT is done selectively only for a small angle window, around those angles that indicate a peak in the *GAH*.

(c) Optimized Peak Detection using *SLEM*: *SLEM* also results in cleaner Hough spaces, which aids in the peak detection operation at the end of Hough voting. Fig. 4.6 shows the Hough spaces generated by applying the HT on the Sobel edge maps and *SLEM*s of these images, using MATLAB. The intensity of the gray level is directly proportional to the relative peak heights of the values in the $\rho - \theta$ array in the Hough space. That is, a distinct peak with low neighboring values is represented by shades close to white, while a peak that is not so distinct because of surrounding noisy peaks, is denoted by values of lower intensity. Also, denser plots reflect higher number of votes, which in turn means higher computational costs for Hough voting. It can

4. Architecture for BLM Detection

be seen that the Hough spaces of the *SLEM* for *Chimney*, *Keys* and *Pillars* images are less dense than their Sobel counterparts. This implies lesser number of edge pixels have been transformed to the Hough space, thereby reducing the crowding effect. The Hough space of the *SLEM* of *House* image closely resembles that generated using Sobel because of a close resemblance between the *SLEM* and the Sobel edge maps. It can also be seen that the peak positions in all four Hough spaces are identical as compared to their corresponding Sobel Hough spaces, verifying the correctness of the proposed approach. Moreover, the peaks are brighter when the *SLEM* is used indicating distinct peaks and cleaner Hough spaces. It is noteworthy that the proposed *SLEM* can now be further processed by any of the Hough voting techniques like RHT [111], PHT [119], PPHT [120], gradient angle aided PPHT [121] and O' Gorman's HT [118]. The *SLEM* aids these techniques by providing a cleaner edge map for them to operate on, thereby reducing the number of HT computations even further. For example, if we apply PHT directly on *Chimney*'s Sobel edge map with an edgel sampling rate of 1 in 5 edgels, a total of 3864 LHT computations are necessary. This number reduces by 85% if PHT is applied on the *SLEM* with the same sampling rate.

(d) Further HT Optimization using Signed Edges and Thickness Criteria:

It was explained above how the use of *SLEM* generates a reduced edge map that leads to cleaner Hough spaces, and hence a lighter Hough peak detection step. The peak detection process is further simplified by the use of signed edges in two ways. Firstly, the number of edge pixels on which HT is performed is further reduced as only the left edge map, which is a partial edge map is sent to the HT, and a cleaner Hough space leads to a more optimized peak detection. Secondly, only for those angles which give a peak in the Hough space of the left edge map, do we check for peaks in the corresponding Hough space of the right edge map. The rest of the Hough space are simply ignored. Moreover, since we know the estimated separation between the left and right edge of the lane markings, the peak position in the Hough space of the right edge map can be limited to a very small set of values.

(e) Computational Savings Using AHT: As it was shown in Sec. 4.3, with the use of AHT, the trigonometric operations that form the basis of HT are replaced by simple addition operations. Moreover, due to the additive property, the Hough Transform

of k^2 pixels can be computed simultaneously, thereby lending well for parallelization. Here, we describe the acceleration in total computation time achieved by AHT and briefly discuss the tradeoffs in area cost also. The total computation time of existing architectures, T' , in [114, 144, 145, 146] can be generalized to the following expression:

$$T' = (1 + p\alpha) m^2 \quad (4.7)$$

where p is the ratio of number of edge pixels to size of edge map and α refers to the additional time required for computation if a pixel happens to be an edge pixel. From (4.7), for an image with no edge content at all, T' is equal to m^2 . This minimum value of T' is necessary for reading the edge map serially. Moreover, T' increases as the edge content p increases. T' varies proportionally with p and α where α depends on the processing element used in the computation. For example, in the case of a pipelined CORDIC based implementation in [144, 145, 146], α depends on the number of angles in Hough space. In the case of an iterative CORDIC based implementation [144, 145], in addition to the number of angles in Hough space, α depends on the number of CORDIC iterations. In the repeated addition based architecture in [114], α is equal to 1. In all these architectures, in addition to m^2 cycles for reading the image, T' has an additional component for computing the Hough Transform of the edge pixels.

The total computation time of the proposed AHT technique T_{prop} in hardware is given by

$$T_{prop} = \left(\frac{m}{k}\right)^2 \quad (4.8)$$

where k is the grid size. The local HTs of points in a block and global HTs of local block origins are precomputed offline and hence they do not contribute any additional delay. Also, reading them from the memory can be pipelined without incurring any additional latency cycles. The computational delay of each processing element is equal to the delay of a two-operand w -bit adder where w is the wordlength local and global HTs.

From (4.7) and (4.8), we see that AHT shows a speed up, given by $\frac{T'}{T_{prop}}$, of atleast k^2 against existing methods. This speed-up is possible due to the inherent parallelism of the proposed AHT method. Increase in k means more number of blocks are being processed in parallel, thereby reducing T_{prop} . However, k also determines the area cost of the implementation. A higher value of k means results in more number of adders in the computation array discussed in Sec. 4.3.2. Also, the space for storing global HT of

4. Architecture for BLM Detection

k^2 local origins increases. On the other hand, space for storing precomputed local HT of points within a block reduces because the block size is given by $(\frac{m}{k})^2$ pixels. Thus, k determines the area-time complexity of the proposed method.

Figure 4.6: Comparison of Hough spaces; Hough spaces for Sobel edge maps (first row) and *SLEM* (second row) for (a) Chimney, (b) Keys, (c) House are shown.

The proposed AHT technique is demonstrated on a 128×128 sized real image in Fig. 4.6(a) with an edge content of 5.19%, i.e. $p = 0.0519$. From Fig. 4.6(b) and (c), it can be seen that the Hough space obtained using the proposed AHT is identical to the one obtained by the conventional method. Table 4.2 lists the total computation time (in cycles) for Hough Transform computation using AHT and existing methods for the edge map in Fig. 4.6(a). The existing techniques require 128^2 cycles to read the edge map. In addition, every edge pixel requires α additional cycles as shown in (4.7). For all CORDIC based architectures in [144, 145, 146], α depends on the the number of iterations, n_c , and the number of angles in Hough space, n_θ . α is equal to $\frac{n_c n_\theta}{2}$ for the iterative CORDIC based architecture [144]-[145], $\frac{n_\theta}{2}$ for pipelined CORDIC based methods in [144, 145, 146] and one for adder based Hough Transform computation in [114]. To compute T in Table 4.2, n_c and n_θ are taken as 16 and 180 respectively. It can be seen from the table that the proposed AHT outperforms the existing techniques by atleast 67 times. This speed up is because of the inherent parallelism of Additive Hough Transform. The speed up increases further as the parallelism indicated by k increases (4.8).

Next, we discuss the area overheads of the proposed AHT. Considering that the proposed AHT is a parallel method and others [114, 144, 145, 146] are not fully parallel

architectures, a comparison of area cost alone is not appropriate. Therefore, we have analyzed the Area-Time product (AT) cost of AHT and compared it with two existing methods. As discussed earlier, the value of k also determines the area cost of AHT. Increasing k increases the parallelism and this generally increases the area cost. We choose the serial addition and angle parallel method in [114] for comparison. In this method, $n_\theta = 180$ additions are performed in parallel. Therefore, the area cost of this method is 180 additions. In the proposed method, for a grid size of $k = 8$, $k^2 \times n_\theta$ constant-adders are required. The area cost of a constant-adder is half the cost of normal-adder as shown in [116]. Therefore, the area cost of AHT is $k^2 \times n_\theta/2$. For $n_\theta = 180$ and $k = 8$, the area cost is 5760 additions. The AT product cost of AHT and [114] using the above area costs and timing values in Table 4.2, are equal to 1474560 and 3102120 respectively. This shows that the AT cost of the AHT is less than half of [114]. Similarly, the AT cost of iterative CORDIC based HT implementation in [144] is found to be 3381376, which is 2.29 times the AT cost of AHT.

Table 4.2: Comparison of total computation time

Design	T Cycles
Proposed AHT based	256
Iterative CORDIC based [145], [144]	1,240,008
Pipelined CORDIC based [145], [144]	105,668
Pipelined CORDIC based [146]	105,668
Repeated addition based [114]	17,234

(f) Mask Generation using AHT: The inequalities in (3.9) and (3.12) that are used to determine the masks for distinguishing BLMs and detecting horizontal BLMs involving repeated computing of the HT equation for the pixel coordinates in the block. As seen from (3.9) and (3.12), the trigonometric operations and multiplications are time consuming operations and they need to be performed for every x and y coordinate till the inequalities are met in a block. These computations can be completely eliminated by employing the AHT in the following way. Let us consider (3.9) to generate the mask for left lane defined by (ρ_L^l, θ_L^l) and (ρ_L^r, θ_L^r) in block $B(i, j)$. Therefore, ρ_L^l and ρ_L^r denote the global HTs of left and right edges of the left lane. Let $HT(O_{B(i,j)}, O)_{\theta_L^l}$ and $HT(O_{B(i,j)}, O)_{\theta_L^r}$ be the global HTs of the origin of $B(i, j)$ for the angles of the left and right edges of the left host lane. Following the additive property of HT, the local

4. Architecture for BLM Detection

HTs of the left and right edges with respect to the block origin are given by a simple subtraction as given below:

$$\rho_L^{l'} = \rho_L^l - HT(O_{B(i,j)}, O)_{\theta_L^l} \quad (4.9)$$

$$\rho_L^{r'} = \rho_L^r - HT(O_{B(i,j)}, O)_{\theta_L^r} \quad (4.10)$$

Given the local HTs of the left and right edges, a simple check needs to now be performed on the LUT of local HTs that have already been precomputed for AHT computation, to find which coordinates have ρ values greater than and lesser than $\rho_L^{l'}$ and $\rho_L^{r'}$, respectively. All the pixel coordinates that satisfy this comparison will be inside the mask. In this manner, the AHT has eliminated all the HT computations for mask generation.

4.5 Summary

In this chapter, an architecture that captures all the computational steps to detect and classify BLM is presented. The Hough Transform (HT) is identified to be one of the major computationally demanding steps, thus setting the motivation to accelerate it. A novel parallel HT algorithm, based on the additive property of HT is proposed that leads to orders of magnitude speedup in the HT computations. A computational cost analysis is presented along with a comparison with other existing HT methods to demonstrate the acceleration in the proposed AHT. It was shown that the proposed methods are highly computationally efficient because of (a) reduced Number of Edge Pixels for HT using SLEM (b) reduced angle space for HT using *SLEM* and priori information (c) optimized peak detection using *SLEM* (d) HT optimization using signed edges and separation criteria and finally (e) computational savings using the proposed parallel AHT instead of the conventional HT.

In the next chapter, complex linear road markings will be studied and the underlying challenges in detecting them will be identified, followed which efficient methods to detect them will be proposed.

5

Complex Linear Road Markings

5.1 Introduction

Complex linear road markings (CLM) refer to road markings that comprise of two or more basic linear markings, placed in a certain orientation. These could either be double linear markings, in the longitudinal or transverse directions, or zigzag markings that run along lane boundaries in the longitudinal directions. Examples of complex linear markings are shown in Fig. 5.1. As it can be seen from Fig. 5.1, a double linear marking consists of two basic linear markings running in parallel at close proximity with each other, while a zigzag marking has multiple basic linear marking segments placed in a zigzag orientation. They are referred to as ‘complex’ as they involve detection of multiple lines, that are close to each other or of varying lengths. The challenges involved in detecting such markings will be addressed in this chapter.

Given that the underlying building blocks of complex linear road markings are a set of basic linear marking segments, it is expected that all the properties of BLM outlined in Chapter 3 also apply to the component linear segments that make up the CLM. Specifically, linear segments of CLM satisfy the following properties: they have linear edges ($P1$), occur at predictable locations($P2$), each linear edge has a corresponding opposite signed edge counterpart ($P3$), satisfy a predictable thickness criteria ($P4$) and exhibit temporal continuity ($P5$) as no sudden movements are expected in the vehicle motion. It is therefore expected that techniques proposed in Chapter 3 for detecting basic linear markings can be extended for complex linear markings too. Section below first explains the attempt to extend the previously proposed techniques for detecting

5. Complex Linear Road Markings

Figure 5.1: Complex Linear Road Markings.

complex linear markings, and the issues encountered.

5.2 Challenges in detecting CLM

It was envisaged that the techniques previously proposed for detecting BLM can be reused to first detect the component linear segments that make up the CLM, and once these linear segments are detected, rules can then be applied to associate them, to check for the presence of the zigzag or double line. However, there were a number of challenges encountered, as explained below. As per Fig. 3.19 in Chapter 3 which summarizes the various steps involved in BLM detection, the input frame is subject to SLEM generation in a block-based fashion, followed by splitting the edge map into signed edges, and these signed edges are fed into the Hough Transform to identify the positions of the linear markings. The same set of steps were applied to test images containing zigzags and double linear markings, with an objective to first isolate its component linear markings.

However, although the edge map generation step using SLEM was shown to be robust and extracts the edges of interest, it was observed that the conventional peak detection used in the Hough Transform step is not effective to detect all the lines of interest. This is illustrated on a sample block shown in Fig. 5.2 (b) for a road scene with a zigzag marking, shown in Fig. 5.2 (a). The left edge map of this block, E_{SLI} , created by splitting the SLEM into signed edges is shown in Fig. 5.2 (c). This is then sent into the Hough Transform and the Hough space generated thereafter is shown in Fig.

Figure 5.2: Issues with conventional Hough peak detection for detecting Zigzag.

5.2 (d). The peak detection is then performed using the conventional peak detection algorithm inbuilt in MATLAB, where the threshold is set to be $0.5p_{max}$, where p_{max} is the highest peak detected. A non-maximal suppression is then performed to suppress the neighborhood of the valid peaks detected. When the number of lines was not specified as an input parameter, then the only 1 Hough line is detected up as shown in Fig. 5.2 (e). Since there are multiple lines in this image, this parameter was reset to 5, and the resulting Hough lines are shown in Fig.5.2 (f).

It can be observed that this approach to peak detection misses out detecting some of the important line features that are critical for defining the zigzag. The zigzag marking that is captured in the block consists of three linear segments, out of which only one of them is picked up, when the default number of peaks is not specified. Increasing the cap on the number of peaks to 5 detects Hough peaks that are redundant, representing the same line, instead of detecting the three different lines as separate peaks. The reason

5. Complex Linear Road Markings

for this is primarily because the linear segments that make up the zigzag markings appear to be of different lengths, due to perspective. Setting the right threshold in such cases becomes a challenging task. The reason is that if the threshold is taken as a fixed percentage of the maximum peak height, as is done in many conventional approaches, then the likelihood of smaller valid peaks getting missed out is very high (as was illustrated above). Setting the threshold to a low value to ensure no peaks are missed out, has the side effect of noise and false peaks. The false peaks are many times due to neighboring peaks that surround a valid peak, thereby giving a false count.

Typically, neighboring peaks around a valid peak are eliminated through a non-maximal suppression step, by choosing an appropriate window size. However, as shown in Fig. 5.2, despite the non-maximal suppression, a number of peaks are left unsuppressed.

Similar issues of missing lines and duplicate lines were encountered when the BLM detection techniques were applied to images containing double linear markings. Consider an example in Fig. 5.3 (a), of an image block containing a double linear marking. The left edge map, its corresponding Hough space and lines detected are shown in Fig. 5.3 (b), (c) and (d), respectively.

The reason for missing peaks in this case is largely due to the proximity of the lines in the image block, which was not so much of an issue for basic linear markings. As seen in the Hough space in Fig. 5.3 (c), the Hough space of such closeby lines shows the two peaks very close to each other. Due to the non-maximal suppression step, there is a tendency to suppress valid peaks that surround another valid peak, as was shown in this example. It is to be noted that this becomes an issue mainly in blocks slightly away from the camera. For cases in the immediate near view, the proposed splitting of the edge map into signed edges mitigates this issue to a large extent, as it separates the closeby lines into two independent edge maps. This is explained more in detail in the next section.

To summarize, what makes detection of complex linear markings into a more challenging problem when compared to basic linear markings is primarily the following two issues.

- Lines of varying length present in a zigzag: When lines of different length are of interest within a given block of interest, then setting the right threshold poses to

Figure 5.3: Issues with conventional Hough peak detection for detecting double linear marking.

be an issue. Having high thresholds would yield false negatives, while lowering the threshold would introduce noise and unnecessary peaks.

- Close proximity between linear edges in double markings: Since double markings occur in pairs of linear segments, a minimum of 4 closeby lines are present in close proximity to each other. Conventional Hough peak detection methods tend to suppress closeby peaks in an attempt to do non-maximal suppression, thereby leading to false negatives.

In the next section, we propose an adaptive Hough peak detection algorithm as a solution for the above issues.

5.3 Adaptive Hough Peak Detection (AHPD)

The motivation for the proposed AHPD is the need to detect peaks in the Hough space, in an adaptive way based on properties of valid peaks and noise in the Hough space. As shown in the previous section, Hough space can be very noisy and complicated, and poses challenges in detecting the lines of interest. There is limited work in the area of Hough peak detection [156, 157, 158, 159]. One of the common methods used is the non-maximal suppression method, that is used in MATLAB. This function is called `houghpeaks()`. In this method, the maximum of the Hough space is computed in an iterative way. In every iteration i , the position of the maximum in the Hough space gives the line $l_i(\rho_i, \theta_i)$. In the next iteration $i + 1$, a given neighborhood around the previous peak at (ρ_i, θ_i) is suppressed and the maximum is found in the updated Hough space to give $l_{i+1}(\rho_{i+1}, \theta_{i+1})$. This process is repeated till the maximum number of peaks is reached (this parameter is defined by the user). Therefore, in MATLAB `houghpeaks()`, there are three parameters that need to be defined by the user depending on the input image - the number of peaks, the neighborhood for suppression and minimum threshold. Depending on these parameters, the `houghpeaks` function either gives too many lines or too little lines. Therefore, this technique is dependent on the user inputs and the input image, both. This was illustrated on the road marking images in the previous section, where either too many lines are picked for the same edge or shorter edges are missed out in the presence of longer edges, if the parameters of `houghpeaks()` are not set appropriately.

In order to tackle these challenges, other more context-aware methods are proposed. In some of these methods, the Hough space itself is modified in such a way that the peaks are amplified like the modified O' Gorman method [160], which increases the accumulation count by a weighted value of the strength of the edge pixel. However, such methods need additional processing in the HT computation stage. Also, they are not generic to any kind of linear edges present in an edge map, i.e. these methods work in scenarios, where the kind of lines (like the orientation) to be picked, is a given input. There are some methods which have been proposed using the patterns formed by the peaks in the Hough space. [156] gives a detailed description of this method. The Hough peaks have been observed to have some specific patterns called butterflies as shown in 5.4 [156]. The method proposed in [156] detects these butterflies using a

Figure 5.4: Butterfly pattern of Hough peak as shown in [156].

template matching method. However, the computational complexity of this method can be high depending on the number of lines that need to be detected. Also, this method is not fully automated as it requires a set of parameters about the lines. In addition, the Hough space is processed to determine some of the parameters used in this algorithm. Hough space can be very noisy depending on the input image and any inaccurate or over or under processing of the Hough space due to the varied nature of the Hough space can result in increased computational complexity.

In this section, a computationally efficient and adaptive method is proposed that uses the peak properties of the Hough space and the butterfly patterns to detect peaks in a given Hough space. We first outline the main properties of the peaks in Hough space, that were observed in different kinds of Hough spaces. These properties will then be used to propose the AHPD algorithm.

5.3.1 AHPD Algorithm

A detailed study of the Hough spaces was conducted to understand the nature of Hough peaks and the noise around the Hough peaks. Let us consider an $n_\theta \times n_\rho$ Hough space, denoted by $\mathcal{H}$, where for every θ we have an n_ρ long accumulation array, represented by H_θ . Therefore, bringing the n_θ H_θ arrays results in $\mathcal{H}$, the 2D accumulation space. It was observed the overall peak in the Hough space is a combination of a peak formation into different directions. Firstly, the peak in $\mathcal{H}$ is contributed by peak formation in

5. Complex Linear Road Markings

each individual accumulation array H_θ for every θ . Next, there is a peak pattern that is observed across H_θ s, i.e. the overall peak is formed by a specific pattern formed by peaks that are obtained from every accumulation array of each θ in the Hough space.

Based on these observations, the proposed method decomposes the Hough peak detection process into the following two main steps:

- **Step 1:** First step is to decompose the 2D Hough space into 1D vectors, i.e. $\mathcal{H}$ is divided into n_θ 1D vectors, each vector corresponding to the accumulation values of each θ , i.e. H_θ . In each of these H_θ vectors, possible peak candidates are found. This step eliminates noisy peaks that interfere in the peak detection process, and highlights only those peaks that are forming the valid peaks in the 2D Hough space $\mathcal{H}$. The first step will result in a reduced Hough space $\mathcal{H}'$ that will have possible peak candidates that are selected in every H_θ . More details about this will be discussed in Sec. 5.3.1.1.
- **Step 2:** Second step is to finalize the valid peaks in $\mathcal{H}'$ that are obtained from Step 1. This includes checking the shape signature formed by individual peaks in every H_θ and determining the valid peak that is formed in the 2D Hough space. This step will be discussed in Sec. 5.3.1.2.

5.3.1.1 Stage 1: Shortlisting Peak Candidates

We first list the observations made on the one-dimensional H_θ s in Hough space along every θ as below:

- A peak in a Hough space is distinctly separable from the rest of the candidates in the Hough space. A peak in the Hough space is usually formed by accumulation counts in its neighborhood, i.e. the neighboring accumulation counts also help to form the peak.
- Noisy peaks occur as a clutter of peaks, i.e., if a candidate is amidst similar ranging candidates, then the whole set of candidates is considered as noise. In an image this would translate to small lines or edge pixels distributed randomly such that they all form small line segments. They all will then be considered as noise.

5.3 Adaptive Hough Peak Detection (AHPD)

Fig. 5.5 illustrates the two observations listed above. It can be seen that there is a valid peak in Fig. 5.5(a) whereas (b) shows a noisy Hough space, though the peaks appear to be peaking up.

Figure 5.5: Observations to illustrate valid peaks and noise (a) peaks are distinctly above clutter of noise (b) clutter of peaks means noise.

In the first stage of the algorithm, each H_θ is processed to eliminate noisy peaks and retain only the candidates that will eventually give a valid Hough peak, when seen in the 2D Hough space $\mathcal{H}$. Let H_θ be a set of values h_i , where h_i represents accumulation count of a ρ value indexed by i , ρ_i , in $\mathcal{H}$. A plot of a typical H_θ is shown in Fig. 5.5(a) & (b).

Given this plot, the vertical range represents the range of accumulation counts for that particular angle θ . This range is divided into bins and the number of *peaks* in each bin is determined. In other words, a histogram of peaks in H_θ , with respect to their peak heights is generated. This is done to capture the region of maximum activity. In other words, the bin with the highest count of peaks most likely represents the upperbound of the noisy peaks. This is usually the case with noisy small peaks that are found in the Hough space. Valid peaks, if present, will be above this bin.

So the peaks in H_θ are computed in the following way. A set of peaks, P_θ , is computed where $p_j \in P_\theta$ is given by:

$$P_\theta = \{p_j | (p_j = h_i) \Leftrightarrow (h'_i > 0) \wedge (h'_{i+1} < 0)\} \quad (5.1)$$

$$N_\theta = \{n_j | (n_j = i) \Leftrightarrow (h'_i > 0) \wedge (h'_{i+1} < 0)\} \quad (5.2)$$

5. Complex Linear Road Markings

The conditions listed above selects peaks from H_θ resulting in P_θ . N_θ gives the positions of the detected peaks in H_θ . Fig. 5.6 shows H_θ with the peaks detected using the algorithm listed above.

Figure 5.6: H_θ with peaks selected.

With the peaks selected, some initial criteria are set that are used to compute the noise band in H_θ . Noise band is defined as the range of accumulation counts in H_θ where there is maximum fluctuation between peaks and valleys. The proposed method adaptively selects out the noise band and sets the threshold for peak detection outside the noiseband.

Given the image size $m \times n$ and block size $b_s \times b_s$, the following criteria are set based on heuristic studies done on hundreds of samples of Hough spaces:

- Minimum separation (M_{sep}) is the desired minimum separation between the noise band and the lowest peak that is to be detected. This is a control knob which the user can set. This parameter ensures that what we consider as peaks are sitting above a minimum separation from the noise. If a noise band is found and there are peaks outside this noise band, we need to qualify them as valid peaks or noisy peaks. They will be qualified as noisy peaks, if they do not sit above this minimum separation. In this way, the stray noisy peaks which were not captured in the first round of setting the noise band are also eliminated. For our experiments, this is

5.3 Adaptive Hough Peak Detection (AHPD)

set to 1/25-th the block size by default. However, this can be preset to any value by the user depending on the level of noise removal the user wants. It should be noted that even if M_{sep} is set to a minimum value, say 0 so that the user does not want to take the risk of losing any valid peaks, the forthcoming steps of the algorithm rejects the noisy peaks which may be passed as valid peaks in the first step. It is to be noted that although M_{sep} can be set to zero by default, setting it to a non-zero value helps to reduce computations in the subsequent AHPD stages, by eliminating spurious peaks that are just about the first stage threshold.

- Minimum length (M_{len}) is the minimum length of the line which can be considered as a valid line segment and needs to be detected. This can be set by the user. If we want the algorithm to all lines that are above a particular minimum length, this particular parameter is set to that value. In our experiments, we set the minimum length to 1/16-th the block size by default.

With these parameters defined, the first check that the algorithm does, is to check P_θ if the maximum peak in P_θ is less than M_{len} . If so, then the entire P_θ is eliminated and no further processing is performed on P_θ because there is no valid peak greater than the minimum peak height. If one or more peaks in P_θ are found to be greater than M_{len} , the following next steps are performed.

In order to determine the noise band, a histogram of peaks is generated using the peaks set P_θ . In order to do this, we set a bin size for the histogram. We tried four bin sizes $s_3=1,2,3$ and 4 and it was heuristically found that the bin size $s_3 = 3$ was found to give the most optimal results. Setting a bin size of 1 or 2 was yielding too many local peaks, which is not desirable. Bin size of 4 was leading to merging of the valid peaks into bigger peaks, thereby losing spatial information of the individual peaks. Hence, the bin size was set to $s_3 = 3$ as this gave optimal results and helped to capture the peak height variations in the noise band. This however again be configured by the user depending on the block sizes and length of lines of interest, thereby allowing programmability of the algorithm.

With a bin size set to s_3 , the accumulation count axis is divided into $\lceil N_{max}/s_3 \rceil$ bins where N_{max} is the maximum accumulation count in H_θ . The histogram is generated by counting the peaks in each bin. Fig. 5.7 shows this binning process with $s_3 = 3$ as

5. Complex Linear Road Markings

the bin size in this example. The frequency counts (or the number of peaks) in each bin are listed on the left.

The *highest* bin with the highest frequency count is taken as the noise bin which becomes the starting point of noise band calculation. The arrow at accumulation count equal to 6 is in Fig. 5.7 shows this starting point.

Figure 5.7: Accumulation counts are divided into bins, each of size $s_3 = 3$. Dotted lines represent the bins and the number of peaks in each bin are listed on the left.

This band B_{max} gives the starting point to determine the noise band. The idea is to determine the fluctuation of noise around this band with maximum number of peaks. If there are valid peaks, they will be distinctly away from this particular band. So we check for the peaks above B_{max} satisfying M_{sep} . All the peaks that are above the $M_{sep} + B_{max}$ are considered for further processing. In Fig. 5.7, it can be seen that $B_{max} = 6$. However, there are peaks between 6 and 8, which do not satisfy $M_{sep} = 3$. Therefore, these peaks are not considered as peak candidates. The remaining peaks in P_θ , which are above $B_{max} + M_{sep}$ are considered as peak candidates and their surroundings are further analyzed.

For every given P_θ , we have marked the possible peak candidates that satisfy M_{sep} above the noise band. However, not all peak candidates are valid peaks in P_θ . Having found the possible peak candidates in H_θ , we now find if there any local maximas in the shortlisted peak candidates. As listed earlier in the properties of peaks, a peak

5.3 Adaptive Hough Peak Detection (AHPD)

in the Hough space is formed by the accumulation counts of its neighboring cells also. Therefore, there is a high likelihood that there could be local maximas that have been released as peaks above the noise band. However, these peaks are only forming the main peak in H_θ . In Fig. 5.7, it can be seen that the two peaks at 129 and 130 that have been selected as peak candidates satisfy the M_{sep} criteria. However, the peak at 130 is only a local maxima that is forming the main peak in H_θ . Therefore such local maximas need to be removed so that the computations in the next steps are reduced.

Before proceeding further, let us consider three possible cases for peaks as shown in Fig. 5.8. In all the three cases, the peak heights are the same and let us assume that they are selected possible peak candidates. We now have to decide which of the three cases can be considered as valid or invalid peaks, i.e. in each case, can we consider both the peaks as valid separate peaks that must be considered for further processing or is one peak a local maxima only? After careful observation of peaks, it was found that whether a peak can be considered as a valid candidate is dependent on its relative height and position with respect to its neighbors. It can be seen from Fig. 5.8 that in case (a), the lower peak is a local maxima and cannot be considered as a valid peak. In cases (b) and (c), both the peaks can be considered as valid peaks. Therefore, the distance between two peaks (as in case (b)) and the difference in the heights of the peak and the valley (as in case (c)) change the perception of the peak. In other words, if the distance and/or the height between peak and valley is greater than a particular value, we can consider both peaks that are separated by such distance or valley as valid peaks. In the next step, we find such separations to find valid peak clusters in the selected peak candidates in P_θ . Each peak cluster has one valid peak and some local maximae that will be forming the valid peak. In order to find these valid peaks from the selected peak candidates, we apply the following operations.

Using (5.1) and (5.2), we define two matrices for the peaks in H_θ . We call these matrices: peak distances matrix Δ_θ and valley between peaks matrix V_θ , which are defined as:

$$\Delta_\theta = \{\delta_j | \delta_j = n_{j+1} - n_j\} \quad (5.3)$$

$$V_\theta = \{v_j | v_j = \min(h_{n_j} : h_{n_{j+1}})\} \quad (5.4)$$

where δ_j represents the distance between the positions of two peaks in P_θ , and v_j is the valley that is found between the two peaks. Using the above equations, P_θ is now

5. Complex Linear Road Markings

Figure 5.8: Illustration showing valid and invalid peaks. Both the peaks in all the three cases have the same peak height but their separations are different: (a) local maxima, which is a part of a major peak (b) two valid peaks separated by a distance (c) two valid peaks separated by a valley.

divided into n_P exclusive sets P_{θ_1} to $P_{\theta_{n_P}}$ such that

$$\mathcal{P} = \{P_{\theta_1}, P_{\theta_2}, \dots, P_{\theta_k}, \dots, P_{\theta_{n_P}}\} \quad (5.5)$$

$$\text{where} \quad (5.6)$$

$$P_{\theta} = P_{\theta_1} \cup P_{\theta_2} \dots \cup P_{\theta_k} \dots \cup P_{\theta_{n_P}}$$

Similarly we divide N_{θ} into n_P sets, where each N_{θ_k} holds the indices of the peaks in P_{θ_k} . Therefore,

$$\mathcal{N} = \{N_{\theta_1}, N_{\theta_2}, \dots, N_{\theta_k}, \dots, N_{\theta_{n_P}}\} \quad (5.7)$$

$$\text{where} \quad (5.8)$$

$$N_{\theta} = N_{\theta_1} \cup N_{\theta_2} \dots \cup N_{\theta_k} \dots \cup N_{\theta_{n_P}}$$

We will now show how these exclusive sets are formed. Each of these P_{θ_k} represents a possible set of peaks, which are *forming* into one valid peak. In other words, each P_{θ_k} has one valid peak surrounded by local maximas or peaks.

The bounds of each P_{θ_k} are found in an iterative manner. In every iteration, we find the maximum peak $\hat{p}$ in the set of peaks P_t and then find associated local maximas (or peaks) that form $\hat{p}$. This set of peaks is denoted by P_{θ_k} where k is the iteration index. The process is repeated with the remaining set of peaks till all peaks in P_{θ} have been exhausted.

5.3 Adaptive Hough Peak Detection (AHPD)

Figure 5.9: Illustration on the process of splitting P_θ .

We describe this decomposition process using the illustration in Fig. 5.9. We start with P_θ as the initial array of peaks for iteration $k = 1$. Hence $P_t = P_\theta$. The maximum peak in P_t is given by $\hat{p} = p_j$. It is to be noted that all indices of array elements are taken with respect to the indices in P_θ . The left and right bounds' indices of P_t are equivalent to that of P_θ , i.e. $l_b = 1$ and $r_b = q$. We now apply function $\Phi(\cdot)$ on P_t to find the left and right bounds' indices that bind $\hat{p}$ with its associated local peaks. The definition of $\Phi(\cdot)$, that is applied on P_t with maximum peak $\hat{p} = p_j$ in the current iteration, is given by:

$$\begin{aligned}
 & m_1 = l_b; m_2 = r_b \\
 & \textbf{for } u = l_b : j \\
 & \quad \textbf{if } (\delta_u > \delta_{min}) \wedge (p_u - v_u > \lambda \hat{p}) \\
 & \quad \quad \{m_1 = u; break\} \\
 & \textbf{endfor} \\
 & \textbf{for } u = j : r_b \\
 & \quad \textbf{if } (\delta_u > \delta_{min}) \wedge (p_u - v_u > \lambda \hat{p}) \\
 & \quad \quad \{m_2 = u; break\} \\
 & \textbf{endfor}
 \end{aligned} \tag{5.9}$$

In words, $\Phi(\cdot)$ gives left bound m_1 and right bound m_2 , by checking the conditions we set for associating local maximas to the maximum peak (as discussed earlier). If m_1 and m_2 are found to be lesser than the bounds of the P_t , i.e. l_b and r_b respectively, P_t is divided into sub arrays P_{t1} , P_{t2} and so on. The array with the bounds as m_1 and

5. Complex Linear Road Markings

m_2 , having the maxima $\hat{p}$ is denoted by $P_{\theta k}$ where k is the iteration index. Referring to Fig. 5.9, we now split P_t at $m_1 = q_1$ and $m_2 = q_2$ resulting in three matrices P_{t1} , $P_{\theta 1}$ and P_{t2} , $P_{\theta 1}$ being the array where $\hat{p}$ was found in P_t . Therefore, in Fig. 5.9, $P_{\theta 1} = P_{\theta}(q_1 : q_2)$. Using the same indices, we get $N_{\theta 1} = N_{\theta}(q_1 : q_2)$. Therefore, the sets $\mathcal{P}$ and $\mathcal{N}$ at the end of first iteration are given by:

$$\mathcal{P} = \{P_{\theta 1}\} \quad \mathcal{N} = \{N_{\theta 1}\} \quad (5.10)$$

In the next iteration, we consider P_{t1} and P_{t2} to get the next maximum peak $\hat{p}$. Therefore, $\hat{p} = \max\{P_{t1}, P_{t2}\}$. P_t is now set to the array where $\hat{p}$ is found. In Fig. 5.9, $P_t = P_{t2}$. $\Phi(\cdot)$ is applied on P_t resulting in $m_1 = q_3$ and $m_2 = q$. It is to be noted that the indices m_1 and m_2 are taken with respect to the original array P_{θ} . The resulting set of peaks from iteration $k = 2$ is $P_{\theta 2} = P_{\theta}(q_3 : q)$ and this process is repeated till all selected peak candidates (that satisfied M_{sep}) in P_{θ} are covered.

The final step of Stage 1 of the proposed automated peak detection is getting the possible peak candidates in the clusters in $\mathcal{P}$. In this step, we go through every cluster $P_{\theta k} \in \mathcal{P}$. We qualify a peak $p_j \in P_{\theta k}$ to the next stage if and only if p_j sits above the other peak candidates in $P_{\theta k}$. In other words p_j must be what we call as *peak of peaks*. In order to do this, we use similar equations as (5.1) to compute the peak of peaks array $\overline{P}_{\theta k}$ for the cluster $P_{\theta k}$ as shown below:

$$\overline{P}_{\theta k} = \{\bar{p}_j | (\bar{p}_j = p_j) \Leftrightarrow (p_j > p_{j-1}) \wedge (p_j < p_{j+1})\} \quad (5.11)$$

where $p_j, p_{j-1}, p_{j+1} \in P_{\theta k}$. In addition, for every element $\bar{p}_j \in \overline{P}_{\theta k}$, we also find the number of peaks on the left and right of $\bar{p}_j$ which form $\bar{p}_j$ as the peak of peaks. These counts on the left and right of $\bar{p}_j$ are given by c_{j-} and c_{j+} respectively. We define $\mathcal{C}$ which is a set of tuples $C_j = (c_{j-}, c_{j+})$ as given below:

$$\mathcal{C} = \{C_1, C_2, \dots, C_j, \dots, C_{n\theta k}\} \quad (5.12)$$

$$\mathcal{C} = \{(c_{1-}, c_{1+}), (c_{2-}, c_{2+}), \dots, (c_{j-}, c_{j+}), \dots, (c_{n\theta k-}, c_{n\theta k+})\} \quad (5.13)$$

Now the final filtering in stage one is performed by checking the count of peaks leading to the peak of peaks in $\overline{P}_{\theta k}$. For every $\bar{p}_j$ we check if its corresponding c_{j-} and c_{j+} satisfy a maximum number to qualify $\bar{p}_j$ as a valid peak candidate for Stage 2 of the

5.3 Adaptive Hough Peak Detection (AHPD)

Figure 5.10: (a) Hough space $\mathcal{H}$ computed from Hough transform, (b) Reduced Hough space $\mathcal{H}'$ obtained after applying Stage 1 of AHPD.

algorithm. This is to ensure that noise is not considered as a valid peak. As discussed earlier, if too many peaks are forming a peak, then this could be due to the presence of spurious peaks created by noise. In order to eliminate these spurious peaks, we apply this final condition on the peaks of peaks in $\bar{P}_{\theta k}$. This final condition was found to be optional during the testing phase of the algorithm. The eliminations that are done till this step eliminate most non-valid or noisy peaks. However, this step can also be added for further tightening the constraints on the peak selection process, if needed.

The peaks which qualify all the above conditions are used to create a new array H'_θ with only the final selected peaks and all other elements set to zero. Compiling all H'_θ s, we get a new *reduced* Hough accumulation space, denoted by $\mathcal{H}'$, which we send for Stage 2 processing. Fig. 5.10 shows the Hough space $\mathcal{H}$ generated after Hough Transform computation and the reduced Hough space $\mathcal{H}'$ that is generated after the first stage of the proposed AHPD. It can be seen that the entire Hough space is cleared and peaks surrounding valid peaks only are remaining in $\mathcal{H}'$.

5.3.1.2 Stage 2: Confirming Hough Peaks Using Peak Patterns

In Stage 1, we looked at the patterns in peaks that are formed along the ρ values for every θ in Hough accumulation space. After getting possible peak candidates from Stage 1, we now look at the patterns for every peak across θ values in Stage 2. It was shown in [156], that the peaks in Hough space form what are termed as “butterflies”. Such butterfly patterns can be seen around the peaks in Fig. 5.10(a). Zoomed in versions of the same are shown in Fig. 5.11. The butterflies formed in $\mathcal{H}$ and $\mathcal{H}'$ are shown in

5. Complex Linear Road Markings

Figure 5.11: Zoomed in versions of the Hough spaces in Fig. 5.10 showing the butterfly in $\mathcal{H}$ and butterfly-like pattern in $\mathcal{H}'$.

Fig. 5.11(a) and (b) respectively. A butterfly is pattern formed by the accumulation counts such that the accumulation counts taper down from the main valid peak as we traverse away from the valid peak position. [156] describe such patterns as butterflies and template matching techniques are used to detect such butterflies.

It can be seen from the reduced Hough space in Fig. 5.11(b) that the butterflies have been cleaned up by Stage 1 of AHPD. We now describe Stage 2 of AHPD which identifies these butterfly-like patterns which are seen in the reduced Hough space $\mathcal{H}'$. The objective is to find the one peak at (ρ, θ) which other peaks in its surrounding are forming in $\mathcal{H}'$.

In order to do this, we evaluate the neighborhood of every peak in the reduced Hough space $\mathcal{H}'$ that we obtained from Stage 1. In every iteration, we first find the maximum p_{max} in $\mathcal{H}'$ and find the set of peaks S across θ s that form p_{max} . These peaks are then removed from $\mathcal{H}'$ and the process is repeated till all peaks are exhausted in $\mathcal{H}'$. We will now describe the algorithm to identify this set of peaks around the maximum peak that form the butterfly-like pattern in the reduced Hough space $\mathcal{H}'$. Given $\mathcal{H}'$, we find the maximum peak value p_{max} , i.e.

$$p_{max} = \max\{\mathcal{H}'\} = \mathcal{H}'(\hat{i}, \hat{j}) \quad (5.14)$$

where $\hat{i}$ and $\hat{j}$ are the indices along ρ and θ axes respectively in $\mathcal{H}'$. Given the location

5.3 Adaptive Hough Peak Detection (AHPD)

of p_{max} , we get a $w \times w$ neighborhood K around p_{max} , i.e.

$$K = \begin{bmatrix} k_{1,1} & k_{1,2} & \cdots & k_{1,\frac{w+1}{2}} & \cdots & k_{1,w} \\ k_{2,1} & k_{2,2} & \cdots & k_{2,\frac{w+1}{2}} & \cdots & k_{2,w} \\ \vdots & & & \vdots & & \vdots \\ k_{\frac{w+1}{2},1} & k_{\frac{w+1}{2},2} & \cdots & k_{\frac{w+1}{2},\frac{w+1}{2}} & \cdots & k_{\frac{w+1}{2},w} \\ \vdots & & & \vdots & & \vdots \\ k_{w,1} & k_{w,2} & \cdots & k_{w,\frac{w+1}{2}} & \cdots & k_{w,w} \end{bmatrix} \quad (5.15)$$

where $k_{i,j} = h'_{i',j'}$. The indices (i,j) and (i',j') are related in the following way:

$$i' = \hat{i} - \left(\frac{w+1}{2} - i \right) \quad \text{and} \quad j' = \hat{j} - \left(\frac{w+1}{2} - j \right) \quad (5.16)$$

K is divided into four quadrants Q_1, Q_2, Q_3 and Q_4 given by the following four matrices:

$$Q_1 = \{k_{i,j}\} \text{ where } 1 \leq i \leq \frac{w+1}{2}; \frac{w+1}{2} \leq j \leq w \quad (5.17)$$

$$Q_2 = \{k_{i,j}\} \text{ where } 1 \leq i, j \leq \frac{w+1}{2} \quad (5.18)$$

$$Q_3 = \{k_{i,j}\} \text{ where } \frac{w+1}{2} \leq i \leq w; 1 \leq j \leq \frac{w+1}{2} \quad (5.19)$$

$$Q_4 = \{k_{i,j}\} \text{ where } \frac{w+1}{2} \leq i, j \leq w \quad (5.20)$$

It should be noted that most of the elements in K will be zeros and only the peaks that were qualified in Stage 1 will be non-zero elements. Referring to the illustration in Fig. 5.12, the quadrants defined above are shown in the $w \times w$ neighborhood around $p_{max} = 45$. For the purpose of illustration, we consider $w = 7$. Each of the squares represents a position in the reduced Hough accumulation space. The empty squares represent an accumulation value of 0. We use this illustration in Fig. 5.12 to describe the proposed Stage 2 of AHPD.

Now, we find sets S_u , where $u = 1, 2, 3, 4$ corresponding to each of the four quadrants Q_1 to Q_4 . Each S_u is a set of non-zero peak values in its respective quadrant, i.e.

$$S_u = \{s_t | s_t = k_{i,j} \text{ if } k_{i,j} > 0 \wedge k_{i,j} \in Q_u\} \quad (5.21)$$

In Fig. 5.12, $S_2 = \{40, 37\}$ and $S_4 = \{40, 39\}$; S_1 and S_3 are empty. For each element $s_t \in S_u$, we check if s_t is lesser than peak at the center $k_{\frac{w+1}{2}, \frac{w+1}{2}}$ to create what we call as the sign set S'_u defined as:

$$S'_u = \{s'_t | s'_t = 1 \text{ if } s_t < k_{\frac{w+1}{2}, \frac{w+1}{2}} \text{ else } s'_t = 0\} \quad (5.22)$$

5. Complex Linear Road Markings

Figure 5.12: Stage 2.

In Fig. 5.12, $S'_2 = \{1, 1\}$ and $S'_4 = \{1, 1\}$. Next for each s_t in S_u , we find its Manhattan distance from the center, i.e. for $s_t = k_{i,j}$, we find

$$d_t = \left| i - \frac{w+1}{2} \right| + \left| j - \frac{w+1}{2} \right| \quad (5.23)$$

If S_u has a non-zero peak, we jump to the farthest element s_f in Q_u . Referring to Fig. 5.12, we jump to 37 in Q_2 and 39 in Q_4 .

Now, keeping s_f as the new center, we check for non-zero peaks in the same direction as s_f w.r.t its center in the previous iteration. In other words, if s_j was found in Q_u , we check Q_u quadrant w.r.t s_f in the current iteration. For example, in Fig. 5.12, after we jump to 37, we keep 37 as the center and check quadrant Q_2 w.r.t 37. Similarly for 39, we check Q_4 w.r.t 39. These quadrants w.r.t 37 and 39 in Q_2 and Q_4 respectively are marked in Fig. 5.12. This movement along the direction of the non-zero element that is less than the center ensures that we are moving in the direction of the butterfly, i.e. the direction of the ridge that is forming the main peak $k_{\frac{w+1}{2}, \frac{w+1}{2}}$.

5.3 Adaptive Hough Peak Detection (AHPD)

In these quadrants, we now find non-zero peaks and append them to their corresponding S_u sets. We also update the corresponding sign set S'_u with respect to the new centers, i.e. $k_{\frac{w+1}{2}, \frac{w+1}{2}} = s_f$. In Fig. 5.12, the non-zero peaks in Q_2 w.r.t 37 are appended to S_2 ; therefore $S_2 = \{40, 37, 35\}$. We check if 35 is lesser than its center, i.e. 37 and update $S'_2 = \{1, 1, 1\}$. We repeat this process of jumping to the farthest peak and checking its respective quadrant, and updating the respective S_u and S'_u sets, until either of the following conditions are met:

1. Either there are no more peaks in the quadrant that is being checked.
2. Or S'_u sees a string of n_{0min} number of zeros.

The first condition indicates that there are no more peak candidates and the second shows that there could be non-decreasing peaks (which are forming another valid peak in the vicinity). Referring to Fig. 5.12, in the 4th iteration, when the quadrant Q_2 is centered on 25, we see S_2 will not have any more non-zero peaks. So this is a termination condition for the movement along Q_2 . In S_4 , we will see a string of increasing peaks after the 4th iteration resulting in a string of 0s resulting in $S'_4 = \{1, 1, 1, 1, 1, 0, 0\}$. This indicates a termination condition for movement along Q_4 .

When all the quadrants around p_{max} terminate. We check if two diagonally opposite quadrants have a minimum string of non-zero peaks that follow a decreasing trend as they move away from p_{max} . This is checked by looking at the sign sets S'_u . In Fig. 5.12, we see that S_2 is diagonally opposite to S_4 and both of them have 4 non-zero peaks that follow a decreasing trend. If the two diagonally opposite quadrants satisfy a minimum number, we consider p_{max} as a valid peak. We consider all the peaks in all four quadrants that contributed to p_{max} as part of p_{max} and remove them from $\mathcal{H}'$.

The whole process is repeated again by finding the next highest peak in $\mathcal{H}'$. This process is repeated till all the peaks in $\mathcal{H}'$ have either been associated with a valid peak or been rejected as a spurious peak. In this manner, only the valid peaks are retained and the peaks that are contributing to the valid peaks are being rejected. We will evaluate the AHPD using benchmark images to study its effectiveness on images with linear edges.

5.4 Evaluation of AHPD

5.4.1 Accuracy Evaluation

Figure 5.13: Sample images chosen (a) synthetic image with closeby edges (b) airplane with lines of varying length (c) bed-lamp with thin features having closeby edges.

Figure 5.14: Synthetic test image containing closeby lines (a) image (b) Hough space.

The proposed AHPD is compared against the `houghpeaks()` function in one of the latest versions of MATLAB version 7.12 (R2011a). The `houghpeaks()` function consists of 3 parameters that can be manually input by the user namely, ‘Threshold’, ‘NHoodSize’ and ‘NUMPEAKS’. The ‘Threshold’ refers to the value above which the entries in the Hough space are even considered for peak detection. Omitting this setting would set this to the default value corresponding to $0.5 * (\max(H))$, where $\max(H)$ refers to the highest peak in the Hough space. ‘NHoodSize’ specifies the size of the suppression neighborhood. This is the neighborhood around each peak that is set to zero after the peak is identified, with an objective to eliminate redundant entries for a given valid peak. The default value is set to the smallest odd values greater than or equal

to $\text{size}(H)/50$, where $\text{size}(H)$ refers to the size of the accumulation array. NUMPEAKS specifies the maximum number of peaks to identify. If NUMPEAKS is omitted, it defaults to 1. For the proposed AHPD, a threshold value is preset, corresponding to the minimum length of lines of interest, corresponding to the minimum height of the entries in the Hough space for them to qualify for the detection process.

Figure 5.15: Hough peak detection results on airplane image with `houghpeaks()` and AHPD (a) `houghpeaks(NHoodSize = default, NUMPEAKS = 10, Threshold = 0.35*max(H))`; (b) `houghpeaks(NHoodSize = 5x5, NUMPEAKS = 10, Threshold = 0.35*max(H))`; (c) `houghpeaks(NHoodSize = 5x5, NUMPEAKS = 30, Threshold = 0.35*max(H))`; (d) `houghpeaks(NHoodSize = default, NUMPEAKS = default, Threshold = 0.35*max(H))`; (e) AHPD with `Threshold = 0.35*max(H)`.

To demonstrate the effectiveness of the proposed adaptive Hough peak detection (AHPD), we use a synthetic image containing closely lines (Fig. 5.13 (a)) and two sample images taken from the Caltech 101 database [161]. The first is that of a airplane 5.13 (b) that contains lines of varying lengths within the image, and the other is that of a bed-lamp containing edges very close to each other 5.13 (c).

5. Complex Linear Road Markings

The Hough space of Fig. 5.14 (a) is shown in Fig. 5.14 (b). There are two peaks at $(\rho, \theta) = (164, 136)$ and $(\rho, \theta) = (161, 136)$. When AHPD is applied to it, both these peaks are picked up, and the reduced intermediate Hough space is shown in Fig. 5.14 (c). However, when the `houghpeaks()` function of MATLAB is applied with the default settings, only one peak at $(\rho, \theta) = (164, 136)$ is picked up. While it can be argued that setting a much smaller suppression neighborhood such as 3×3 will ensure both lines are picked up (as the separation of peaks is 3 pixels), it is important to note that this would trigger a new set of problems with other kinds of images. Having a small suppression neighborhood implies redundant entries are picked up for the same line, and this has serious repercussions for scenarios with varying line lengths, because important lines of medium length are missed out in comparison with redundant entries of the long lines.

In other words, the universality of these settings is not achievable with the standard functions like `houghpeaks()`, thus rendering the approach as non-adaptive. And sometimes, even various combinations of the settings do not yield the desired result. This is illustrated in Fig. 5.15 and Fig. 5.16. In Fig.5.15 (a) and Fig.5.15 (b), it can be seen that multiple lines are picked up for the longer edges of the airplane, while other important edges of medium length are missed out. Increasing the number of peaks from 10 to 30 (Fig.5.15 (c)) does not help to pick up these missed lines, but only adds to more redundancy in the longer lines. Setting the number of peaks to default value (Fig.5.15 (d)) leads to only one long line being picked up. Thus, it can be seen that many combinations of settings in the `houghpeaks()` do not yield the desired result, as lines of interest are either missed out, and/or the redundancy in detection is too high.

The result of peak detection using the proposed AHPD is shown in Fig.5.15 (e), with the same threshold setting as the `houghpeaks()` set to $0.35 * (\max(H))$. It can be seen that all the major lines that are above the threshold are picked up, with almost a zero redundancy. This is because the proposed AHPD uses shape information of the peaks and does not depend solely on the local heights of the entries in the Hough space.

In another example shown in Fig. 5.16, the black spokes forming the shade of the lamp have very closeby edges. It is seen that when the `houghpeaks()` function is used with number of peaks set to 10, it misses out on some of the closeby lines. Increasing the number of peaks to 30 does not help to detect the missed lines, but only increases the redundancy Fig. 5.16 (b). The AHPD successfully detects close-by lines adaptively.

Figure 5.16: Hough peak detection results on bed-lamp image using (a) hough-peaks(NHoodSize = 5x5, NUMPEAKS = 10, Threshold = 0.35*max(H)), (b) hough-peaks(NHoodSize = 5x5, NUMPEAKS = 30, Threshold = 0.35*max(H)), (c) AHPD.

5.4.2 Computation Cost Analysis

The computation cost of the proposed AHPD is discussed in this sub-section. It was shown in the previous section that the accuracy of AHPD is better than the conventional non-maximal suppression based Hough peak detection method. Also, AHPD is an automated method, in terms of the number of peaks detected in an image, as compared to the conventional method, which needs the number of peaks as a manual input. Therefore, it would be unfair to compare the computational complexity of these methods against each other. The butterfly based peak detection scheme in [156] was chosen to compare the computation cost against the proposed AHPD because it is a semi-automated Hough peak detection method. This method is based on template matching of butterfly-like templates over the entire Hough space to detect the peaks. The computation complexity of this method is directly based on the number of the templates that are being scanned using normalized cross correlation (NCC) method. Consider a $k \times k$ sized template. The complexity of the NCC using such a template is dominated by the number of multiplications in the template matching process. The total number of operations in one NCC operation is given by:

$$\begin{aligned} C_{NCC}^{MUL} &= 3k^2 + 3 \\ C_{NCC}^{ADD} &= 3k^2 \end{aligned} \tag{5.24}$$

5. Complex Linear Road Markings

where C^{MUL} and C^{ADD} are the number of multiplications and additions, respectively. The computation complexity of square-root and division operations are considered as the same as multiplication operation for simplicity (giving the benefit to this method).

For the sake of comparison, the detection of one peak in the Hough space using both the methods are considered. Consider a $u \times v$ neighborhood around the peak to determine computational cost of the peak detection process. The computation cost of the method in [156] to find the peak in this $u \times v$ neighborhood, using $n_{templates}$ number of templates for NCC matching, is given by:

$$C_{[156]} = ((3k^2 + 3)w + 3k^2)uv \times n_{templates} \text{ additions} \quad (5.25)$$

where it is considered that one multiplication is composed of w additions, w being the data-width. In AHPD, considering the worst case scenario, for a $u \times v$ window, we would need $4uv$ number of comparisons in the first stage to select possible peaks for the next stage. It is now assumed that 4 peaks are selected by the first stage processing in each column of the $u \times v$ window (this is an exaggeration and a worst case scenario). For each peak, 16 comparisons are required, i.e. 4 in each direction in stage 2 of AHPD. Though it will not be needed in all directions, but it is being considered as the worst possible scenario. Therefore, the total number of comparisons in AHPD that are needed to find one peak in a $u \times v$ neighborhood is given by:

$$C_{AHPD} = 68uv \text{ comparisons} \quad (5.26)$$

Assuming that the cost of each comparison is equivalent to a multiplication, the cost of AHPD C_{AHPD} is equal to $68uvw$ additions, where w is the datawidth.

Table 5.1 shows the comparison of computation cost (in terms of additions) between AHPD and butterfly template algorithm in [156]. The comparisons are made for a 50×50 window and different template sizes and number of templates for [156]. It can be seen from Table 5.1 that the scenario with number of templates being set to 1, which is the best possible condition for [156], is computationally more complex than the proposed AHPD method. As the number of templates increase or size of the templates increase, the cost savings also increases. The area overhead of AHPD is dependent on the implementation of the AHPD but the results in Table 5.1 shows that the overall computation cost of AHPD is lesser than existing similar methods.

Table 5.1: Computation cost comparison of AHPD with [156]

	# templates	Size of template	Cost in additions	Savings
Proposed	-	-	2720000	-
[156]	1	7	6367500	0.58
	1	5	3307500	0.18
	2	7	12735000	0.79
	2	5	6615000	0.59

5.5 CLM Detection Using AHPD

In this section, the adaptive Hough peak detection algorithm is deployed to detect and classify complex linear markings. First, detection of double linear markings is described, following which, an algorithm to detect zigzag markings is developed using AHPD.

5.5.1 Detecting Double Linear Markings Using AHPD

As illustrated in Fig. 5.3, the use of conventional Hough peak detection methods poses challenges in detecting the closely lines that make up the double lane marking. Replacing the conventional peak detection with the proposed AHPD resolves this issue and the closely lines of interest are detected successfully. Consider the same image previously discussed in Fig. 5.3. The lines that were missed out earlier are now successfully detected by using AHPD are shown in Fig. 5.17. Fig. 5.17 (a), (b) and (c) once again show the image, edge map and the line detection output using conventional peak detection, respectively. Fig. 5.17 (d) shows the output when AHPD is applied.

Figure 5.17: Illustration to show AHPD successfully detecting closely lines in a double lane marking.

In order to check whether the marking type is single or double, it is essential to first detect all the BLMs that are present in the region of interest. The BLM detection

5. Complex Linear Road Markings

module is modified to detect double linear markings by replacing the conventional Hough peak detection with the AHPD proposed here. Then, using the output of the modified BLM detect module, which can now detect double markings, the lane markings are classified into the following four categories - single solid, double solid, single dashed and double dashed. The output of the BLM detect algorithm is fed into a simple step to differentiate single versus double markings, based on the number of parallel BLMs that are detected, that satisfy the lane marking properties.

Recapping, at the end of the steps described for detecting BLM in Sec. 3.9, it was shown that if all the criteria for lane markings are met for a line pair $l - r$ in the block being processed, then their positions and angles (ρ_l, θ_l) and (ρ_r, θ_r) are output, where l and r refer to the left and right counterparts of the line pair. If multiple such pairs are present in the block, then all the line pairs are identified and detected at this stage. An additional step to check the number of valid line pairs will tell whether the type of marking is a single or double marking. If the number of such valid line pairs is only one, then it detects a single lane marking. On the other hand, if two such line pairs are detected, then a double lane marking is detected.

Once the line pairs are given by the modified BLM detection unit, the spatial and temporal continuity check explained in Sec. 3.10.1 and 3.10.2 of Chapter 3 respectively, are performed. In cases where multiple valid line pairs are output by the BLM detection unit, it is sufficient to perform these continuity checks on any one of them. The broad steps for classifying the lane markings as is summarized in Fig. 5.18. Similarly, for the horizontal double and dashed markings, the spatial continuity in the horizontal direction is checked, as proposed in Sec. 3.10.1 of Chapter 3, with the only difference being the use of AHPD to detect closeby peaks, thereby aiding the detection of double markings.

5.5.2 Detecting Zigzag Markings Using AHPD

A zigzag marking has multiple BLMs placed in a zigzag orientation. When seen from an aerial view (Fig. 5.19 (a)), a zigzag marking would reveal two sets of BLMs (sets A and B as illustrated in Fig. 5.19 (b),(c)), oriented at angles θ_A and θ_B respectively, meeting each other in an alternating fashion. The zigzag is subjected to perspective skewing when seen from the host camera, as shown in Fig. 5.19 (d). It can be observed that since the zigzag markings are drawn along the lane direction, the overall marking

Figure 5.18: Detecting double lane markings and classification into (1) single solid (2) single dashed (3) double solid and (4) double dashed.

runs towards the vanishing point. However, the individual strokes of lines that make the zigzag marking do not converge into the vanishing point.

This property of multiple BLM in the region of lane markings, that are oriented in directions that do not converge towards the vanishing point, are employed in the proposed zigzag marking technique.

Figure 5.19: Aerial view of zigzag marking.

One important observation regarding zigzag road markings is that they occur only

5. Complex Linear Road Markings

at road boundaries or carriageway boundaries, not in between lanes of the same direction. Also, they can occur only along the lane, never across. Considering that the default marking at road boundaries or carriageway boundaries is a solid lane marking (be it single or double), the most common scenario for a zigzag marking is when a solid marking precedes a zigzag, as shown in Fig. 5.20 (a). However, sometimes, due to a cross roads merging into the host lane, dashed markings (single or double) are seen replacing solid markings at those stretches, followed by zigzags. This scenario is illustrated in Fig. 5.20 (b). It is also possible that the zigzag marking starts from a region of no lane markings, as shown in Fig. 5.20 (c). Finally, in these different scenarios, the zigzag marking can be either single or double. An example of a double zigzag marking is shown in in Fig. 5.20 (d).

Figure 5.20: Possible zigzag marking scenarios (a) Most common with solid marking preceding zigzag (b) Case with dashed marking at cross road preceding zigzag.

Deriving trigger conditions for zigzag marking detection: One of the indicators for a zigzag marking is when the lane marking that was detected in history is not found in the current frame. When the recent history shows the presence of a dashed or solid lane marking, single or double, but the current frame does not detect the same marking in the expected direction, then it could be due to one of the three

cases below:

1. The most common case when the near view of the current frame yields a negative, although history information reveals a valid marking, is when host vehicle is in a zone of dashed linear markings. As described earlier, dashed markings show an alternating series of positive and negative detection, temporally. The presence of dashed markings can be verified by doing a continuity analysis in the spatial and temporal domains, as described in Sec. 3.10.1 and 3.10.2 of Chapter 3.
2. If this is not the case, then the other possibility is that the host vehicle has entered a zone of zigzag markings. In this case, the angles of the component strokes forming the zigzag would be different from the angle of the expected lane markings (as per history), and hence the detection shows a negative for the marking in the current frame.
3. If it is neither the case of zigzags, nor the case of dashed lines, then it is possible that the host vehicle has entered a zone of no lane markings.

Based on these observations, we propose the zigzag detection algorithm as shown in Algorithm 3. This algorithm summarises the steps of zigzag detection, including the trigger condition. Consider k to be the index of the current image frame; inputs to the algorithm are the current frame, I_k , and detection history of past N frames. $R(k)$ is the binary variable introduced in Sec. 3.10.2 of Chapter 3 to store the detection outcome of the k^{th} frame. When this is a 0 and when the history shows a dashed marking, then spatial continuity check is done to confirm the presence of dashed markings. If not, the frame is checked for presence of zigzag marking.

The region of interest for finding the zigzag marking is derived based on the host lane boundaries, considering that zigzag markings are drawn in the region of lane boundaries, along the road. From history, the expected lane boundaries on the left $((\rho_k^l, \theta_k^l))$, and right $((\rho_k^r, \theta_k^r))$ of the host lane are inferred from history by an averaging operation of the (ρ, θ) values of the past N frames. Next, a mask is created around the expected lane boundary position, by considering a parallelogram bound by lines $l_1(\rho_k + \Delta\rho_k, \theta_k)$ and $l_2(\rho_k - \Delta\rho_k, \theta_k)$. This is illustrated in Fig. 5.21, with the RoI represented by the shaded parallelogram.

5. Complex Linear Road Markings

Algorithm 3 Algorithm for Zigzag marking detection

```

1: Input: current frame  $I_k$ , history of past  $N$  frames
2: if  $R_k == 0$  & no dashed linear marking found then
3:   start zigzag detection
4:    $\rho_k^l = 1/N \left[ \sum_{i=1}^N \rho_{k-i}^l \right]$ ,  $\theta_k^l = 1/N \left[ \sum_{i=1}^N \theta_{k-i}^l \right]$ 
5:    $\rho_k^r = 1/N \left[ \sum_{i=1}^N \rho_{k-i}^r \right]$ ,  $\theta_k^r = 1/N \left[ \sum_{i=1}^N \theta_{k-i}^r \right]$ 
6:   
$$\begin{bmatrix} \cos\theta_k^l & \sin\theta_k^l \\ \cos\theta_k^r & \sin\theta_k^r \end{bmatrix}^{-1} \begin{bmatrix} vp_x \\ vp_y \end{bmatrix} = \begin{bmatrix} \rho_k^l \\ \rho_k^r \end{bmatrix}$$

7:    $M$ : create mask bound by lines  $l_1(\rho_k + \Delta\rho_k, \theta_k)$  and  $l_2(\rho_k - \Delta\rho_k, \theta_k)$ 
8:    $I_M := I_k \otimes M$ 
9:    $\{l_1(\rho_1, \theta_1), \dots, l_P(\rho_P, \theta_P)\} = blmDetect(I_M, \theta_k \pm \Delta\theta)$ 
10:  if  $(P \geq 2)$  then
11:    for  $i := 1$  to  $P$  do
12:       $vpcheck = ptLine(\rho_i, \theta_i, vp_x, vp_y)$ 
13:      if  $vpcheck == 0$  then
14:         $z = z+1$ 
15:      end if
16:    end for
17:    if  $z \geq 1$  then
18:      Output:  $zzDetect := 1$ 
19:    else
20:      Output:  $zzDetect := 0$ 
21:    end if
22:  end if
23: end if

```

Once the RoI is set, BLM detection algorithm proposed in Chapter 3 is applied on the image within the RoI, for a wider angle range. In terms of the function `blmdetect()`, the input parameter $\delta\theta$ is set to a large value to accommodate linear segments of the zigzag, that have angles of orientation that are more spread as compared to straight lane markings. The output of the `blmdetect()` function are subject to the condition that the lines do not align towards the vanishing point. As per the vanishing point criteria described above, although the overall zigzag marking is drawn in the general direction towards the vanishing point, the individual BLMs that constitute the zigzag are not aligned towards the vanishing point. If atleast two such BLMs are detected, then the presence of the zigzag marking is confirmed.

Figure 5.21: Region of interest for zigzag detection.

5.6 Results and Discussion

It was shown how the BLM detection was extended to detect double linear markings using the proposed AHPD. The algorithm was extended to classify lane markings as one of the following four types, viz. single-dashed, single-solid, double-dashed and double-solid. This algorithm was applied on the dataset that was described previously in Sec. 3.6. 40 image sequences representing all the four types of lane markings were chosen. It was seen that the algorithm leads to 98 % accuracy in detecting the type of lane marking. The misdetections were primarily in cases with road boundaries and pavement edges contributing to multiple edges that sometimes get detected as double linear markings.

One of the same sequences is shown in Fig. 5.22, which shows an image sequence with a double solid lane marking on the left and a single-dashed lane marking on the right. The classification output has been embedded as inset images on the top-left and top-right corners of the images in the sequence, for validation purposes. Similar tests were done on the rest of sequences and it was found that the modified BLM detect module with AHPD is able to detect multiple lane markings when double lane markings are present along the host lane.

The proposed zigzag marking detection technique was evaluated on the dataset that was used earlier in Chapter 3. Specially, 35 image sequences containing zigzag markings and 35 of them with straight lane markings on either side were chosen as the sample set. Some sequences had zigzags on both sides, while some had it on one

5. Complex Linear Road Markings

Figure 5.22: Classification results of single/double and solid/dashed.

side. It was shown that the method yields a 100 % accuracy where all sequences with zigzag markings were detected successfully. The negative samples without the zigzag marking were shown to lead to negatives in the detection, confirming its performance in negative samples. Table 5.2 shows the results of the proposed technique on 8 different instances of roads where zigzag markings are found. Sample images corresponding to the 8 different road scenes are shown in Fig. 5.23. It can be seen from Fig. 5.23 that cases 1, 2, 6 & 7 have zigzag markings on both sides of the road. However, in cases 4 & 5, the zigzag marking is seen only on the right side, whereas the left side has a single continuous lane marking. Case 3 does not have any lane marking on the left but has zigzag marking on the right side. In case 8, we do not have any zigzag markings at all; both left and right sides have single dashed lane markings.

It was seen that the proposed algorithm is able to detect the presence of zigzag and if there is no zigzag marking, it determines what kind of other road marking is present in the image. These results are shown in Table 5.2. For every image in Fig. 5.23, the vanishing point, the (ρ, θ) pairs for left and right edges of the left and right host lanes that are detected by the proposed algorithm are shown in Table 5.2. Furthermore,

Figure 5.23: Images on which Algo. 3 are applied and results summarized in Table 5.2.

5. Complex Linear Road Markings

the ρ value that is obtained by using the vanishing point and the θ values of each line segment is also shown in Table 5.2 (this value is computed only for the left edge of the left and right lanes in the table; the same can be computed for the right edge of each lane also). This value is shown under ρ_{vpt}^L and ρ_{vpt}^R for the left and right lanes respectively in Table 5.2. These values are used to find the flag that is set to *yes*(Y) if the lane marking passes through vanishing point and vice versa. The flags are denoted as f_{Vpt}^L and f_{Vpt}^R for left and right lanes respectively in Table 5.2. It can be seen from Table 5.2 that in the cases where there is a zigzag marking, multiple linear edges are detected by the proposed technique and most of the linear edges do not pass through the vanishing point. In the case of non-zigzag marking cases, only one linear edge is detected which passes through the vanishing point. This is consistently shown in all the test cases that were considered.

5.7 Scale & Distortion Invariance for Linear Marking Detection

The movement of vehicles are typically modelled [147, 148, 149] by using two or more of following 6 degrees of motion: (a) Translation - up and down (heaving), (b) Translation - left and right (swaying), (c) Translation - forward and backward (surging), (d) Rotation - forward and backward (pitching), (e) Rotation - left and right (yawing) and (f) Rotation - side to side (rolling). Of these, it is common to assume that a vehicle on-road has limited heaving, swaying, pitching or rolling [148], based on the assumption that wheels are in contact with the ground at most times. Hence, the main degrees of freedom are limited to surging and yawing, as shown in Fig. 5.24.

When the vehicle moves on the lane, exhibiting the two movements above, the appearance of the road markings as captured by the vehicle camera, also changes accordingly. Specifically, a yawing of the vehicle changes the **viewpoint**, which inturn shifts the vanishing point laterally along the vanishing line. This leads to a distortion of the markings, which are now aligned towards the new vanishing point. This is illustrated in 5.25 (a). A forward surging along the lane causes an increase in the **size** of the marking, while a backward movement causes a reduction in the size. The change in size does not follow a simple scaling, but mixed with perspective distortions. Fig. 5.25 (b) illustrates the perspective distortion and scale variations of markings

5.7 Scale & Distortion Invariance for Linear Marking Detection

Table 5.2: Results of Zigzag Detection

#	V_{pt}	Left Lane				Right Lane			
		Left Edge	Right Edge	ρ_{vpt}^R	$f_{V_{pt}}^L = Y$	Left Edge	Right Edge	ρ_{vpt}^R	$f_{V_{pt}}^R = Y$
1	(612,546)	(90,-38)	(140,-34)	51	N	(765,33)	(771,34)	794	N
2	(635,562)	(694,-90)	(-654,-85)	-612	N	(758,22)	(750,18)	736	N
		(339,-20)	(232,-27)	296	N	(771,51)	(791,43)	818	N
		(894,48)	(896,48)	820	N	(739,19)	(751,23)	720	N
		(318,-23)	(330,-22)	264	N	(844,37)	(850,36)	833	N
3	(552,581)	(-348,-62)	(-369,-63)	-302	N	(763,63)	(748,66)	819	N
		(280,-24)	(305,-22)	248	N	(789,21)	(789,21)	757	N
		(825,53)	(722,74)	844	N				
		(311,-27)	(394,-21)	262	N				
4	(681,546)	(-183,-56)	(-181,-56)	-145	N				
		(233,-30)	(281,-27)	214	N				
		(-65,-41)	(-21,-38)	-43	N	(749,19)	(756,21)	740	Y
		(282,-22)	(303,-21)	243	N				
5	(624,559)	(-486,-68)	(-529,-71)	-440	N				
		(280,-21)	(239,-24)	254	N				
		(-132,-47)	(5,-39)	-87	N	(731,18)	(731,17)	724	Y
		(244,-27)	(248,-26)	213	N				
6	(515,536)	(-632,-83)	(-554,-77)	-563	N				
		(289,-23)	(203,-28)	267	N				
		(170,-36)	(188,-35)	123	N	(730,22)	(734,22)	692	N
		(-505,-81)	(-352,-69)	-431	N	(719,348)	(718,38)	734	N
7	(521,505)	(272,-27)	(258,-28)	235	N	(712,22)	(711,21)	693	N
		(820,45)	(740,69)	743	N	(724,42)	(710,49)	743	N
		(-89,-47)	(-32,-43)	-43	N	(669,49)	(657,53)	723	N
		(291,-23)	(290,-23)	254	N	(705,23)	(712,26)	670	N
8	(545,574)	(-340,-67)	(-384,-70)	-287	N	(497,83)	(-329,-79)	573	N
		(302,-21)	(310,-20)	275	N	(678,20)	(693,23)	653	N
8	(545,574)	(31,-44)	(53,-42)	26	Y	(772,31)	(776,31)	772	Y

at various distances from the camera position. The degree of perspective distortion increases with the distance of marking from the camera, and negligible when the distance tends to zero. Therefore, in the near-view, the markings undergo mainly a scale related transformation and less of distortion due to perspectives. However, in the far view, the perspective related distortions are more evident, along with scale changes. Any algorithm that is designed to detect road markings should be such that it caters to the variations in their appearances, both in terms of scale, and distortions that are caused due to perspective and viewpoint changes. In other words, the algorithm should be scale-invariant and distortion-invariant.

In the case of solid lane markings, including single and double solid lane markings,

5. Complex Linear Road Markings

Figure 5.24: Main degrees of freedom of vehicle (a) surging - forward/backward movement and (b) yawing - turning left/right.

the overall appearance of the marking remains unchanged with regards to scale. This is illustrated by the two sequences in Fig. 5.26, captured with the test vehicle being subject to a simple forward surge. The images in the top and bottom sequences contain double and single solid markings respectively, on the right side of the host lanes. The images are ordered from left to right with increasing timestamps, as the vehicle moves forward. It can be seen that the overall appearance of the solid markings does not increase in scale. This can be explained by the fact that the stretch of solid markings in the near view is continuously replaced by the stretch of solid markings in the far view; and that in the far view is constantly replaced by new stretches that enter the camera view. So, when the vehicle is passing through solid lane markings, the continuous presence of the solid marking throughout the stretch on the sides, from near to far view, at all times, makes the marking invariant to scale. While it is true that a particular segment of the marking increases in scale as it moves from far to near view, the overall appearance of the marking remains constant w.r.t. scale. In other words,

5.7 Scale & Distortion Invariance for Linear Marking Detection

Figure 5.25: Effect of scale and perspective on road markings.

scale-invariance is not an issue in the detection of solid linear markings.

Figure 5.26: Effect of scale on linear marking detection.

In the case of dashed linear markings, many existing methods treat each constituent ‘dash’ as a separate object, then these dashes will appear to increase in size, as the vehicle is subject to a forward surge, as illustrated in the dashed marking in in Fig. 5.26. Lane marking detection techniques that are based on template matching treat each dash as an object to be identified using the templates, and hence need to deal

5. Complex Linear Road Markings

with scale-invariance. The same holds for double dashed lane markings also. One of the approaches adopted by many researchers to deal with this issue is to subject the image to a preprocessing step of inverse perspective mapping. By doing this, the perspective effect is removed and all markings appear to be of uniform size. However, one issue with this is high computational costs, and the other is the quality of markings in the mapped images, that decreases as we move further away from the camera and this affects the matching score with the templates. In the lane marking detection method proposed in this thesis in Chapter 3, the techniques are invariant to scale changes. The first step of feature extraction by applying *SLEM* would yield the straight line edges of the dashed markings, irrespective of the scale of the markings. Similarly, the outlier removal step using signed edges and Hough Transform are not affected by the scale of the markings. This is because the left and right signed edges are determined based on the direction of edge transition and not based on the size of the features. Also, the edges of the different dashes are all aligned to have the same (ρ, θ) values and hence all markings jointly contribute to the peak height when Hough Transform is applied. It was shown earlier in Chapter 3 that the dashed lane markings were picked up successfully in video sequences irrespective of their size.

In the context of horizontal road markings, effect of scale is reflected in the thickness of the marking, which increases with vehicle proximity to the marking, as shown in Fig. 5.27. In our algorithm for detecting horizontal markings using the `blmdetect()` module (Sec. 3.11), ‘thickness criteria’ parameter is set to accommodate for the variations of thickness in the near view. In the case of zigzag markings, a similar phenomena as solid linear lane marking is witnessed, as the zigzag shows a continuous presence as long as the vehicle is moving in the zone of the marking. But the only difference is that the direction of the constituent line strokes in the near view keeps changing in an alternate fashion, as against single solid lane markings that appear unchanging in their appearance. However, the technique proposed for zigzag marking detection in Sec. 5.5.2 is not specific to any particular configuration of these strokes its detection.

Changing the viewpoint distorts the appearance of longitudinal linear markings by changing the angle of orientation of the constituent lines. This applies to all kinds of longitudinal markings, such as solid or dashed or zigzag lane boundary markings, that are single or double in nature. A example of this is illustrated in Fig. 5.29. It can be seen that as the viewpoint changes, the angle made by the solid lines with the reference

Figure 5.27: Unchanging appearance of linear markings along the road with scale changes.

Figure 5.28: Unchanging appearance of linear markings along the road with scale changes.

axes changes. In the proposed methods for linear marking detection in Chapters 3 and 5, one of the parameters that is fed into the algorithm as priori information is the range of angles that the markings are expected to be in. This range is set based on the allowable changes that can happen to these angles due to the yawing of the vehicle within the lane. Changes in viewpoint do not have a visible impact in the case of horizontal markings, given that the thickness of horizontal markings is mainly affected by scale.

Figure 5.29: Effect of viewpoint changes on linear marking detection.

To summarize, scale does not affect the detection of solid lane markings, such as single or double linear markings along the lane and zigzags. Scale-invariance is also not a concern in the case of dashed linear lane markings, in techniques that do not treat each dash as a separate object. Horizontal lines are affected by scale only with respect to thickness, which can also be easily tackled. Therefore, scale-invariance is

5. Complex Linear Road Markings

not a major concern when it comes to detecting linear markings. As for viewpoint changes, all types of longitudinal markings undergo a change in their angle, and this is easy to incorporate in line detection based methods. Horizontal linear markings remain unaffected by viewpoint changes. Therefore, scale and viewpoint invariance is not a major concern when dealing with linear markings.

5.8 Architecture for Detecting BLM and CLM

Figure 5.30: Overall block architecture for detection of simple and complex linear road markings.

The core computational module for detecting the various kinds of linear markings, including basic and complex linear markings, is presented in Fig 5.31. It largely resembles the computational module for BLM detection, that was presented in 4.2. However,

5.8 Architecture for Detecting BLM and CLM

the peak detection module is now replaced with the AHPD proposed in this chapter, in order to cater to detection of complex line markings. This modified functional block using the AHPD is now referred to as `blmDetect_AHPD()`, with its inputs being the region of interest (RoI), the range of angles to search for lines in GAH and HT ($\delta\theta$) and thickness criteria, given by δd . The output is S , corresponding to the set of valid BLMs detected (5.27). This core module is reused to detect various kinds of basic and complex linear markings as shown in Fig. 5.30, with the help of supporting modules `countBLM()`, `zigzagDetect()` and `chkContinuous()`.

Figure 5.31: Block architecture for complex linear road markings detection.

`countBLM()` analyses the output set S of `blmDetect_AHPD()`, and generates a binary output n that can either take the values of '0' or '1' depending on whether the markings are found to be 'single' or 'double'. This is done by simply checking the number of parallel BLMs detected that satisfy the properties of lane markings. Based on the algorithm that was presented for zigzag marking detection in 3, function `zigzagDetect()` outputs a binary value z , depending on whether a zigzag is detected or not. Function `chkContinuous()` takes the parameters 'spatial' or 'temporal', and determines whether the marking is solid or dashed, as described in Chapter 3 Sec. 3.10.

$$S = \text{blmDetect_AHPD}(\Delta\theta, I_{ROI}, \Delta d) n = \text{countBLM}(S) z = \text{zigzagDetect}(S) \quad (5.27)$$

5. Complex Linear Road Markings

where S corresponds to set of m valid line pairs detected, given by

$$\{ \{ (\rho_{l1}, \theta_{l1}), (\rho_{r1}, \theta_{r1}) \}, \{ (\rho_{l2}, \theta_{l2}), (\rho_{r2}, \theta_{r2}) \} \dots \{ (\rho_{lm}, \theta_{lm}) \} \}.$$

Fig. 5.30 summarizes how the core module of `blmDetect_AHPD()` is used in a cascaded fashion along with the other supporting wrapper modules described above, to detect and distinguish the various basic and complex linear road markings.

5.9 Summary

In this chapter, techniques for detecting complex linear markings, including double linear markings and zigzags, were presented. Firstly, the challenges in detecting complex linear markings were addressed, arising primarily from the limitations of the conventional Hough peak detection. Next, an adaptive Hough peak detection (AHPD) method was proposed that is able to detect closeby lines and lines of varying lengths. The BLM detection module proposed in Chapter 3 is now improved using the AHPD, and the resulting function is called as `blmDetect_AHPD`. This improved BLM detection module using AHPD is then deployed to detect double linear markings and zigzag markings. Results for detecting double linear markings and zigzag markings are presented. Finally, a computational architecture that consolidates detection of all linear markings is presented, by cascading the core modules. In the next chapter, a study on detecting arrow markings is presented, which requires scale and distortion invariance.

Detecting Arrow Markings

6.1 Introduction

Figure 6.1: (a) Potential scenario that would benefit from a ‘microGPS’, (b) Snapshot of some advanced features of Tomtom [1].

Arrow markings indicate valid traffic flow directions in the lanes, and help to channelize the traffic smoothly, thereby mitigating chaotic traffic or jamming at junctions. Existing literature on detecting arrow markings, although limited in number, mainly aim to provide the driver with additional information for enhanced decision making [9, 17, 23, 24, 41]. However, arrow markings can be used for assisted navigation as well. Consider a scenario wherein a vehicle is on a multiple lane highway, being guided by a GPS based navigation system. Now, if the next GPS instruction is to ‘TURN RIGHT’,

6. Detecting Arrow Markings

and if the vehicle is in the center lane, then it needs to first move to the rightmost lane, before taking the right turn. This is illustrated in Fig. 6.1 (a). Now, what if we had an advanced ‘microGPS’ system that would not only guide the driver to take a right turn, but would also be intelligent enough to advise the driver to move to the right lane, so as to be prepared to take the right turn? Imagine the current ‘state’ of the vehicle is determined based on the type of arrow marking on the host lane and the adjacent lanes, and this real-time outcome is used to assist the driver in micro-navigation from one lane to the next. In other words, the valid traffic flow direction indicated by the arrow markings can be exploited to come up with a more comprehensive navigation system. However, even some of the most advanced navigation systems available in the market today do not give any lane-level navigation based on the current state the vehicle is in. Tomtom [1] and Garmin Nuvi [150], rated as the best navigation systems in the market today, as per a survey in [151], only go as far as using pre-stored map data to provide a more detailed display image of the junction, including relevant road signs, numbers and directions (Fig. 6.1 (b)). It is envisaged that extraction and classification of arrow markings on the host lane and adjacent lanes, and integrating this information to the existing GPS systems could lead to advanced lane navigation systems.

In this chapter, the challenges in detecting road arrow markings are addressed, mainly covering the issue of scale and distortion invariance. Limitations of existing methods to tackle this problem are also outlined. Next, novel techniques to detect and classify arrows in road scenes are presented using scale invariant shape signatures. These are based on decomposition of the arrows into their component parts and the use of signed edge maps to uniquely define each arrow type. The validation of the method on the arrows dataset is then performed, and the misdetections are analysed. Next, detection of arrows in side lanes are addressed by relying on host lane arrows as reference. Results of arrow detection and classification in side lanes is presented next.

Typically, arrows on the road surface are of the following types: (a) Simple Forward Arrow, (b) Simple Right Arrow, (c) Simple Left Arrow, (d) Left/Right Arrow, (e) Forward/Left Arrow, (f) Forward/Right Arrow and (g) Forward/Left/Right Arrow, as shown in Fig. 6.2.

Figure 6.2: Arrow types on roads.

6.2 Scale & Distortion Invariance for Arrow Marking Detection

Unlike linear markings that are comprised of a set of simple line strokes, that are easy to detect even in the presence of scale changes and distortion, arrow markings are more complex shapes that are formed by criss-crossing of multiple lines, each with a different (ρ, θ) value. This means that any viewpoint changes or perspective, causes each constituent line of the shape to be distorted differently, based on its position and orientation. This results in multiple possible shape definitions if all allowable distortions are considered. An example with an arrow marking captured from different viewpoints is shown in Fig. 6.3. As it can be seen, changing the viewpoint drastically changes the appearance of the marking accordingly.

Figure 6.3: Varying appearance of road marking with changing viewpoints.

6. Detecting Arrow Markings

Moreover, unlike lane boundary markings that run along the lane as repeatable patterns, arrow markings are localized because they are confined within a specific area, and have a distinct boundary that encloses them. It may be recalled that it was due to the repeatability in solid linear lane markings, that the overall appearance remains unaffected by scale changes (Sec. 5.7). However, in the case of arrows, due to their non-repeating nature, any change in the scale causes a more distinct change in the overall appearance of the marking. To illustrate this, let us consider the example of two sample sequences shown in Fig. 6.4 (a) and (b), where the camera is moving towards the arrow markings. The distinct change in the appearance of the arrow markings due to the scale change is apparent.

Figure 6.4: Changing scales of arrow marking with vehicle motion.

The above reinforces the need for an efficient scale-invariant and distortion-invariant algorithm for detection of complex shape markings, such as arrow markings and pedestrian markings.

6.2.1 Limitations of Existing Methods

Existing recent literature on road arrow marking detection was previously outlined in Chapter 3, although low in number. Most of them treat arrow markings as shapes, and use template matching to detect them [17, 40, 42]. Template matching based approaches are shown to be effective in known environments, where parameters such as viewpoint, scale etc. of the markings are known to be fixed. However, in environments where these are variable, template matching based object detection has been pointed out to be computationally infeasible [143]. This is because, for an invariant template matching, the whole template matching process needs to be repeated for templates of various scales, skews etc., thereby resulting in a very large number of underlying computations. In fact, many recent popular shape detection methods deal with the scale problem by performing an exhaustive search over all possible object positions and scales [152, 153, 154]. This exhaustive search imposes severe constraints, both on the detectors computational complexity and on its discriminance, since a large number of potential false positives need to be excluded [155].

To understand this better, one of the common template matching algorithms called normalized cross correlation (NCC) is discussed. For a sub-image $w(x, y)$ of size $K \times L$ within an image $f(x, y)$ of size $M \times N$, the normalized correlation between $w(x, y)$ and $f(x, y)$ at a point (i, j) is given by

$$C(i, j) = \frac{\sum_{x=0}^{K-1} \sum_{y=0}^{L-1} (w(x, y) - \bar{w}) (f(x+i, y+j) - \bar{f}(i, j))}{\left[\sum_{x=0}^{K-1} \sum_{y=0}^{L-1} (w(x, y) - \bar{w})^2 \right]^{1/2} \left[\sum_{x=0}^{K-1} \sum_{y=0}^{L-1} (f(x+i, y+j) - \bar{f}(i, j))^2 \right]^{1/2}} \quad (6.1)$$

where $i = 0, 1, \dots, M-1, j = 0, 1, \dots, N-1$, $\bar{w}$ is the average intensity value of the pixels in the template w . The variable $\bar{f}(i, j)$ is the average value of f in the region coincident with the current location of w . C lies in the range -1 to 1 and is independent of scale changes in the intensity values of f and w .

This is illustrated in Fig. 6.5, with top left corner of image f being the origin. Template w is moved, pixel-by-pixel around the entire image area and correlation value C is calculated for every position. The position where C is maximum is where w best matches with f . From equation 6.1, it can be seen that the correlation operation is multiplication-intensive and considering that the correlation needs to be performed

6. Detecting Arrow Markings

Figure 6.5: Illustration of classic correlation.

for every position of the template, the overall computational complexity and time is sizeable. It has been reported in [113] that the popular template matching methods such as Normalized Cross Correlation operation does not meet speed requirements for many applications.

It might be argued that an alternative less complex approach could be to analyse the straight line features of the arrow marking using HT. However, directly analysing the Hough space to detect the arrow signature is neither conclusive nor straightforward. This is because of the following reasons:

1. Firstly, the different (ρ, θ) pairs of the arrow edges, for a given arrow type, do not always occur at the same pre-defined (ρ, θ) positions. This is because the arrows can vary in position, scale and orientation. Therefore, it is more than just a simple check for the presence or absence of peaks in pre-defined (ρ, θ) positions. Rules need to be generated, that associate peaks with each other, based on ‘relative’ position and orientation of peaks w.r.t other peaks.
2. The number of lines forming an arrow is atleast six. This means a sizeable number of rules to associate the peaks with each other.
3. The (ρ, θ) values of the arrow edges can span the entire range of angles in Hough space. Because of the large number of allowable (ρ, θ) combinations, stray lines

6.3 Scale & Distortion Invariant Shape Signatures for Defining Arrows

can cause false positives. Any stray line falling in that angle range could be easily mistaken for a valid edge of an arrow.

4. Moreover, peak detection in Hough space inherently suffers from the problem of clutter. For these reasons, the Hough space alone is not often used to detect patterns.

Fig. 6.6(a) & (b) show Hough accumulation spaces of forward arrows corresponding to the forward arrows in Fig. 6.3(a) & (c) respectively. It can be seen from Fig. 6.6 that the two Hough spaces look substantially different with different number of peaks, locations of peaks and relative association between them.

Figure 6.6: Variations in Hough spaces generated by same arrow marking from different viewpoints: (a) shows Hough space of arrow in Fig. 6.3(b), and (b) shows Hough space of arrow in Fig. 6.3(c).

6.3 Scale & Distortion Invariant Shape Signatures for Defining Arrows

In this section the proposed method to detect arrow markings on roads is described in detail. The proposed method comprises three main steps: (1) Signed Edge Maps Generation (2) Hough Transform Computation (3) Arrow Signature Detection in Hough Accumulation Space. Before we propose the different methods, we first define the region of interest (RoI) where we look for the arrow markings. The ‘near-view’ in the host lane is taken as the RoI, where near-view corresponds to the bottom slice of the image closest to the camera. The host lane region is extracted from the previously proposed

6. Detecting Arrow Markings

lane marking detection techniques in Chapter 3, and is considered to be an input for arrow detection.

The first step in the proposed arrow marking detection is the generation of ‘signed edge maps’ from the edge detection process. Signed edges were previously introduced in Chapter 3 and subsequently used in Chapter 5. The process is briefly recapped below and further explained in the context of arrow markings. The conventional edge detection process involves applying gradient kernels such as Sobel [97] on the RoI I to give gradient maps G_x and G_y in the x and y directions. G_x and G_y are then used to compute the gradient magnitudes G . The edge map E is computed by thresholding G with threshold Th . Sometimes, just the horizontal or vertical gradient maps are thresholded to yield partial edge maps E_x or E_y respectively. The following equations put the entire conventional edge map generation in a nutshell:

$$\begin{aligned} G_x &= I \otimes S_x & G_y &= I \otimes S_y \\ G &= \sqrt{G_x^2 + G_y^2} \\ E &= G > Th, & E_x &= |G_x| > Th, & E_y &= |G_y| > Th \end{aligned} \quad (6.2)$$

It can be seen that conventionally, the sign of gradient value will not affect the decision to qualify that pixel as an edge pixel in the edge map. We now propose to use the gradient signs to decompose the horizontal edge map E_x further into two signed edge maps called positive and negative edge maps, E_{x+} , and E_{x-} , such that:

$$\begin{aligned} E_{x+}(x, y) &= 1 \text{ if } G_x(x, y) \geq 0 \wedge E_x(x, y) = 1 \\ E_{x-}(x, y) &= 1 \text{ if } G_x(x, y) < 0 \wedge E_x(x, y) = 1 \\ E_x &= E_{x+} \cup E_{x-} \end{aligned} \quad (6.3)$$

Let us now understand what these signed edges mean, specifically in the context of arrow markings. We identify an arrow marking as a light colored closed shape, bounded by linear edges, having a dark background (road surface). If we scan the image from left to right, then we encounter one set of bounding edges of the arrow, which have dark $\rightarrow$ light intensity transitions ($\rightarrow$ is used to denote direction of intensity transition), and another set of edges, which have light $\rightarrow$ dark intensity transitions. This direction of the intensity transition is captured by the sign of the edge map. E_{x+} has the edge pixels that are formed by dark $\rightarrow$ light transition when we scan the image in from left to right, i.e. x -direction. E_{x-} comprises edges that are formed by light $\rightarrow$ dark transition in the x -direction. This also implies that E_{x+} contains the edges on the left of the arrow, and E_{x-} has the right edges of the arrow, if the image were to be scanned from left to right.

6.3 Scale & Distortion Invariant Shape Signatures for Defining Arrows

For ease of understanding and explanation, we call E_{x+} as the left edge map E_l and E_{x-} as the right edge map E_r , in the rest of the chapter. The decomposed edge maps are processed further in subsequent steps to ascertain the presence of arrow markings.

The rationale behind the decomposition of E_x into the signed edge maps is two-fold. Firstly, we bring in an additional constraint in the line search, by specifying the exact direction of the intensity transitions. By doing so, we can, not only look out for a given position or angular orientation of the lines, but also whether they are bounding edges in the left or right of the arrow. In other words, these signed edge maps help us to propose unique signatures to identify each type of arrow. Secondly, they help in reducing the complexity of Hough accumulation space, thereby reducing the effort needed in its analysis. Since the edge map is split into simpler edge maps, the Hough accumulation space is also split into simpler spaces that can be analyzed faster and effectively.

Fig. 6.7 shows an example of the two signed edge maps E_l and E_r for an arrow image. It can be seen that the left edges of the arrow are captured in E_l and the right edges in E_r in Fig. 6.7(b) and (c) respectively.

Figure 6.7: (a) Sample forward arrow (b) E_l (c) E_r .

After the edge map is decomposed into E_l and E_r , we apply Hough transform (HT) on both the edge maps to generate Hough accumulation spaces H_l and H_r respectively. The different linear edges in E_l and E_r are captured as peaks in H_l and H_r . The (ρ, θ) values corresponding to the peaks in the Hough spaces are stored separately in P_l and P_r , i.e. the linear edges in the left edge map E_l are now captured in $P_l = \{(\rho_{l1}, \theta_{l1}), (\rho_{l2}, \theta_{l2}), \dots, (\rho_{ln}, \theta_{ln})\}$. Similarly the other peaks' set P_r is generated. These values will now be analyzed to look for the signatures of the arrow markings.

6. Detecting Arrow Markings

The next step is to search for arrow signatures in the generated peak sets P_l and P_r . In order to do this, we first define arrow signatures that are unique to each of the arrow types, and also scale and distortion invariant, using the decomposed edge maps E_l and E_r . We break the arrow into its component parts and build the arrow signature using signatures of each of its parts. From Fig. 6.2, it can be seen that an arrow of any type comprises of a main arrow tail and one or more of the following types of arrow heads (1) Forward facing arrow head, (2) Left arrow head and (3) Right arrow head.

Figure 6.8: Left-slant (solid) and right-slant (dashed) lines are indicated on different arrow parts: (a) Main arrow tail, (b) Forward facing arrow head, (c) Right arrow head, and (d) Left arrow head.

It can be observed that each of these arrow parts can be defined by a pair of lines - one slanting to the left (referred to as left-slant line) and the other to the right (right-slant line). Fig. 6.8 shows the left and right slant lines as solid and dashed lines respectively, that define each of the arrow parts. A line is tagged as a right-slant or a left-slant line based on the θ value of the line. If a line l_1 is given by (ρ_1, θ_1) , then l_1 is tagged as left-slant or right-slant based on the equations below:

$$l_1 = \left\{ \begin{array}{l} \text{right slant if } 15^\circ \leq \theta_1 \leq 85^\circ \\ \text{left slant if } -85^\circ \leq \theta_1 \leq -15^\circ \end{array} \right\} \quad (6.4)$$

These right and left slant lines are captured in E_x and in turn reflected in the decomposed signed edge maps E_l and E_r , based on the nature of intensity transitions (defined by equation (6.3)). Let us take the example of the arrow head in Fig. 6.8(b). There is one right-slant line and one left-slant line. The right-slant lines forms the left edge of the arrow head. Hence it is captured in E_l . Likewise, the left-slant line is captured in E_r .

6.4 Arrow Detection and Classification using Shape Signatures

Similarly, for each part of the arrow, the left-slant and right-slant lines of the pair occur either both in E_l , or both in E_r , or one in E_l and the other in E_r . For example, the main arrow tail for all arrow types will have one right-slant line that is captured in the left edge map E_l and one right-slant line that is captured in the right edge map E_r . Whereas, the right arrow head will have both the left-slant and right-slant lines occurring in E_r . Table 6.1 lists the edge maps in which the right and left slant lines are captured for each part of the arrows. Based on this table, we first extract the

Table 6.1: Location of Right and Left Slant Lines for Each Arrow Part

Arrow Part	Right Slant	Left Slant
Main Arrow Tail	1 in E_l	1 in E_r
Forward Arrow Head	1 in E_l	1 in E_r
Left Arrow Head	1 in E_l	1 in E_l
Right Arrow Head	1 in E_r	1 in E_r

candidate line pairs for each constituent part of the arrow and subject them to a set of simple association rules to confirm the presence of that part. The main arrow tail is first detected, followed by the detection of arrow heads, as explained below.

6.4 Arrow Detection and Classification using Shape Signatures

6.4.1 Detecting the Main Arrow Tail

The main arrow tail is the part common to all arrow types. Also, it is one of the most prominent features of the arrow markings. Hence, we first detect the presence of the main arrow tail to infer the presence of an arrow marking. In order to do this, all line pairs that are potential candidates representing the main arrow tail are first extracted. From Table 6.1, the main arrow tail is supposed to have one right-slant line in E_l and one left-slant line in E_r . Therefore, all right-slant lines in E_l and all left-slant lines in E_r are selected and paired up in various combinations to yield the candidate line pairs. An illustration of all the candidate line pairs extracted in this step for arrow tail detection, is shown in Fig. 6.9 (the top row), for a simple forward arrow. As there

6. Detecting Arrow Markings

are two left-slant lines in E_r and two right-slant lines in E_l , in the case of a simple forward arrow, this results in a total of four line pairs, as shown in Fig. 6.9. These lines are given by the (ρ, θ) tuples in P_l and P_r , i.e. $P_l = \{(\rho_{l1}, \theta_{l1}), (\rho_{l2}, \theta_{l2})\}$ and $P_r = \{(\rho_{r1}, \theta_{r1}), (\rho_{r2}, \theta_{r2})\}$.

Figure 6.9: (a)-(d): Top row shows the candidate line pairs for main arrow tail of a simple forward arrow; Bottom row shows the axis for each of the line pairs.

But only one of these pairs is a valid pair that actually forms the bounding edges of the arrow tail. If (ρ_{l1}, θ_{l1}) and (ρ_{r1}, θ_{r1}) correspond to the right and left slant lines of the main arrow tail, then we need associate these two pairs together as the bounding edges of the main arrow tail, i.e. (ρ_{l1}, θ_{l1}) must be paired with (ρ_{r1}, θ_{r1}) only, and not with (ρ_{r2}, θ_{r2}) , in P_r .

This is accomplished by a simple association rule based on the ‘axis’ of each candidate line pair. Axis is defined as the line that equally divides the angle of intersection of the two component lines. The association rule that we set on each candidate pair, to confirm the presence main arrow tail is that, the axis of the valid line pair passes through the vanishing point. This is because the main arrow tail is always pointed towards the vanishing point, as illustrated in Fig. 6.10 (a). The vanishing point is obtained by the intersection of the left and right lane markings, as shown by the solid black lines in Fig. 6.10 (a). It can be seen that the axis of the main arrow tail (shown as dashed line) passes through the vanishing point.

Now applying this constraint on the four candidate line pairs shown in the top row of Fig. 6.9, the line pairs Fig. 6.9(b) & (d) are eliminated. This is because, for the forward arrow that is considered in Fig. 6.9, the vanishing point is vertically in front

Figure 6.10: Sample road scene showing: (a) axis of main arrow tail passing through the vanishing point, (b) axis of side arrow head being a horizontal.

of the arrow head, which is satisfied by line pairs in Fig. 6.9(a) & (c). Since the main arrow tail is invariably longer than the arrow head, we can further confirm that the line pair in Fig. 6.9 (c) is the main arrow tail, using the Hough peak heights.

Mathematically, this constraint with the axis of the main arrow tail is checked in the following way, for every pair of right and left slant lines. Referring to Fig. 6.11, given the (ρ, θ) tuples of right and left slant lines as (ρ_l, θ_l) and (ρ_r, θ_r) , respectively, the equations of right and left-slant lines, rs and ls , are given by:

$$\begin{aligned} \text{right-slant}(rs) : \quad x \cos \theta_l + y \sin \theta_l &= \rho_l \\ \text{left-slant}(ls) : \quad x \cos \theta_r + y \sin \theta_r &= \rho_r \end{aligned} \quad (6.5)$$

Consider any two horizontal lines drawn at y_1 and y_2 such that they intersect two slant lines rs and ls at points P_1, P_2, P_3 and P_4 , as shown in Fig. 6.11. The points of intersection are computed by substituting y_1 and y_2 in the equations in (6.5). If P_{m1} and P_{m2} are the mid points of P_1 and P_2 , and P_3 and P_4 , respectively, then the axis is defined by the line $P_{m1}P_{m2}$. Now in order to verify if this axis passes through the vanishing point P_{vp} , we check the collinearity of P_{m1}, P_{m2} and P_{vp} .

Thus, the arrow tail being the most prominent feature is used as the main indicator to detect the initial presence of potential arrow markings, before performing subsequent steps to verify the arrow, as shown in sections to follow.

Figure 6.11: Figure showing how the axis of main arrow tail is computed.

6.4.2 Detecting Arrow Heads

Once the detection of main arrow tail is confirmed, we now look for the other arrow components in exactly the same way as the main arrow tail. The only difference is that: (1) the line pairing will vary for each part according to the listing in Table 6.1, and (2) the orientation of the axis is dependent on the direction of the arrow head. The axis is pointed towards the vanishing point in the case of forward arrow head, whereas the axis is found to be a horizontal line for right and left arrow heads. An example is shown in Fig. 6.10(b) for right arrow head using a dashed line.

6.4.3 Arrow Classification based on Sub-components of Arrows

Once we infer the presence or absence of the various arrow sub-components based on their signatures, as explained in the previous sections, we can confirm the presence of the arrow marking and classify it. Table 6.2 summarizes the various combinations of arrow parts that define each arrow type.

The entire algorithm to detect different arrow markings is illustrated in Fig. 6.12, showing all the processing steps.

6.4 Arrow Detection and Classification using Shape Signatures

Table 6.2: Arrow Classification using Presence/Absence of Arrow Parts

Type of Arrow	Main Arrow Tail	Forward Arrow Head	Left Arrow Head	Right Arrow Head
Forward Only	✓	✓		
Left Only	✓		✓	
Right Only	✓			✓
Forward & Left Turn	✓	✓	✓	
Forward & Right Turn	✓	✓		✓
Left & Right Turn	✓		✓	✓
Anywhere Turn	✓	✓	✓	✓

Figure 6.12: Block diagram showing the proposed method to detect different arrow markings.

Role of AHPD in Extracting Arrow Features : The arrow marking detection algorithm described above relies on the lines detected using AHPD for the peak detection step of HT, proposed earlier in Chapter 5. The role of AHPD in detecting these markings is significant, and in order to demonstrate this, the edges detected by AHPD and MATLAB `houghpeaks()` are shown in Fig. 6.13 for a sample arrow marking. It can be seen that AHPD is able detect the line features of the arrow marking successfully

6. Detecting Arrow Markings

without any duplicates. On the other hand the `houghpeaks()` shows many duplicate lines as a result of redundant peak detections. Dealing with the problem of redundant lines explained in Chapter 5 is particularly important for arrow detection. The reason is that this may lead to multiple and sometimes incorrect associations between the left and right edge maps. This is resolved successfully by the use of AHPD. Also, another problem with conventional methods as explained in Chapter 5, was manual threshold adjustments that were required to cater to lines of varying lengths. This is applicable to arrow markings where the tail edges are much longer compared to those of the arrow heads. It is shown here that the AHPD is able to detect linear edges of varying lengths in an automated manner, as shown in Fig. 6.13.

Figure 6.13: Comparison between AHPD and MATLAB `houghpeaks()` on arrow image (a) Hough space of edge map shown in (b), (c) lines detected using `houghpeaks()` (d) lines detected using AHPD (e) AHPD result with an alternate visualization.

6.5 Results & Discussion

In this section, we present the results of arrow marking detection and classification using the proposed method. Considering there were no publicly available datasets containing different kinds of arrow markings on the road, the arrow-dataset consisting of 289

Table 6.3: Results of Arrow Detection

Arrow Type	No. of Samples	No. of Successful Detections
forward	125	123
left	45	43
right	47	46
forward-left	24	23
forward-right	23	22
left-right	12	11
forward-left-right	13	12
non-arrow samples	250	249

images was created from actual road test drives conducted as part of this research. These images were selected such that they represent various illumination, weather and shadow conditions. The number of samples of each arrow type are summarized in Table 6.3. Apart from these samples with valid arrow markings, the arrow marking detection algorithm was also evaluated on 250 negative samples with no arrow markings present in the near-view. The images considered are of size 720×1280 , and the block settings of 90×128 that was previously set in Chapter 3 for SLEM generation are maintained. The bottom-most row of image blocks was taken as the RoI for arrow detection and images were chosen such that the arrows are present in the RoI.

The results are summarized in Table 6.3. A detection is considered successful when the algorithm detects the presence of an arrow marking and also classifies it successfully. First, the arrow tail is detected to give the initial indication of the presence of arrow marking, followed by the detection of arrow head in order to confirm the arrow type. The detection outcome is compared against the arrow type visually. It can be seen that the algorithm yields an average detection rate of 97%. Also, the rate of true negatives is as high as 99.99%, as 249 out of 250 negative image samples successfully did not show any arrow marking in the near-view.

Fig. 6.14 shows the some of test cases and the detection result (as inset) using the proposed method. It can be seen that the proposed method is successfully able to detect the different types of arrows shown in Fig. 6.14. The scale and distortion invariance of the proposed method is illustrated in Fig. 6.15. The two images in the left column (Fig. 6.15(a)) show the detection of the forward arrow marking of two different scales as the vehicle moved forward. In two images in the right column (Fig. 6.15(b)), the

6. Detecting Arrow Markings

Figure 6.14: Different types of arrow markings are detected using the proposed method.

forward-right turn arrow varies in scale as well as orientation. The proposed method is able to detect the above arrows successfully, invariant to both scale and distortion.

Figure 6.15: Scale and distortion invariance of proposed method.

There are a few cases when the proposed method failed to detect the arrow markings. It was seen that this happens mostly when the arrow markings are faded to an extent that the block-based SLEM does not detect the edge features to start with. Also, some cases where the arrow markings were partially occluded by vehicles show errors in classification.

Once the arrow marking detection algorithm was evaluated on still images as shown

above, it was incorporated into image sequences. The RoI in this case is set a little differently. The bottom two rows of the image within the host lane are taken as the RoI. First, the penultimate row of blocks (row of blocks above the bottom most row) is monitored for the presence of arrow markings. If no arrow is detected, then the algorithm overlays a “No Arrow” symbol on the top left of the image as the output. However, if the arrow tail is detected, then it indicates a potential arrow marking approaching, and thus an “Arrow Ahead” symbol is displayed. Once the arrow presence is detected in the penultimate row, then the bottom-most row is now monitored for both detection and classification of the arrow. The final arrow type is then displaced in the output. The rationale behind this is that, the near-view in the bottom most row of blocks provides a sufficiently clear display of the arrow to enable classification, while the presence of arrow itself can be triggered earlier on. An illustration of the arrow detection on image sequences is shown in Fig. 6.16. While it can be argued that some worded markings like STOP may also have features that resemble an arrow tail and can potentially lead to wrong indication of arrow presence, these cases can be easily rejected once the arrow-head detection step is performed.

The computation complexity of the proposed arrow detection method is now computed and compared with a recent method in [23]. In order to determine the computation complexity of the proposed method, it is divided into two parts. The first part is to compute the edge maps and the Hough transform. The second part is determining the association between the different line segments. The computation complexity of the first part is dependent on the total number of pixels in an $M_b \times N_b$ sized block, i.e. $O(M_b N_b)$. The computation complexity of the second part is dependent on the number of line segments detected in the block (or adjacent blocks). Therefore, if N_l represents the number of line segments detected in a block, the computation complexity of the proposed algorithm is $O(N_l)$.

When compared to existing arrow detection methods like [23], the proposed method has lower computation complexity. The overall computation cost of detection and identification of arrows in [23] is higher because of the computation of Haar wavelets, which are fed into a multi-class SVM classifier. Unlike the proposed method, [23] also depends on the number of scales n_s for which the SVM is trained. Therefore, in addition to the pre-processing cost of generating the inverse perspective map in [23], which is equivalent to $O(M_b N_b)$ (same as the first stage of the proposed method), [23] also incurs

6. Detecting Arrow Markings

Figure 6.16: Output of proposed arrow detection and classification on an image sequence; Outputs (i) “No Arrow” when no arrow seen in near-view (ii) “Arrow Detected” when arrow is detected, but not classified yet (iii) arrow type when arrow is close enough to perform classification.

an additional complexity of the order of $O(n_s n_c)$, where n_s is the number of scales for which the detection is being done and n_c is the number of SVM classes for which the classification is being performed. Given that $n_s n_c > N_l$, the overall computation cost of the arrow detection method in [23] is higher than the proposed method.

6.6 Detecting Arrow Markings in Side Lanes

So far, we have presented techniques to detect arrow markings in the host lane. However, in order to apply arrow detection outputs to advanced navigation, as suggested in Sec 6.1, it is necessary to also have an idea about the nature of arrow markings in the immediate neighboring lanes (the terms ‘neighboring lane’ and ‘side lane’ are used interchangeably). For example, in the illustration shown in Fig. 6.1 (a), if the vehicle is to change lane first to get onto the rightmost lane, before taking a right turn, then it would not only be useful to know that the current lane does not support a direct right turn, but the lane right of it has a right-arrow painted on it, confirming the possibility of a right turn upon moving onto it.

Detecting arrow markings in side lanes is a challenging problem because of two reasons. One is that the markings on the side lanes are further away from the center of the camera, as compared to those on the host lanes. As a result of this, the markings appear less focussed and fainter. This poses a challenge in the edge detection step. Secondly, the degree of shape distortion due to sideward perspective is very high. This leads to lateral compression of the features, thus making it difficult to distinguish the various arrow parts from each other. An example of a forward-right arrow in the neighboring lane and host lane is shown in Fig. 6.17 (a) and (b) respectively. It can be seen that the arrow marking is fainter and exhibits compressed features in the neighboring lane, compared to the same type of marking on the host lane, where it is more prominent and less distorted laterally. Therefore, we cannot directly apply the algorithm that was developed for detecting arrow markings in the host lanes, for the neighboring lanes. In Sec. 6.6.1, we propose a simplified algorithm for detecting arrow markings in the side lanes.

6. Detecting Arrow Markings

Figure 6.17: Forward-left arrow in host lane versus neighboring lane (a) faint marking and compressed features (b) prominent features and less distortion.

6.6.1 Algorithm

In order to deal with the above mentioned challenges, various clues that can be inferred from the outcome of arrow detection and classification in the host lane, are relied upon. Firstly, the arrow markings are all usually drawn at the same level, next to each other, in a parallel fashion in the adjacent lanes, when seen from a bird's eye view. This means that the position of the arrow marking in the host lane already gives an indication of the position of the arrows in the neighboring lanes. Secondly, the angular orientation of the arrow markings in the host and neighboring lanes have a pre-defined relationship based on the lane width, which is a priori information. Next, depending on the type of arrow on the host lane, only a set of arrow types are allowed on the neighboring lanes. For example, if the host lane has a forward-left arrow marking, then the lane left of it can only have a left arrow marking. All other markings lead to vehicle conflict on the lanes, and hence ruled out automatically. List of allowable arrow marking combinations found on lanes is summarized in Table 6.4.

The first step in the algorithm is to check whether or not there is a neighboring lane on either side, before attempting to detect any arrow marking there. Checking whether a neighboring lane exists or not can be done by simply checking the type of lane marking that bounds the host lane. From previously proposed techniques in Chapter 3, we are in a position to detect presence of solid linear markings (single or double) or zigzag markings, which are the kind of markings that typically indicate road boundaries. A dashed lane marking is an indicator for the presence of neighboring lanes. Only if a neighboring lane is detected, the algorithm proceeds to predict the location of the arrows in it, using parameters such as axis of arrow, angular orientation

6.6 Detecting Arrow Markings in Side Lanes

Table 6.4: Possible Arrow Combinations

Left Lane	Host Lane	Right Lane
forward left forward-left	forward	forward right forward-right
left	left	left right forward forward-left forward-right forward-left-right left-right
right left forward forward-left forward-right forward-left-right left-right	right	right
left	forward-left	right forward forward-right
left forward forward-left	forward-right	
left	left-right	right
left	forward-left-right	right

and horizontal boundaries. This prediction is done based on information about host lane arrow markings, using the following equations:

$$\begin{aligned}\theta_N &= \theta_H - \Delta\theta \\ \rho_N &= \begin{bmatrix} x_{vpt} \\ y_{vpt} \end{bmatrix}^T \begin{bmatrix} \cos \theta_n \\ \sin \theta_n \end{bmatrix}\end{aligned}\tag{6.6}$$

where (ρ_H, θ_H) denotes the parameters of the axis of the arrow in host lane, (ρ_N, θ_N) denotes axis of neighboring arrows, VPT refers to the vanishing point given by coordinates x_{vpt} and y_{vpt} , as shown in Fig. 6.18. The equation above can be explained from the observation that the axis of the arrows in all lanes pass through the vanishing point. Therefore, given the axis of the arrow in the host lane and the vanishing point, we derive the axis of the arrow in the neighboring lanes. Through heuristics, $\Delta\theta$ is set to be approximately 30° .

6. Detecting Arrow Markings

Figure 6.18: Relationship between arrows in host and side lanes.

After deducing the axis of the neighboring arrow given by (ρ_N, θ_N) , we then derive the horizontal upper and lower boundaries of the arrow marking, by using fact that the heights of the arrows in all lanes are aligned with each other. In Fig. 6.18, it is illustrated that arrows in all lanes share the same upper and lower boundaries, defined by lines l and m respectively. The algorithm incorporates this horizontal cropping to get the region of interest, by selecting the corresponding rows of blocks where arrows are expected. Considering that classification of arrows in the host lane was triggered only when the arrow has moved into near-view (taken as bottom-most row of blocks), it would suffice to choose this same row of blocks for neighboring arrows as well.

Once the blocks of interest are chosen, we first confirm the presence of the arrow by detecting the arrow tail, as that is the most prominent arrow feature, even in the neighboring lanes. In order to do this, we look for a pair of lines separated by a small distance d , whose axis has similar (ρ, θ) values as that of the predicted axis in Step 1. A similar process was described previously for host lane arrow tail detection, in Sec 6.4.1. The only difference is that in the case of host lane, the line pair consisted of a left slant line and a right slant line. However, in the case of left neighboring lane, both the lines are expected to be left slanted; and similarly in the case of right neighboring lanes, both the lines are expected to be right slanted. The left and right edges of the line pair are searched in the left and right edge maps respectively, and a check is performed to confirm that their axis passes through the vanishing point. If such a valid line pair is found, then that is confirmed to be the tail of the arrow in the neighboring lane.

6.6 Detecting Arrow Markings in Side Lanes

Given that the arrow tail is one of the most prominent features in an arrow marking, irrespective of which lane it is found in, the chances of finding this is quite high. Also, given the expected (ρ, θ) values of the tail axis, the peak detection step in the Hough Transform is robust, as the search space is reduced.

Figure 6.19: Deducing arrow type based on edge pixel count in arrow head masks.

Once the presence of arrow in the side lanes is confirmed, we then proceed to classify the arrow marking. As pointed out earlier, the arrows in the side lanes do not compressed features that are not easy to detect. Hence, detecting signed edge shape signatures like we did for the case of host lanes does not yield conclusive results. Given that priori information from the host lane arrow has helped to already narrow down the arrow on the side lanes, we generate simple rules based on distribution of edge pixels. For this, five mask regions are generated and the number of edge pixels in each of these masks is studied to deduce the arrow type. The masks denoted by A, B, C, D and E are depicted in Fig. 6.19. These masks are derived from lines l_1 to l_9 , represented below, where (ρ_N, θ_N) is the parametric representation of the axis of the arrow, w and h are width and height of the masks, set heuristically, l_8 and l_9 represent the left and right edges of the arrow tail respectively, after it is detected in the step above.

$$\begin{aligned}
 &axis : (\rho_N, \theta_N) \\
 &l_1 : (\rho - 3w/2, \theta_N) \\
 &l_2 : (\rho - w/2, \theta_N) \\
 &l_3 : (\rho + w/2, \theta_N) \\
 &l_4 : (\rho + 3w/2, \theta_N) \\
 &l_5 : x = x_1 \\
 &l_6 : x = x_1 + h \\
 &l_7 : x = x_1 + 2h
 \end{aligned} \tag{6.7}$$

6. Detecting Arrow Markings

The above lines are used to generate masks as below:

$$\begin{aligned}
 A &: (l_1 > 0) \cap (l_2 < 0) \cap (l_5 > 0) \cap (l_6 < 0) \\
 B &: (l_2 > 0) \cap (l_3 < 0) \cap (l_5 > 0) \cap (l_6 < 0) \\
 C &: (l_3 > 0) \cap (l_4 < 0) \cap (l_5 > 0) \cap (l_6 < 0) \\
 D &: (l_1 > 0) \cap (l_8 < 0) \cap (l_6 > 0) \cap (l_7 < 0) \\
 E &: (l_9 > 0) \cap (l_4 < 0) \cap (l_6 > 0) \cap (l_7 < 0)
 \end{aligned} \tag{6.8}$$

Once the masks are generated, the edge pixel count is computed in each of these masks.

Table 6.5: Classification of Arrows in Neighboring Lanes using Edge Counts in Masks

Type	A+D	B	C+E	D	E
Forward Only		✓			
Left Only	✓				
Right Only			✓		
Forward & Left Turn		✓		✓	
Forward & Right Turn		✓			✓
Left & Right Turn	✓		✓		
Anywhere Turn		✓		✓	✓

If the number of edge pixels exceeds a threshold in that mask, then it is considered to have a high edge activity. Based on the distribution of edge pixels in these various masks, the type of the arrow marking is deduced, using Table 6.5.

The algorithm for detecting arrows in side lanes was evaluated on 120 images, which represented a total of 165 side arrows, considering that not all images had arrows on both sides of the host lane. Out of these, 123 of them gave correct detections leading to an accuracy of 75 %. It is observed that the detection rate of arrows in side lanes is much lesser than that in host lanes, mainly because the features are more compressed and subjected to higher distortions compared to those in host lanes. The techniques have potential to be developed into more robust algorithms and this has been identified as an item for future work in Chapter 8.

6.7 Architecture for Arrow Marking Detection

Fig. 6.21 shows the various computational blocks involved in detecting arrows in host and side lanes. The BLM detection module is reused by invoking the GAH, signed edge generation, HT and the AHPD modules. The linear edges extracted are sent into

6.7 Architecture for Arrow Marking Detection

Figure 6.20: Flowchart summarizing steps in arrow detection in side lanes.

6. Detecting Arrow Markings

Figure 6.21: Block diagram summarising computational modules for arrow detection and classification.

the arrow detection module in host lanes. Here the lines are first associated using the vanishing point criterion to check for presence of arrow tail. Once the presence of arrow is confirmed by detecting the tail, the arrow heads are studied for classifying the arrows into different types. The output of the arrow detection in host lane is then fed into the side lane arrow detection module. The location of the host lane arrow is used for generating the region of interest for the side lane arrows. Further, the possible arrow combinations in host and side lanes are fed in to improve robustness of the detection algorithm.

6.8 Summary

In this chapter, techniques to detect and classify arrows in road scenes are presented. The decomposition of the arrows into their sub-components and the use of signed edge maps, enable generation of simple signatures that are scale and distortion invariant, to uniquely define each arrow type. It has been shown how the left and right edge maps, coupled with axis direction, suffice to constrain the lines belonging to arrows and reject all non-arrow candidates. The arrow detection method proposed in this chapter has been evaluated on more than 250 images containing arrow markings in

the near-view, and it was shown to yield a detection accuracy of about 97%. Next, a method was proposed to detect arrow markings in side lanes, by relying on the arrow detection outputs in the host lane. Finally, an architecture for arrow marking detection is presented.

In the next chapter, scale and distortion invariant techniques to detect pedestrian markings are presented.

6. Detecting Arrow Markings

7

Detecting Pedestrian Markings

7.1 Introduction

Detecting pedestrian crossings and alerting about their presence in advance to the driver, is one of the key steps towards traffic safety. As shown in Sec. 2.4.3 of Chapter 2, existing pedestrian crossing detection techniques mainly cater to assist the blind by automatically detecting them. However, these techniques cannot be directly applied for an ADAS application to assist the driver, mainly because the views of the pedestrian marking are orthogonal to each other in the two applications (Fig. 7.1). In this chapter, a scale and distortion invariant algorithm to detect pedestrian markings (as seen by the driver) is proposed. The appearance of the pedestrian markings is vulnerable to scale and distortion effects, and this necessitates a scale and distortion invariant detection technique for detecting pedestrian markings. Considering that the pedestrian crossing is made of a collection of linear segments, previously proposed techniques for extracting linear features are reused. The approach is further strengthened by using visual and non-visual cues to predict the presence of an upcoming pedestrian markings.

7.2 Scale & Distortion Invariant Pedestrian Marking Detection

Pedestrian markings can be treated as shapes, typically formed by thick white strips in the longitudinal direction, along with a horizontal line that binds them together. As in the case of other ‘shape’ markings such as arrows, pedestrian markings also are

7. Detecting Pedestrian Markings

Figure 7.1: View of a pedestrian crossing (a) as seen by a pedestrian (b) as seen by a driver.

subjected to the effects of scale and distortions that are perceivable with change in distance or viewpoint of the camera with respect to the marking. Fig. 7.2 illustrates the same pedestrian markings with varying scales and viewpoints. Fig. 7.2 (a) and (c) are captured at varying scales, while Fig. 7.2 (a), (b) and (d) are captured with varying camera viewpoints. It can be observed that overall appearance of the pedestrian marking undergoes significant changes with changing scales and viewpoints.

Application of conventional shape matching approaches would therefore require a set of templates that represent all these variations in scale and distortion. As was shown in the case of arrows (Sec. 6.3 of Chapter 6), the computation cost with the use of multiple templates for matching is significantly high.

7.3 Scale & Distortion Invariant Shape Signatures for Defining Pedestrian Markings

In this section, an algorithm is proposed that detects the presence of pedestrian marking using scale and distortion invariant signatures. Let us first define the signature of the pedestrian marking. Fig. 7.3 shows two views of the pedestrian marking: top view in Fig. 7.3(a), and the perspective view as seen from an on-board camera in Fig. 7.3(b). As seen from top-view, pedestrian markings can be defined by a set of thick white linear stripes, drawn parallel to each other, in the direction of the lane. In addition to the stripes, we also have a horizontal line binding the stripes. This top-view or real-world view of the lane marking, when viewed from on-board camera has the stripes narrowing down and pointing towards the vanishing point, and the stripes are still bound by the horizontal marking. Based on these observations, and referring to the Fig. 7.3(b), we

7.3 Scale & Distortion Invariant Shape Signatures for Defining Pedestrian Markings

Figure 7.2: Scale and distortion effects on pedestrian marking detection (a) and (c) are captured at varying scales, while (a), (b) and (d) are captured with varying camera viewpoints.

Figure 7.3: (a) Aerial view and (b) Perspective view as seen from on-board camera, based on which signature of pedestrian marking is defined.

define the following properties for the signature of a pedestrian marking using edges in the following way:

1. The pedestrian marking is a set of bounded edges, which comprises two subsets of edges, i.e.

$$S_P = S_H \cup S_E \quad (7.1)$$

where S_H corresponds to horizontal marking, and S_E contains the edges of the stripes.

7. Detecting Pedestrian Markings

2. The horizontal marking indicates the beginning of the stripes, i.e. the height in the image where the edges in S_E start to appear is indicated by the edges in S_H .
3. The host lane markings bind the edges of the stripes on either side, i.e. the edges in S_E must lie between the left and right lane markings.
4. The linear edges in S_E , i.e. the stripes are formed by the alternating dark-to-light and light-to-dark transition of intensities.
5. All the edges in S_E pass through the vanishing point.
6. The pedestrian marking runs uniformly across the length of the whole road, and are uniformly spaced in the horizontal direction.

7.4 Pedestrian Marking Detection Using Shape Signatures

Figure 7.4: Image sequence illustrating the various components of pedestrian marking signature that are perceived at various scales (a) Horizontal is the most prominent when vehicle is furthest away (b) Both horizontal and stripes are seen when vehicle moves closer to pedestrian marking (c) Stripes are the most prominent features when vehicle is at the pedestrian marking.

In this section, a technique is proposed that detects the signature of the pedestrian marking invariant to its changing scales. It was observed that the *entire* signature given by the set S_P is not visible or perceivable in the beginning because of the scaling effect of the signature. Fig. 7.4 shows the different scales of the pedestrian marking as the vehicle approaches the marking. It can be seen that in Fig. 7.4 (a), except for the horizontal marking S_{H1} , the other edges of the stripes cannot be seen even by the driver. As the vehicle approaches the marking, the entire pedestrian marking becomes more visible and discernible. In Fig. 7.4 (b), both the horizontal and the stripes are visible, and finally when the vehicle is at the pedestrian marking (Fig. 7.4 (c)), the stripes are the most prominent features of the pedestrian marking that are observed.

7.4 Pedestrian Marking Detection Using Shape Signatures

The proposed algorithm addresses these scale changes in the following way. The first step in the proposed algorithm is to detect the edges of the first horizontal marking, i.e. S_{H1} because that is visible to the driver even before the entire signature is seen. This is accomplished using the proposed **blmDetect** module with horizontal input settings (discussed previously in Chapter 3). When a horizontal marking is detected, the proposed algorithm monitors the region above the horizontal marking for possible edges that form the stripes, i.e. S_E , in every frame thereafter. In other words, in every frame F_i , we check for the horizontal marking first. If a horizontal marking is detected, the host region above the horizontal marking (region bound by the left and right markings of host lane) (Property 3 above), is checked for edges in S_E . If they are not found, we repeat the same process in the next frame, i.e. F_{i+1} . S_E may not be perceivable till F_{i+p} , when the proposed algorithm to detect the stripes starts detecting S_E . After S_E edges are detected in F_{i+p} -th frame, both horizontal and stripes signature are monitored in frames thereafter. In other words i.e. S_H and S_E are continuously monitored in every frame.

When it is found that S_H is no longer visible but S_E is still visible, it is understood that the vehicle is either close to the pedestrian marking or crossing it. After the vehicle crosses S_E , the pedestrian marking indicator is flagged off. We will show next, how the varying scale of the markings does not affect the detection of the different components of the pedestrian marking shape signature.

In Chapter 5, it was shown that the scale changes for horizontal BLMs are not evident. The length of the horizontal BLMs change as the vehicle approaches them. But the proposed **blmDetect** by its very nature of functioning handles scale and perspective changes, given that the thickness of the horizontal marking lies in a predictable range.

Next, the stripes in the pedestrian marking are addressed. Based on the observations above, the edges that form the pedestrian markings will show the following properties. The fourth property (discussed above) implies the presence of a set of edges S_E such that S_E can be defined in the following manner:

$$S_E = \{l_1^+, l_1^-, l_2^+, l_2^-, l_3^+, l_3^-\} \quad (7.2)$$

where l_i represents a linear edge given by (ρ, θ) tuple from the Hough transform. The $+$ or $-$ in the superscript of l_i indicates whether the linear edge is formed by the left edge or the right edge respectively. In other words if l_i is a linear edge of the pedestrian

7. Detecting Pedestrian Markings

marking that is formed because of dark-to-light transition, then we have l_i^+ . Similarly we have l_i^- if the linear edge is formed by light-to-dark transition of gradients. In the above equation (7.2), the set S_E gives an alternating arrangement of edges that are formed from the two different transitions.

Next, each of these lines in S_E must pass through the vanishing point, i.e. if a line l_i given by (ρ_i, θ_i) is taken (the $+/-$ is dropped for brevity), then the vanishing point $V_{pt}(x_{vp}, y_{vp})$ must satisfy the line l_i . All the lines in S_E must satisfy this condition.

The third observation gives the information of the bounding lines between which the set of lines in S_E must be found. If the left and right lane markings are given by lines $l_L(\rho_L, \theta_L)$ and $l_R(\rho_R, \theta_R)$ respectively, then all $l_i \in S_E$ must line between l_L and l_R .

Figure 7.5: Properties displayed by edges of stripes (S_E).

These properties and observations are used to find the set of lines S_E in the following manner. We have with us the locations of the left and right lane markings, i.e. $l_L(\rho_L, \theta_L)$ and $l_R(\rho_R, \theta_R)$. Referring to Fig. 7.5, we show these lane markings along with the edges in S_E that need to be detected. It can be seen from Fig. 7.5, a given linear edges $l_i(\rho_i, \theta_i)$, that forms the pedestrian markings, satisfies the following relationship:

$$\theta_L < \theta_i < \theta_R. \quad (7.3)$$

This relationship is useful in more than one way in the proposed method to extract the pedestrian marking. For the time being, let us assume that the region of interest (RoI), where we look for the pedestrian marking, is known (we will show how this is set in the next section).

7.4 Pedestrian Marking Detection Using Shape Signatures

Given the horizontal gradient magnitudes $|\nabla_x I|$, we get the left and right edge maps as E_l and E_r using the following equations:

$$\begin{aligned} E_l &= |\nabla_x I| > 0 \\ E_r &= |\nabla_x I| < 0 \end{aligned} \quad (7.4)$$

SLEM is used to further reduce the edge points in E_l and E_r because the angle range in GAH, where the peaks need to be searched for, is given by the range in (7.3).

Hough transform is then applied on E_l and E_r to extract the lines in each edge maps. Let L^+ and L^- be the sets of lines detected by the HT from E_l and E_r respectively. The angle range in (7.3) can be used for HT computation also to reduce the computational complexity (the computation cost analysis will be discussed in later). Therefore, if L^+ and L^- are defined as follows:

$$\begin{aligned} L^+ &= \{l_1^+, l_2^+, l_3^+ \dots\} \\ L^- &= \{l_1^-, l_2^-, l_3^- \dots\} \end{aligned} \quad (7.5)$$

where $l_i^+ = (\rho_i^+, \theta_i^+)$ and $l_i^- = (\rho_i^-, \theta_i^-)$. We then check if the lines detected in L^+ and L^- are passing through the vanishing point. If a line l_i is not passing through the vanishing point, it is removed from L^+ and L^- as an outlier or noise.

After removing such outliers, we now check for the alternating pattern of dark-to-light and light-to-dark transiting edges using the edge information we have in L^+ and L^- . Let the angles corresponding to the lines in L^+ and L^- be denoted by Θ^+ and Θ^- respectively, i.e.,

$$\begin{aligned} \Theta^+ &= \{\theta_1^+, \theta_2^+, \theta_3^+ \dots\} \\ \Theta^- &= \{\theta_1^-, \theta_2^-, \theta_3^- \dots\} \end{aligned} \quad (7.6)$$

These two separate angle lists Θ^+ and Θ^- are then combined into a single array Θ , i.e.,

$$\Theta = \Theta^+ \cup \Theta^- = \{\theta_1^+, \theta_2^+, \dots, \theta_n^+, \theta_1^-, \theta_2^-, \dots, \theta_n^-\} \quad (7.7)$$

If the pedestrian marking is present, then the linear edges in L^+ and L^- should show an alternating sign of the edge, when sorted based on the angle. In other words, the i -th linear left edge l_i^+ is sandwiched between the $(i-1)$ -th linear right edge, i.e. l_{i-1}^- , and the i -th linear right edge, l_i^- . Similarly, the i -th linear right edge should be preceded by i -th linear left edge and $(i+1)$ -th linear left edge. This can be seen in Fig.

7. Detecting Pedestrian Markings

7.5. In terms of the angles of these linear edges, the following conditions are seen for the i -th left edge and i -th right edge,

$$\theta_{i-1}^- < \theta_i^+ < \theta_i^- \quad (7.8)$$

$$\theta_i^+ < \theta_i^- < \theta_{i+1}^+ \quad (7.9)$$

Combining the above conditions for all the angles in Θ in (7.7), we will get the following:

$$\theta_L > \theta_1^+ > \theta_1^- > \theta_2^+ > \theta_2^- > \theta_3^+ > \theta_3^- \dots \theta_R \quad (7.10)$$

The above relationship implies that if the linear edges are arranged in the increasing order of their angles, an alternating pattern of the edges is obtained, i.e., a left edge followed by a right edge, which is then followed by a left edge again. The bounding edges, that bind the pedestrian crossing markings, are given by the lane marking edges that define the left and right boundaries of the RoI (given by θ_L and θ_R in the above equations). Therefore, we sort Θ and check if the above mentioned occurrence of the left and right edges happens.

Table 7.1 summarizes the (ρ, θ) values of each of the lines in the RoI of Fig. 7.5. It can be seen that when the lines are arranged based on increasing θ values, then their edges show an alternating pattern of their edge type.

Figure 7.6: Lettered markings that can be potentially mistaken for a pedestrian marking.

Though the detection of the above signature is sufficient in most cases to detect the presence of pedestrian markings, there is one possible situation, where a false positive could be detected based on the above criteria. This is the case when there are lettered markings on the road. Words like STOP, SLOW, X-ING AHEAD, STRIPS etc. painted on the road (Fig. 7.6) can also give linear edges that satisfy all the above conditions, i.e. alternating edges, edges passing through vanishing point and linear edges bound between the lanes.

7.4 Pedestrian Marking Detection Using Shape Signatures

In order to remove this particular case of false positives, an additional condition is added to conclusively determine the presence of a pedestrian marking. This condition is based on the width of the white stripes of the pedestrian marking. Considering that the pedestrian crossing has wider white stripes than other alphabetical linear markings, it can be seen that no more than 3 or 4 such stripes of the pedestrian crossing can be drawn within a lane.

Given that the pedestrian marking is bound by the left and right lanes, $l_L(\rho_L, \theta_L)$ and $l_R(\rho_R, \theta_R)$, the width of the lane at a given a -coordinate is given by:

$$w_x = y_R - y_L \quad (7.11)$$

$$= \frac{\rho_R - x \cos \theta_R}{\sin \theta_R} - \frac{\rho_L - x \cos \theta_L}{\sin \theta_L} \quad (7.12)$$

Based on the observation on the number of pedestrian crossing stripes, the distance between the alternating linear edges is given by:

$$\frac{w_x}{5} \leq w_s \leq \frac{w_x}{4} \quad (7.13)$$

where w_s is the width of the lane between two alternating edges. Therefore, if we consider two consecutive edges, i.e. one left edge l_i^+ and one right edge l_i^- , which are satisfying the angles' condition in (7.10), we compute w_s for this pair of linear edges at the same x -coordinate in the following way:

$$w_s = y_{l_i^-} - y_{l_i^+} \quad (7.14)$$

$$= \frac{\rho_i^- - x \cos \theta_i^-}{\sin \theta_i^-} - \frac{\rho_i^+ - x \cos \theta_i^+}{\sin \theta_i^+} \quad (7.15)$$

In case of road markings like STOP, w_s will not satisfy the width condition in (7.11). The above width criteria can also be used to eliminate multiple edges that may be detected due to presence of other features like tar color changes etc. on the pedestrian crossing. For example, let us assume the following sequence of angles:

$$\theta_L > \theta_1^+ > \theta_1^- > \theta_2^+ > \theta_{21}^+ > \theta_2^- > \theta_3^+ > \theta_3^- \dots \theta_R \quad (7.16)$$

It can be seen from above that there are two θ s between θ_1^- and θ_2^- , i.e. there are two left edges given by θ_2^+ and θ_{21}^+ , which need to associated with θ_2^- and θ_1^- . Also assume

7. Detecting Pedestrian Markings

that both these linear edges pass through the vanishing point. Therefore, if these edges in the above angle range do belong to a pedestrian crossing, only one of the two linear edges is a valid left edge. In order to find the correct left edge, the width criteria is applied in (7.11) and (7.14) to each edge, i.e. θ_2^+ and θ_{21}^+ , to check the width w_s . If these edges belong to a pedestrian crossing, one of them will satisfy the width criteria. Accordingly, the other edge is rejected.

The flow chart in Fig. 7.7 describes the entire algorithm.

Figure 7.7: Flowchart describing steps for detecting pedestrian marking.

7.5 Other Markings as Cues for Pedestrian Marking Detection

Detecting pedestrian markings well in time is critical for the safety of the pedestrians and the driver. In order to facilitate this, there are various other visual and non-visual cues that can be used to predict the possible presence of a pedestrian marking. We now present two different scenarios where a pedestrian marking is usually found; and the different *additional* visual and non-visual cues for each scenario are presented (these scenarios and cues are considered for Singapore road scenario, which follows the

7.5 Other Markings as Cues for Pedestrian Marking Detection

British road safety standards). The term *additional* refers to the cues that are present in addition to the pedestrian marking. These cues are found on the road surface well before the actual pedestrian marking.

Figure 7.8: Component parts of a pedestrian crossing region (A) Zigzag marking (B) Horizontal linear marking (C) Pedestrian Crossing.

Pedestrian Crossing on Straight Road In this scenario, we are considering the case when there is a pedestrian crossing across the straight road. In this case, the pedestrian crossing is usually preceded by single zigzag marking and a horizontal line (Fig. 7.8). The horizontal marking was already addressed as part of the signature of pedestrian marking. However, it is once again included here along with other markings that usually accompany pedestrian markings, for sake of completeness. The presence of single zigzag marking along the lane is an indicator for an approaching pedestrian crossing. Thereafter, the presence of a horizontal basic linear marking across the lane indicates a stop line before the pedestrian crossing. Therefore, when a single zigzag marking is detected, the driver can be informed for the possible presence of a pedestrian crossing ahead. Also, this starting of zigzag marking can also be used to set the region of interest to detect the pedestrian crossing. The proposed algorithm of detecting the pedestrian marking is complimented by the detection of zigzag marking. The zigzag marking can be detected using the techniques proposed earlier in Sec. 5.5.2 of Chapter 5. When a zigzag is detected, we set an RoI spanning two rows of the image to detect the pedestrian crossing with that RoI. Before detecting the pedestrian marking, the BLM-detect module is used to detect the horizontal line across the lane in the selected RoI. If this line is detected, this is a strong indicator for the presence of a pedestrian marking.

7. Detecting Pedestrian Markings

The proposed technique for detecting this marking is then triggered to conclusively detect the presence of pedestrian marking.

Therefore, it is proposed to cascade the previously developed zigzag detection module and horizontal BLM detection modules, along with the pedestrian marking detection module, as shown in Fig. 7.9.

Figure 7.9: Cascading computational modules for detecting pedestrian markings.

Thus, the presence of an upcoming pedestrian marking can be predicted when a zigzag marking and a horizontal marking are detected sequentially, one after the other. Computational techniques for detecting zigzag lines and horizontal lines were presented in Chapter 5 and 3 respectively. If a zigzag marking is first detected, followed by a horizontal marking, then the likelihood of a pedestrian marking following them, is very high.

Pedestrian Crossing on Left Branch Road It is very common for pedestrian crossings to be found on left branch roads, especially when the left branch road is preceded by a zigzag marking. Therefore, the visual cues that can be used to predict the presence of pedestrian markings in such scenarios are: presence of a zigzag along the lane, and the presence of a left arrow on the host lane prior to entering into the turn. Just like in the previous scenario, the presence of a zigzag is a strong indicator for approaching pedestrian marking, even in the case of left branch roads. To indicate that the vehicle is about to enter into a left turn, visual cues like left-turning arrows on the host lane preceding the turn, or non-visual cues such as the left indicator signal can also be integrated. It is to be noted that the setting of the region of interest cannot be simply done by using current lane position of the history information of lane positions, because of the turn. Based on the camera calibration, the image blocks that roughly comprise of the region ahead on the host lane are considered as the RoI in such cases.

7.6 Results

Table 7.1: Detection of Stripes in Pedestrian Markings

Test Image	Line parameters
Line l_N	θ_N	ρ_N	δ_E
	l_1^+	623	55.92	+
	l_1^-	480	80.54	-
	l_2^+	-188	110.92	+
	l_2^-	74	134.09	-
	l_3^+	254	149.53	+
	l_3^-	297	154.84	-
Line l_N	θ_N	ρ_N	δ_E
	l_1^+	593	54.68	+
	l_1^-	418	83.66	-
	l_2^+	-87	117.41	+
	l_2^-	117	136.53	-
	l_3^+	249	149.04	+
	l_3^-	283	153.43	-
Line l_N	θ_N	ρ_N	δ_E
	l_1^+	763	34.16	+
	l_1^-	785	45.00	-
	l_2^+	742	66.04	+
	l_2^-	-445	101.31	-
	l_3^+	-237	119.48	+
	l_3^-	-7	136.40	-
Line l_N	θ_N	ρ_N	δ_E
	l_1^+	623	55.92	+
	l_1^-	480	80.54	-
	l_2^+	-188	110.92	+
	l_2^-	74	134.09	-
	l_3^+	254	149.53	+
	l_3^-	297	154.84	-
Line l_N	θ_N	ρ_N	δ_E
	l_1^+	769	30.74	+
	l_1^-	800	40.03	-
	l_2^+	768	62.74	+
	l_2^-	609	87.88	-
	l_3^+	-286	116.57	+
	l_3^-	9	137.73	-

7. Detecting Pedestrian Markings

Line l_N	θ_N	ρ_N	δ_E
l_1^+	749	30.96	+
l_1^-	765	42.71	-
l_2^+	694	66.80	+
l_2^-	-416	99.46	-
l_3^+	-45	128.66	+
l_3^-	91	139.97	-

The proposed algorithm was validated on a total of 41 video sequences, divided into two sets. The first 15 sequences contain valid pedestrian markings, and the remaining 21 sequences do not contain pedestrian markings. The 15 sequences with pedestrian markings include the two scenarios described above - the straight road scenario as well as the left branch road scenario. The 21 sequences without pedestrian markings were considered for robust evaluation against negative samples, and are the same test sequences used in Chapter 3 for validating the lane detection and classification.

The validation was done using MATLAB based only on the algorithm described in Fig. 7.7, and does not include integration of multiple cues for prediction, as described in Sec 7.5. The latter has been discussed in this chapter as a proposed means to make the algorithm more robust, and developing an integrated system combining all the cues has been included as an item for future work. Table 7.1 shows the intermediate (ρ, θ) values of the lines of stripes detected within the host lane region, and the alternating directions after sorting the lines based on the angle. The first two entries in the table are for pedestrian markings in left branch road scenario. 7.10 shows a sample sequence with the detection output displayed on the top left corner of the image. It can be seen that once the pedestrian marking moves into the near view, the algorithm detects its presence.

It was observed that the algorithm yields a 100% detection accuracy, wherein all the 15 pedestrian markings were detected in the first set containing positives, and no pedestrian markings were detected in the second set with negatives. It must be noted that although the horizontal line is used as the initial indicator to detect the presence of pedestrian markings, in some cases the horizontal line is missing. However, the algorithm to detect the stripes is conclusive and leads to a successful detection of pedestrian markings even in the absence of horizontal lines.

Figure 7.10: Pedestrian marking detection on a video sequence.

The steps described in the proposed method are independent of both scale and distortion variations, thus making the detection of pedestrian markings independent of scale, and distortions due to perspective and viewpoint changes.

The computation complexity of the pedestrian detection algorithm involves two components. The first component is generating the Hough space using pixels in the region of interest. The second component is looking for the association between the linear edges as shown in Section 7.3. This involves a series of comparisons and is dependent on the number of linear segments detected. The computation complexity of this step is given by $O(N_l)$ where N_l is the number of linear edges detected in the region of interest. Considering that HT is the most computationally complex operation in the proposed technique, and the proposed AHT reduces the computation cost drastically, the overall computation cost of the pedestrian detection algorithm is also reduced significantly. This coupled with appropriate computation platform should pave the way for high-speed implementations.

7.7 Architecture for Pedestrian Marking Detection

The block diagram summarizing the overall process for pedestrian marking detection is shown in Fig. 7.11. The pedestrian marking detection process described in this chapter relies on some of the computational modules proposed in the earlier chapters. Firstly, to get the region of interest defined by the left and right lane markings, the `blmDetect_AHPD()` module is used. Next, in order to detect a zigzag marking that is one of the strongest indicators of an upcoming pedestrian marking, the previously proposed `zigzagDetect()` module is used. A horizontal marking following the zigzag marking is detected by appropriately configuring the `blmDetect_AHPD()`. An additional left arrow marking in the case of left branched road scenario, can be detected by the `arrowDetect()` module described in Chapter 6.

Figure 7.11: Block architecture summarizing pedestrian marking detection.

7.8 Summary

In this chapter, the signature of a pedestrian marking, as seen by the driver, is first described using fundamental edge features. Unique properties satisfied by the lines bounding the pedestrian marking are identified, and these properties form the basis of the pedestrian marking detection algorithm presented in this chapter. As a means to make the detection more robust, it is suggested to use other markings such as a zigzag marking followed by a horizontal line, which together serve as strong indicators

for an upcoming pedestrian marking. The method was evaluated against 41 test videos and it was shown to have a 100 % detection accuracy. A discussion on scale-distortion invariance has been included. Finally, a block architecture integrating the various road marking detection modules in the context of pedestrian markings, has been presented.

7. Detecting Pedestrian Markings

Conclusions and Future Research

Contributions described in this thesis have led to the development of compute-efficient novel techniques that are capable of detecting road markings such as lanes, arrows and pedestrian crossings.

The proposed approach for detecting single lane markings systematically exploits the underlying properties of lane markings to achieve robustness and computational efficiency. It was shown that shadows from trees and other randomly distributed edge pixels can be effectively removed using the proposed gradient angle histograms (GAH) based analysis in the feature extraction step. The more geometrically regular edge artifacts that closely resemble lane markings are removed by employing Hough Transform (HT) on the signed edges together with the distance criterion in the outlier removal step.

It was shown that the pruned edge map (called Straight Line Edge Map of SLEM) generated using GAH can substantially collapse the complexity of HT computations. Further, the angle ranges for HT computations can be reduced by referring to the GAH peaks that represent potential lines. The reduced edge map also leads to relatively less cluttered Hough spaces, thereby strengthening the HT peak detection process. The use of temporal history information further enhances the detection of host lanes by extracting outliers across image frames.

It was established that the block size plays a crucial role to influence the quality of the SLEM. For example, a relatively large block size was shown to include most of the edge pixels in the original edge map, thereby resulting in an SLEM with high number of unwanted edge candidates. On the other hand, setting a relatively small

8. Conclusions and Future Research

block size compared to the line length eliminates many valid lines of interest. It was established that the block size must be appropriately chosen to ensure that the lines within the block are sufficiently prominent to facilitate robust identification. It was inferred from experimental study that a block setting of 90×128 yields the best quality SLEM for the test images of size 720×1280 . The proposed single lane marking detection method was shown to yield a high detection accuracy of approximately 98% when evaluated on an extensive dataset of over 6,000 images, encompassing various complex road scenarios such as shadows from trees, varying illumination and weather conditions.

The proposed two-pronged approach based on continuity analysis in spatial and temporal domains efficiently distinguishes between solid and dashed lane markings. Additionally, it was shown that the analysis of the temporal relationship is sufficient to determine the characteristics (i.e. closely spaced or sparse) of dashed lane markings. The proposed BLM detection module can be readily configured to detect BLMs of any orientation including horizontal markings.

The proposed novel HT approach called Additive Hough Transform (AHT) exploits the additive property of HT to replace complex trigonometric computations merely with additions, thereby drastically collapsing the computational complexity of HT. Unlike existing architectures, which operate on the edge map serially, AHT computes the HT in parallel using a grid based architecture. It was shown that for a 128×128 image with 5% edge content, AHT results in a speed up of at least 67 times against existing methods, when an 8×8 grid was used. It was also shown that the area-time complexity AHT can be further managed by appropriate selection of the grid size.

Conventional HT peak detection methods are not capable of detecting complex linear markings (CLM) such as double linear markings and zigzag markings. This prompted the development of novel techniques to overcome the limitations of conventional methods. It is shown that the proposed adaptive Hough peak detection (AHPD) algorithm can successfully detect lines of close proximity and of varying length, without the need to manually set the threshold value or window size for non-maximal suppression, thereby overcoming the severe limitations of conventional peak detection approaches. The AHPD involves the use of an iterative climbing process to adaptively cope with varying noise levels in the Hough space. Also, it is shown to differentiate effectively, local maxima from a valid peak in the reduced Hough space, by using shape

properties. The signed edge maps, used as inputs to the HT process, are also used to aid the AHPD algorithm for separating line features of close proximity into left and right edge maps.

The AHPD forms the basis for extracting line features of uneven lengths in zigzags and lines of close proximity in double linear markings. Further, the scale and distortion effects in the detection of linear markings can be easily managed by altering the angle windows in the BLM detection module. The proposed algorithms for detecting zigzag lines and double linear markings based on the AHPD were shown yield detection rates of 98% and 100%, respectively.

Shape signatures were introduced to ensure that the proposed method for detecting arrow markings is scale and distortion invariant. Decomposing of the arrow into its constituent parts by representing each part with a unique Hough-based signature has led to a novel scale-invariant representation of the arrows. This made it possible for arrow markings to exhibit distinct characteristics from other features on the road. Associating the signatures with the vanishing point makes them more robust, thereby limiting the number of false positives. It was shown that the proposed method is computationally efficient when compared to other existing methods that rely on exhaustive matching operations performed using templates of varying scale and perspective to achieve scale-invariance and distortion-invariance. Evaluation of the proposed method using a dataset containing more than 250 arrow markings of different types confirms that the detection accuracy can be as much as 97%. It was observed that arrows in the immediate neighbor lanes often suffer from greater perspective distortion and compression. In order to overcome the challenges posed by arrows in the immediate neighbor lanes, information from host lane arrows were applied to enhance robustness of the detection process.

The proposed scale-invariant and distortion-invariant shape signatures associated with pedestrian markings represent inherent vertical stripes and horizontal lines. The shape signatures of the pedestrian markings together with the characteristics such as unusually large thickness of the vertical stripes and the vanishing point criteria are replied upon to distinguish pedestrian markings more affirmatively. This has resulted in a more robust and efficient algorithm for identifying pedestrian crossings. Techniques developed for SLEM, signed edges and HT were redeployed to locate other markings such as zigzag and horizontal markings to further ascertain the presence of upcoming

8. Conclusions and Future Research

pedestrian marking on straight roads. It was observed that signatures of the vertical stripes alone contribute significantly to the detection of the pedestrian markings.

Finally, the computationally efficient techniques proposed in this thesis for detecting the various road markings can be efficiently integrated to realize a core module of a high-performance ADAS to enhance driver assistance.

8.1 Suggestions for Future Research

We recommend the following directions for further research:

- Although the road marking detection algorithms proposed in this work yield high detection accuracy, further investigations must be carried out to cope with irregular patterns associated with road boundaries that are often marked with stripes, etc. It is therefore necessary to examine the road boundaries as a special case so as to prevent miss-detections.
- Signature based approach for detecting arrows has significantly improved the detection accuracy for arrows in the host lane. However, more work is needed to enhance the robustness of arrow detection in side lanes so that severe distortions due to angle of view and distance from the camera can be managed efficiently. Use of high resolution cameras together with more robust scale invariant signatures should help improve the detection rate.
- The proposed techniques for road marking detection were evaluated using test images obtained in Singapore. This captures essential considerations such as tree shadows and local traffic rules and regulations. It is worth extending the proposed techniques to accommodate the variations in the road markings associated with different traffic rules and regulations of other countries. Extending the work to cope with all weather and lighting conditions is also necessary.
- The various compute efficient architectures proposed in this thesis to detect the different road markings can be incorporated into an embedded computing platform to evaluate the system for real-life deployment. This will necessitate the tweaking of the algorithms to align with the characteristics of the chosen camera and exploiting the inherent parallelism to deliver real-time performance.

- In the proposed research, detection of road markings in the near view of the host vehicle has been considered. It would be interesting to extend the detection of road markings in the far-view so that drivers can be notified in advance. This may necessitate the investigations into more complex road models of higher orders as lanes can be curved in far view. In addition, robust methods must be developed to overcome the challenges posed by fainter road markings that will be subjected to much higher perspective distortions due to them being farther away from the camera.
- Four major categories of road markings are tackled in this research work. It is worth exploring the possibility of incorporating the detection of other markings such as chevrons and worded markings such as STOP and SLOW. The core techniques proposed in this thesis should form the basis for extending the work to accommodate the detection of additional road markings.
- It is increasingly becoming the norm for drivers to carry GPS enabled devices such as car navigation systems and mobile phones. This coupled with the local geographical information should make it possible to enhance detection of road markings using camera sensors. This may pave the way for a more robust detection of road markings in low visibility and those that fade away over time. The wireless communication overhead must be kept to a minimum in order to make it attractive for low-cost deployment.
- The arrow detection module could be combined with the vehicle navigation system to further assist the driver to promptly move to the appropriate lane. For example, if the instruction from the navigation system is to turn right, then the arrow marking in the host lane must also correspond to the intended maneuver. This will also contribute to improving road safety and to minimize traffic congestion.
- Throughout the thesis, all validation was carried out on a fixed frame rate of 30 frames/second and the image resolution was considered to be 720×1280 for the maximum speed of 25 m/s, which is equivalent to 90 km/hr., i.e. the speed limit of expressways in Singapore. It would be interesting to study the computational savings that could result from lowering the frame rate at lower vehicle speeds. For

8. Conclusions and Future Research

example, when the vehicles navigate along roads that are not expressways where the typical speed limit is 70 km/hr., the algorithm can adapt to use a lower frame rate, thereby resulting in increased computational savings.

Appendix A

List of road markings in Singapore and their significances

Table A.1: Road Markings Across the Road and their Significance

Name of Marking	Illustration	Significance
Single Broken White Line		A broken white line drawn in the center of a single-lane two-way road means that vehicles should keep to the left of this line. Motorists can overtake, park or make a U-turn.
Single Continuous White Line		A continuous white line drawn in the center of a single-lane two-way road means that vehicles should keep to the left of this line. Parking is not allowed on either side of the road. However, motorists can overtake, park or make a U-turn.
Continued on next page		

A. List of road markings in Singapore and their significances

Table A.1 – continued from previous page

Name of Marking	Illustration	Significance
Double Continuous White Lines		A pair of parallel continuous white lines drawn in the center of a single-lane two-way road indicates that parking is not allowed on either side of the road. Vehicles are also not allowed to cross these lines.
Single Continuous Yellow Line		A continuous yellow line on the edge of a road indicates no parking between 7.00am and 7.00pm on that side of the road where the line is drawn, except on Sundays and public holidays.
Double Continuous Yellow Lines		A pair of parallel yellow lines indicate no parking at all times on that side of the road where these lines are drawn, except for the immediate picking up and letting down of passengers
Single Zigzag White Line		Zigzag lines by the side of the road give advance warning of the presence of a Pedestrian Crossing. Vehicles should not overtake, wait or park in the vicinity. Pedestrians also prohibited from crossing at the zigzag areas.
Continued on next page		

Table A.1 – continued from previous page

Name of Marking	Illustration	Significance
Single Zigzag Yellow Line		A single yellow zigzag line indicates no parking at all times on that side of the road where the line is drawn, except for the immediate picking up and letting down of passengers. Violation of this rule attracts demerit points and a fine.
Double Zigzag Yellow Lines		A pair of parallel yellow zigzag lines indicate no stopping at all times on that side of the road where the line is drawn. Violation of this rule attracts demerit points and a fine.
Single Broken White Line		A single broken white line drawn across the road indicates that traffic approaching this line must give away to traffic on a major road.
Double Broken White Line		A pair of parallel broken white lines drawn across the road indicates that traffic approaching these lines must give away to traffic on a major road.
Continued on next page		

A. List of road markings in Singapore and their significances

Table A.1 – continued from previous page

Name of Marking	Illustration	Significance
Single Continuous White Line		A single continuous white line drawn across the road indicates that traffic is required to stop and must stop close to and before this line.
Zebra Crossing		A zebra crossing drawn across the road indicates that one must slow down and stop if there are pedestrians crossing or about to cross the road.

References

- [1] TOMTOM. **Tomtom - Advanced lane guidance**. <http://www.tomtom.com/page/lane-guidance>. [Online; accessed 01-Dec-2012]. xi, 139, 140
- [2] JOHN WILEY AND SONS. **Advanced Driver Assistance Systems (ADAS): Vehicle Control for the Future**. *IMechE Seminar Publications*. 11
- [3] KLAUS KOMPASS AND WERNER HUBER. **Advanced Driver Assistance - How Far Should they go?**. http://www.autofocusasia.com/knowledge_bank/articles/advanced_driver_assistance.htm. [Online; accessed 01-Dec-2012]. 11
- [4] S. SUCHITRA AND R.K. SATZODA. **Datasets for testing by author**. <https://docs.google.com/folder/d/0B6eAY208msWGRm5m0DBXckVPTXc/edit>. [2012]. 5
- [5] FROST AND SULLIVAN. **Automotive and Transport Research and Consulting**. <http://www.automotive.frost.com>. 11, 16, 32
- [6] REUTERS. **Research and Markets: The Market for Advanced DAS (ADAS) Advanced Driver Assistance Systems Is Estimated To Reach 143 Million Euros By 2015**. http://www.researchandmarkets.com/research/66581b/strategic_analysis. 12
- [7] B. DE SCHUTTER O. GIETELINK, J. PLOEG AND M. VERHAEGEN. **Development of advanced driver assistance systems with vehicle hardware-in-the-loop simulations**. In *Vehicle System Dynamics*, 44, page 569590, july 2006. 12

REFERENCES

- [8] BRUNO EMILE STEPHANE BEGOT VINCENT VOISIN, MANUEL AVILA AND JEAN-CHRISTOPHE BARDET. **Road Markings Detection and Tracking Using Hough Transform and Kalman Filter.** *Proceedings of the Advanced Concepts for Intelligent Vision Systems.* **44**, page 76-83, July 2005. 12, 20, 21, 26, 27, 34
- [9] P. FOUCHER, Y. SEBSADJI, J.-P. TAREL, P. CHARBONNIER, AND P. NICOLLE. **Detection and recognition of urban road markings using images.** In *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, pages 1747 –1752, oct. 2011. 12, 13, 16, 20, 24, 25, 28, 29, 33, 139
- [10] HEALTH AND SAFETY EXECUTIVE. **Road Markings.** <http://www.hse.gov.uk/pubns/wpt27.pdf>, 2011. [Online; accessed 25-Oct-2012]. 12, 13
- [11] SINGAPORE TRAFFIC POLICE. *Basic Theory of Driving - The Official Handbook.* HB Media Holdings Pte Ltd, eighth edition, 2012. 12, 14
- [12] ROAD SAFETY MARKINGS ASSOCIATION. **White Lines Saves Lives.** <http://www.rsma.co.uk/index.php/resources/public-documents.html>, 2011. [Online; accessed 25-Oct-2012]. 12
- [13] ITS INTERNATIONAL. **Study proves road safety benefits of wider edge markings.** <http://www.itsinternational.com/event-news/intertraffic-2012/study-proves-road-safety-benefits-of-wider-edge-markings/>, 2012. [Online; accessed 25-Oct-2012]. 12
- [14] Y. OTSUKA, S. MURAMATSU, H. TAKENAGA, Y. KOBAYASHI, AND T. MONJ. **Multitype lane markers recognition using local edge direction.** In *Intelligent Vehicle Symposium, 2002. IEEE*, **2**, pages 604 – 609, june 2002. 12, 17, 18, 19, 33, 46
- [15] G. MASTORAKIS AND E.R. DAVIES. **Improved line detection algorithm for locating road lane markings.** *Electronics Letters*, **47**(3):183 –184, 3 2011. 12, 15, 17, 20, 26
- [16] R.K. SATZODA, S. SUCHITRA, AND T. SRIKANTHAN. **Robust extraction of lane markings using gradient angle histograms and directional signed**

-
- edges. In *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pages 754 –759, june 2012. 12, 15, 17, 18, 19, 20, 21, 33
- [17] TAO WU AND A. RANGANATHAN. **A practical system for road marking detection and recognition.** In *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pages 25 –30, june 2012. 12, 13, 15, 16, 17, 28, 29, 30, 31, 33, 34, 139, 143
- [18] X. AN, M. WU, AND H. HE. **A Novel Approach to Provide Lane Departure Warning Using Only One Forward Looking Camera.** *International Symposium on Collaborative Technologies and Systems*, pages 356 – 362, May 2006. 12, 15, 17
- [19] SIO-SONG IENG, JEAN-PHILIPPE TAREL, AND R. LABAYRADE. **On the design of a single lane-markings detectors regardless the on-board camera’s position.** In *Intelligent Vehicles Symposium, 2003. Proceedings. IEEE*, pages 564 – 569, june 2003. 12, 17
- [20] V. POPESCU, R. DANESCU, AND S. NEDEVSCHI. **On-road position estimation by probabilistic integration of visual cues.** In *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pages 583 –589, june 2012. 12, 13, 16, 27, 33
- [21] R. SCHUBERT, K. SCHULZE, AND G. WANIELIK. **Situation Assessment for Automatic Lane-Change Maneuvers.** *Intelligent Transportation Systems, IEEE Transactions on*, **11**(3):607 –616, sept. 2010. 12, 27, 32, 33
- [22] J.M. COLLADO, C. HILARIO, A. DE LA ESCALERA, AND J.M. ARMINGOL. **Detection and classification of road lanes with a frequency analysis.** In *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE*, pages 78 – 83, june 2005. 12, 16, 27
- [23] NAN WANG, WEI LIU, CHUNMIN ZHANG, HUAI YUAN, AND JIREN LIU. **The detection and recognition of arrow markings recognition based on monocular vision.** In *Control and Decision Conference, 2009. CCDC ’09. Chinese*, pages 4380 –4386, june 2009. 12, 13, 15, 16, 20, 28, 29, 33, 139, 157, 159

REFERENCES

- [24] G. MAIER, S. PANGERL, AND A. SCHINDLER. **Real-time detection and classification of arrow markings using curve-based prototype fitting.** In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 442 –447, june 2011. 12, 13, 28, 29, 33, 139
- [25] V. ANDREY AND KANG-HYUN JO. **Road Guidance Sign Recognition in Urban Areas by Structure.** In *Strategic Technology, The 1st International Forum on*, pages 293 –296, oct. 2006. 12, 13
- [26] C. BAGHDASSARIAN, H. LANGE, H. SAHLI, AND C. LAURGEAU. **Recognition of arrows in the environment of road markings.** In *Intelligent Vehicles '94 Symposium, Proceedings of the*, pages 219 – 224, oct. 1994. 12, 13
- [27] R. DANESCU AND S. NEDEVSCHI. **Detection and classification of painted road objects for intersection assistance applications.** In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pages 433 –438, sept. 2010. 12, 14, 16
- [28] S. SE. **Zebra-crossing detection for the partially sighted.** In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, **2**, pages 211 –217 vol.2, 2000. 12, 13, 14, 15, 16, 31
- [29] M.S. UDDIN AND T. SHIOYAMA. **Detection of pedestrian crossing and measurement of crossing length - an image-based navigational aid for blind people.** In *Intelligent Transportation Systems, 2005. Proceedings. 2005 IEEE*, pages 331 – 336, sept 2005. 12, 13, 14, 16, 31, 32
- [30] K.C. KLUGE. **Performance evaluation of vision-based lane sensing: some preliminary tools, metrics, and results.** In *Intelligent Transportation System, 1997. ITSC '97., IEEE Conference on*, pages 723 –728, nov 1997. 12, 32
- [31] JOON WOONG LEE. **A Machine Vision System for Lane-Departure Detection.** In *Computer Vision and Image Understanding*, **86**, pages 52–78, april 2002. 12, 17, 18, 19, 32, 33, 46
- [32] J.C. MCCALL AND M.M. TRIVEDI. **Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation.** *Intelligent*

-
- Transportation Systems, IEEE Transactions on*, **7**(1):20–37, march 2006. 12, 15, 16, 17, 18, 20, 26, 27, 28, 32, 33, 39
- [33] B.A. DRAPER S.D. BULUSWAR. **Color machine vision for autonomous vehicles.** *Engineering Applications of Artificial Intelligence*, **11**. 12, 32
- [34] J.C. MCCALL AND M.M. TRIVEDI. **Visual context capture and analysis for driver attention monitoring.** In *Intelligent Transportation Systems, 2004. Proceedings. The 7th International IEEE Conference on*, pages 332 – 337, oct. 2004. 12, 17, 32
- [35] DONGWOOK KIM, SEUNGWUK MOON, JAEMANN PARK, H.J. KIM, AND KYONGSU YI. **Design of an Adaptive Cruise Control / Collision Avoidance with lane change support for vehicle autonomous driving.** In *ICCAS-SICE, 2009*, pages 2938 –2943, aug. 2009. 12, 17, 32
- [36] A.BROGGI M.BERTOZZI. **GOLD: a parallel real-time stereo vision system for generic obstacle and lane detection.** *IEEE Transaction Image Processing*, **7**(1):62 – 81, 1998. 12, 16, 24, 25, 32
- [37] S. HABENICHT, H. WINNER, S. BONE, F. SASSE, AND P. KORZENIETZ. **A maneuver-based lane change assistance system.** In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 375 –380, june 2011. 12, 32
- [38] B. MORRIS, A. DOSHI, AND M. TRIVEDI. **Lane change intent prediction for driver assistance: On-road design and evaluation.** In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 895 –901, june 2011. 12, 32
- [39] J. DIAZ ALONSO, E. ROS VIDAL, A. ROTTER, AND M. MUHLENBERG. **Lane-Change Decision Aid System Based on Motion-Driven Vehicle Tracking.** *Vehicular Technology, IEEE Transactions on*, **57**(5):2736 –2746, sept. 2008. 12, 32
- [40] I.M. CHIRA, A. CHIBULCUTEAN, AND R.G. DANESCU. **Real-time detection of road markings for driving assistance applications.** In *Computer Engineering and Systems (ICCES), 2010 International Conference on*, pages 158 –163, 30 2010-dec. 2 2010. 12, 20, 24, 25, 27, 28, 30, 34, 143

REFERENCES

- [41] K. TOU, T. IRIE, J.K. TAN, AND S. ISHIKAWA. **Producing a Three-dimensional Road Map for Car Navigation.** In *SICE-ICASE, 2006. International Joint Conference*, pages 4776 –4779, oct. 2006. 13, 33, 139
- [42] S. VACEK, C. SCHIMMEL, AND R. DILLMANN. **Road-marking Analysis for Autonomous Vehicle Guidance.** In *European Conference on Mobile Robotics, 2007. Proceedings.*, pages 1–6, 2007. 13, 27, 28, 29, 30, 33, 34, 143
- [43] ARIZONA DEPARTMENT OF TRANSPORTATION. **Pedestrian Crosswalks - How Safe Are They?** <http://www.azdot.gov/Highways/traffic/XWalk.asp>. [Online; accessed 27-Oct-2012]. 13
- [44] M.S. UDDIN AND T. SHIOYAMA. **Detection of pedestrian crossing using bipolarity feature-an image-based technique.** *Intelligent Transportation Systems, IEEE Transactions on*, **6**(4):439 – 445, dec. 2005. 14, 31, 32
- [45] KPMG CENTER FOR AUTOMOTIVE RESEARCH. **Self-driving cars: The next revolution.** <https://www.kpmg.com/US/en/IssuesAndInsights/ArticlesPublications/Documents/self-driving-cars-next-revolution.pdf>, 2012. [Online; accessed 28-Oct-2012]. 14
- [46] Z. SUN, G. BEBIS, AND R. MILLER. **On-road Vehicle Detection: A Review.** *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **28**(5):694 – 711, May 2006. 14, 15, 16, 32
- [47] XUN DAI, A. KUMMERT, SU BIRM PARK, AND U. IURGEL. **Vehicle centroid estimation based on radar multiple detections.** In *Vehicular Electronics and Safety, 2007. ICVES. IEEE International Conference on*, pages 1 –5, dec. 2007. 14
- [48] GIANCARLO ALESSANDRETTI, ALBERTO BROGGI, AND PIETRO CERRI. **Vehicle and Guard Rail Detection Using Radar and Vision Data Fusion.** *Intelligent Transportation Systems, IEEE Transactions on*, **8**(1):95 –105, march 2007. 14

-
- [49] JIANXIN FANG, HUADONG MENG, HAO ZHANG, AND XIQIN WANG. **A Low-cost Vehicle Detection and Classification System based on Unmodulated Continuous-wave Radar.** In *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*, pages 715 –720, 30 2007-oct. 3 2007. 14
- [50] XIN LIU, ZHENPING SUN, AND HANGEN HE. **On-road vehicle detection fusing radar and vision.** In *Vehicular Electronics and Safety (ICVES), 2011 IEEE International Conference on*, pages 150 –154, july 2011. 14
- [51] 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 *Intelligent Transportation Systems, 2005. Proceedings. 2005 IEEE*, pages 552 – 556, sept. 2005. 14
- [52] LILI HUANG AND M. BARTH. **Tightly-coupled LIDAR and computer vision integration for vehicle detection.** In *Intelligent Vehicles Symposium, 2009 IEEE*, pages 604 –609, june 2009. 14, 15
- [53] M. MAHLISCH, R. SCHWEIGER, W. RITTER, AND K. DIETMAYER. **Sensor-fusion Using Spatio-Temporal Aligned Video and Lidar for Improved Vehicle Detection.** In *Intelligent Vehicles Symposium, 2006 IEEE*, pages 424 –429, 0-0 2006. 14, 15
- [54] C. PREMEBIDA, G. MONTEIRO, U. NUNES, AND P. PEIXOTO. **A Lidar and Vision-based Approach for Pedestrian and Vehicle Detection and Tracking.** In *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*, pages 1044 –1049, 30 2007-oct. 3 2007. 14, 15
- [55] GUANG LU AND MASAYOSHI TOMIZUKA. **LIDAR Sensing for Vehicle Lateral Guidance: Algorithm and Experimental Study.** *Mechatronics, IEEE/ASME Transactions on*, **11**(6):653 –660, dec. 2006. 14, 15
- [56] K. TAKAGI, K. MORIKAWA, T. OGAWA, AND M. SABURI. **Road Environment Recognition Using On-vehicle LIDAR.** In *Intelligent Vehicles Symposium, 2006 IEEE*, pages 120 –125, 0-0 2006. 14, 15

REFERENCES

- [57] G. QIAN R. CHELLAPPA AND Q. ZHENG. **Vehicle Detection and Tracking Using Acoustic and Video Sensors.** pages 793–796, 2004. 14
- [58] B. STEUX, C. LAURGEAU, L. SALESSE, AND D. WAUTIER. **Fade: a vehicle detection and tracking system featuring monocular color vision and radar data fusion.** In *Intelligent Vehicle Symposium, 2002. IEEE*, **2**, pages 632 – 639 vol.2, june 2002. 14
- [59] F. GARCIA, P. CERRI, A. BROGGI, A. DE LA ESCALERA, AND J.M. ARMINGOL. **Data fusion for overtaking vehicle detection based on radar and optical flow.** In *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pages 494 –499, june 2012. 14
- [60] J. GUNNARSSON, L. SVENSSON, L. DANIELSSON, AND F. BENGTSSON. **Tracking vehicles using radar detections.** In *Intelligent Vehicles Symposium, 2007 IEEE*, pages 296 –302, june 2007. 14
- [61] H. WEIGEL, P. LINDNER, AND G. WANIELIK. **Vehicle tracking with lane assignment by camera and lidar sensor fusion.** In *Intelligent Vehicles Symposium, 2009 IEEE*, pages 513 –520, june 2009. 15
- [62] C. ADAM, R. SCHUBERT, N. MATTERN, AND G. WANIELIK. **Probabilistic road estimation and lane association using radar detections.** In *Information Fusion (FUSION), 2011 Proceedings of the 14th International Conference on*, pages 1 –8, july 2011. 15
- [63] P. LINDNER, E. RICHTER, G. WANIELIK, K. TAKAGI, AND A. ISOGAI. **Multi-channel lidar processing for lane detection and estimation.** In *Intelligent Transportation Systems, 2009. ITSC '09. 12th International IEEE Conference on*, pages 1–6, oct. 2009. 15
- [64] B. MA, S. LAKSHMANAN, AND III HERO, A.O. **Simultaneous detection of lane and pavement boundaries using model-based multisensor fusion.** *Intelligent Transportation Systems, IEEE Transactions on*, **1(3)**:135 –147, sep 2000. 15

-
- [65] DAN LEVI AHARON BAR HILLEL, RONEN LERNER AND GUY RAZ. **Recent progress in road and lane detection: a survey.** In *Machine Vision and Applications, Springer-Verlag*, pages 1-19, feb 2012. 15, 16, 17, 20, 32
- [66] SHENGYAN ZHOU, YANHUA JIANG, JUNQIANG XI, JIANWEI GONG, GUANGMING XIONG, AND HUIYAN CHEN. **A novel lane detection based on geometrical model and Gabor filter.** In *Intelligent Vehicles Symposium (IV), 2010 IEEE*, pages 59 –64, june 2010. 15, 17, 20, 22, 46
- [67] CHENGQUN QIU. **An edge detection method of lane lines based on mathematical morphology and MATLAB.** In *Cross Strait Quad-Regional Radio Science and Wireless Technology Conference (CSQRWC), 2011, 2*, pages 1266 –1269, july 2011. 15
- [68] DONG JUNG KANG, JANG WON CHOI, AND IN SO KWEON. **Finding and tracking road lanes using line-snakes.** In *Intelligent Vehicles Symposium, 1996., Proceedings of the 1996 IEEE*, pages 189 –194, sep 1996. 15, 23, 24
- [69] S. NEDEVSKI, R. SCHMIDT, T. GRAF, R. DANESCU, D. FRENTIU, T. MARITA, F. ONIGA, AND C. POCOL. **3D lane detection system based on stereovision.** In *Intelligent Transportation Systems, 2004. Proceedings. The 7th International IEEE Conference on*, pages 161 – 166, oct. 2004. 15
- [70] CHUNZHAO GUO, S. MITA, AND D. MCALLESTER. **Lane detection and tracking in challenging environments based on a weighted graph and integrated cues.** In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 5543 –5550, oct 2010. 15, 17, 18, 24, 25, 26, 27, 33
- [71] S.A. SAMAD M.S. JAVADI, M.A. HANNAN AND A. HUSSAIN. **A Robust Vision-based Lane Boundaries Detection Approach for Intelligent Vehicles.** In *Information Technology Journal*, pages 1184–1192, june 2012. 15
- [72] MOHD HANAFI AHMAD HIJAZI JUMAT SULAIMAN AZALI SAUDI, JASON TEO. **Fast Lane Detection with Randomized Hough Transform.** *International Symposium on Information Technology*, 4. 15, 20, 21, 34

REFERENCES

- [73] MAKOTO MANABE TAKESHI TAOKA AND MASAHIRO FUKUI. **An Efficient Curvature Lane Recognition Algorithm by Piecewise Linear Approach.** *Proc. of IEEE 65th Vehicular Technology Conference VTC2007-Spring*, pages 2530–2534, 2007. 15, 20, 22, 23, 46
- [74] T. GAVRILOVIC, J. NINOT, AND L. SMADJA. **Frequency filtering and connected components characterization for zebra-crossing and hatched markings detection.** page A:43, 2010. 16, 27, 31, 33
- [75] HUGUES MOUNIER MOEZ AMMAR, SYLVIE LE HGARAT-MASCLE. **Road Surface Marking Classification Based on a Hierarchical Markov Model.** In *8th International Conference, ICIAR 2011*, pages 348–359, june 2011. 16, 31, 33
- [76] Y. SEBSADJI, J.-P. TAREL, P. FOUCHER, AND P. CHARBONNIER. **Robust road marking extraction in urban environments using stereo images.** In *Intelligent Vehicles Symposium (IV), 2010 IEEE*, pages 394 –400, june 2010. 16, 33
- [77] JOON WOONG LEE. **A lane-departure identification based on LBPE, Hough transform, and linear regression.** In *Computer Vision and Image Understanding*, **99**, page 359383, april 2005. 17, 20
- [78] M. KUTILA, M. JOKELA, G. MARKKULA, AND M.R. RUE. **Driver Distraction Detection with a Camera Vision System.** In *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, **6**, pages 201–204, 16 2007-oct. 19 2007. 17
- [79] F. FRIEDRICHS, M. MIKSCH, AND BIN YANG. **Estimation of lane data-based features by odometric vehicle data for driver state monitoring.** In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pages 611 –616, sept. 2010. 17
- [80] F.A. SIDDIQUI, S. AMIR, M. ASIF, AND Z.A. ALI. **Lane tracking and autonomous cruise control for automatic highway system.** In *Signal Processing and Communications Applications (SIU), 2011 IEEE 19th Conference on*, pages 542 –545, april 2011. 17

-
- [81] S. HOLD, S. GO ANDRMER, A. KUMMERT, M. MEUTER, AND S. MU ANDLLER SCHNEIDERS. **ELA - an exit lane assistant for adaptive cruise control and navigation systems.** In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pages 629–634, sept. 2010. 17
- [82] HANENE BEN-ABDALLAH NADRA BEN ROMDHANE, MOHAMED HAMMAMI. **A Comparative Study of Vision-Based Lane Detection Methods.** In *Advanced Concepts for Intelligent Vision Systems, Lecture Notes in Computer Science*, **6915**, pages 46–57, 2011. 17, 20, 21, 23, 24
- [83] P. LINDNER, S. BLOKZYL, G. WANIELIK, AND U. SCHEUNERT. **Applying multi level processing for robust geometric lane feature extraction.** In *Multisensor Fusion and Integration for Intelligent Systems (MFI), 2010 IEEE Conference on*, pages 248–254, sept. 2010. 17, 19, 34
- [84] R.K. SATZODA, S. SATHYANARAYANA, T. SRIKANTHAN, AND S. SATHYANARAYANA. **Hierarchical Additive Hough Transform for Lane Detection.** *Embedded Systems Letters, IEEE*, **2**(2):23–26, june 2010. 17, 18, 20, 21
- [85] PEI-YUNG HSIAO, CHUN-WEI YEH, SHIH-SHINH HUANG, AND LI-CHEN FU. **A Portable Vision-Based Real-Time Lane Departure Warning System: Day and Night.** *Vehicular Technology, IEEE Transactions on*, **58**(4):2089–2094, may 2009. 17, 18
- [86] WEN HE, XIAODONG WANG, GUISHENG CHEN, MU GUO, TIANLEI ZHANG, PENG HAN, AND RUIJIAN ZHANG. **Monocular based lane-change on scaled-down autonomous vehicles.** In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 144–149, june 2011. 17, 20, 21, 24, 25, 34
- [87] XIN LIU, BIN DAI, JINZE SONG, HANGEN HE, AND BO ZHANG. **Real-Time Long-Range Lane Detection and Tracking for Intelligent Vehicle.** In *Image and Graphics (ICIG), 2011 Sixth International Conference on*, pages 654–659, aug. 2011. 17, 18
- [88] M. MEUTER, S. MULLER-SCHNEIDERS, A. MIKA, S. HOLD, C. NUNN, AND A. KUMMERT. **A novel approach to lane detection and tracking.** In

REFERENCES

- Intelligent Transportation Systems, 2009. ITSC '09. 12th International IEEE Conference on*, pages 1–6, oct. 2009. 17, 18, 22, 23, 26, 27
- [89] B. FARDE, U. SCHEUNERT, H. CRAMER, AND G. WANIELIK. **A new approach for lane departure identification.** In *Intelligent Vehicles Symposium, 2003. Proceedings. IEEE*, pages 100–105, june 2003. 17, 18, 20, 21
- [90] C. KREUCHER AND S. LAKSHMANAN. **A frequency domain approach to lane detection in roadway images.** In *Image Processing, 1999. ICIP 99. Proceedings. 1999 International Conference on*, **2**, pages 31–35, 1999. 17, 19
- [91] C. KREUCHER AND S. LAKSHMANAN. **LANA: a lane extraction algorithm that uses frequency domain features.** *Robotics and Automation, IEEE Transactions on*, **15**(2):343–350, apr 1999. 17, 19
- [92] K. KLUGE AND S. LAKSHMANAN. **A deformable-template approach to lane detection.** In *Intelligent Vehicles '95 Symposium., Proceedings of the*, pages 54–59, sep 1995. 17, 19, 23, 24
- [93] F. COSKUN, O. TUNER, M.E. KARSLIGIL, AND L. GUVENC. **Real time lane detection and tracking system evaluated in a hardware-in-the-loop simulator.** In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pages 1336–1343, sept. 2010. 17, 20, 23, 24, 25
- [94] LUO-WEI TSAI, JUN-WEI HSIEH, CHI-HUNG CHUANG, AND KUO-CHIN FAN. **Lane detection using directional random walks.** In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 303–306, june 2008. 17, 20, 23
- [95] SHIH-SHINH HUANG, CHUNG-JEN CHEN, PEI-YUNG HSIAO, AND LI-CHEN FU. **On-board vision system for lane recognition and front-vehicle detection to enhance driver's awareness.** In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, **3**, pages 2456–2461, april-1 may 2004. 17, 20
- [96] M.J.T. REINDERS B. J. LEI, EMILE A. HENDRIKS. **On Feature Extraction from Images.** In *Technical Report, Information and Communication Theory Group, TUDelft*, pages 1–57, jun 1999. 18

-
- [97] R. C. GONZALEZ AND R. E. WOODS. *Digital Image Processing*. Prentice Hall, second edition, 2002. 18, 19, 41, 146
- [98] PANGYU JEONG AND S. NEDEVSCHI. **Efficient and robust classification method using combined feature vector for lane detection**. *Circuits and Systems for Video Technology, IEEE Transactions on*, **15**(4):528 – 537, april 2005. 18
- [99] A. BORKAR, M. HAYES, AND M.T. SMITH. **Polar randomized hough transform for lane detection using loose constraints of parallel lines**. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 1037 –1040, may 2011. 20, 21, 34
- [100] J. P. LEWIS. **Fast Template Matching**. *Vision Interface*, pages 120 – 123, 1995. 20, 21, 30, 34
- [101] A. JAIN B. YU. **Lane Boundary Detection Using a Multiresolution Hough Transform**. *International Conference on Image Processing (ICIP97)*, **2**. 20
- [102] NADRA BEN ROMDHANE, MOHAMED HAMMAMI, AND HANENE BEN-ABDALLAH. **An Artificial Vision-Based Lane Detection Algorithm**. In *Computer Science and its Applications, 2009. CSA '09. 2nd International Conference on*, pages 1 –6, dec. 2009. 20, 21, 22, 23, 34, 46
- [103] ZUWHAN KIM. **Robust Lane Detection and Tracking in Challenging Scenarios**. *Intelligent Transportation Systems, IEEE Transactions on*, **9**(1):16 –26, march 2008. 20, 26, 33
- [104] DINGGANG SHEN YUE WANG, EAM KHWANG TEOH. **Lane detection and tracking using B-Snake**. In *Image and Vision Computing*, **22**, page 269280, oct 2004. 20, 23, 24
- [105] D. TERZOPOULOS M. KASS, A. WITKIN. **Snakes: active contour models**. In *Internatuional Journal of Computer Vision*, **1**, page 321331, oct 1987. 20, 23

REFERENCES

- [106] WENHONG ZHU CHAO XU MIN TIAN, FUQIANG LIU. **Vision Based Lane detection for Active Security in Intelligent Vehicle.** *Proc. of IEEE International Conference on Vehicular Electronics and Safety.* 20, 22
- [107] JOHN B. McDONALD, JOHN B. MC DONALD, ROBERT SHORTEN, AND JOCHEN FRANZ. **Application of the Hough Transform to Lane Detection in Motorway Driving Scenarios,** 2001. 20, 21
- [108] R. O. DUDA AND P. E. HART. **Use of the Hough transform to detect lines and curves in pictures.** *ACM Comm.*, **15**(1):11 – 15, 1972. 21, 48
- [109] CHIH-HSIEN YEH AND YUNG-HSIN CHEN. **Development of vision-based lane and vehicle detecting systems via the implementation with a dual-core DSP.** In *Intelligent Transportation Systems Conference, 2006. ITSC '06. IEEE*, pages 1179 –1184, sept. 2006. 21
- [110] T. KASAI AND K. ONOGUCHI. **Lane Detection System for Vehicle Platooning using Multi-information Map.** In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pages 1350 –1356, sept. 2010. 21
- [111] L. XU AND E. OJA. **Randomized Hough transform (RHT) - basic mechanism, algorithms and computational complexities.** *CVGIP - image understanding*, **57**(2):131–154, 1993. 21, 22, 34, 90
- [112] G. LOY AND A. ZELINSKY. **Fast radial symmetry for detecting points of interest.** *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **25**(8):959 – 973, aug. 2003. 22
- [113] D. H. BALLARD AND C. M. BROWN. *Computer Vision.* Prentice Hall, 1982. 22, 30, 34, 144
- [114] MING-YANG CHERN AND YI-HSIANG LU. **Design and integration of parallel Hough-transform chips for high-speed line detection.** *Proceedings of 11th International Conference on Parallel and Distributed Systems*, **2**:42 – 46, July 2005. 22, 34, 91, 92, 93

-
- [115] R. K. SATZODA, S. SUCHITRA, AND T. SRIKANTHAN. **Parallelizing Hough transform computation.** *IEEE Signal Processing Letters*, **15**(1):297 – 300, 2008. 22, 34
- [116] S. SUCHITRA, R. K. SATZODA, AND T. SRIKANTHAN. **Exploiting inherent parallelisms for accelerating linear Hough transform computation.** *IEEE Transactions on Image Processing*, **18**(10):2255 – 2264, 2009. 93
- [117] R. CUCCHIARA, G. NERI, AND M. PICCARDI. **A real-time hardware implementation of the Hough transform.** *Journal of Systems Architectures*, **45**:31 – 45, 1998. 22, 34
- [118] F. O’GORMAN AND M. B. CLOWES. **Finding picture edges through collinearity of feature points.** *IEEE Transactions on Computers*, **C-25**(4):449 – 456, April 1976. 22, 23, 34, 90
- [119] N. KIRYATI, Y. ELDAR, AND A. M. BRUKSTEIN. **A probabilistic Hough transform.** *Pattern Recognition*, **24**(4):303 – 316, 1991. 22, 34, 90
- [120] C. GALAMHOS, J. MATAS, AND J. KITTLER. **Progressive probabilistic Hough transform for line detection.** *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1:554 – 560, June 1999. 22, 34, 90
- [121] C. GALAMBOS, J. KITTLER, AND J. MATAS. **Using gradient information to enhance the progressive probabilistic Hough transform.** In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, **3**, pages 560 – 563 vol.3, 2000. 22, 23, 34, 90
- [122] D. WOLF A. KASKE, R. HUSSON. **Chi-square fitting of deformable templates for lane boundary detection.** In *IAR Annual Meeting 95, Grenoble France*, nov 1995. 23
- [123] YIFEI WANG, N. DAHNOUN, AND A. ACHIM. **A novel lane feature extraction algorithm implemented on the TMS320DM6437 DSP platform.** In *Digital Signal Processing, 2009 16th International Conference on*, pages 1 – 6, july 2009. 24, 34

REFERENCES

- [124] O. GALLO AND R. MANDUCHI. **Reading 1D Barcodes with Mobile Phones Using Deformable Templates.** *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **33**(9):1834 –1843, sept 2011. 24, 34
- [125] J. J. LITTLE H. A. MALLOT, H. H. BLTHOFF AND S. BOHRER. **Inverse perspective mapping simplifies optical flow computation and obstacle detection.** In *Biological Cybernetics*, **64**, pages 177–185, 1991. 24, 25, 30
- [126] UNIVERSITY OF PARMA ARGO AUTONOMOUS VEHICLE PROJECT. **The GOLD System.** <http://www.argo.ce.unipr.it/argo/english/techdet/ipm.html>. [Online; accessed 06-Nov-2012]. 24
- [127] MARCOS NIETO JON ARRSPIDE, LUIS SALGADO AND FERNANDO JAU-REGUIZAR. **Real-time vehicle detection and tracking based on perspective and non-perspective space cooperation.** In *Real-Time Image and Video Processing, SPIE*, **7244**, pages 1–12, 2009. 25
- [128] MARTIN A. FISCHLER AND ROBERT C. BOLLES. **Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography.** In *Comm. of the ACM*, **24**, page 381395, 1981. 26
- [129] R. DANESCU AND S. NEDEVSCI. **Probabilistic Lane Tracking in Difficult Road Scenarios Using Stereovision.** *Intelligent Transportation Systems, IEEE Transactions on*, **10**(2):272 –282, june 2009. 26, 27, 28
- [130] W.T. FREEMAN AND E.H. ADELSON. **The design and use of steerable filters.** *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **13**(9):891 –906, sep 1991. 28
- [131] J. REBUT, A. BENSRAIR, AND G. TOULMINET. **Image segmentation and pattern recognition for road marking analysis.** In *Industrial Electronics, 2004 IEEE International Symposium on*, **1**, pages 727 – 732, may 2004. 28, 29
- [132] P. CHARBONNIER, F. DIEBOLT, Y. GUILLARD, AND F. PEYRET. **Road markings recognition using image processing.** In *Intelligent Transportation System, 1997. ITSC '97., IEEE Conference on*, pages 912 –917, nov 1997. 29

-
- [133] R.O. DUDA AND P.E. HART. *Pattern Classification and Scene Analysis*. Wiley-Interscience publication. John Wiley & Sons, 1973. 30
- [134] DAVID A. FORSYTH AND JEAN PONCE. *Computer Vision, a modern approach*. Prentice Hall, 2003. 26, 34
- [135] F. COLLETO, M. MARCON, A. SARTI, AND S. TUBARO. **A Robust Method for the Estimation of Reliable Wide Baseline Correspondences**. In *Image Processing, 2006 IEEE International Conference on*, pages 1041 –1044, oct. 2006. 30
- [136] E. ROSTEN, R. PORTER AND T. DRUMMOND. **Faster and Better: A Machine Learning Approach to Corner Detection**. In *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **1**, pages 105 –119, jan 2010. 30
- [137] N. DALAL AND B. TRIGGS. **Histograms of oriented gradients for human detection**. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, **1**, pages 886 –893, june 2005. 30
- [138] G.J. VANDERBRUG AND A. ROSENFELD. **Two-Stage Template Matching**. *Computers, IEEE Transactions on*, **C-26**(4):384 –393, april 1977. 30, 34
- [139] GOSHTASBY AND ARDESHIR. **Template Matching in Rotated Images**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **7**(3):338 – 344, May 1985. 30, 34
- [140] W. KRATTENTHALER, K.J. MAYER, AND M. ZEILLER. **Point correlation: a reduced-cost template matching technique**. In *Image Processing, 1994. Proceedings. ICIP-94., IEEE International Conference*, **1**, pages 208 –212 vol.1, nov 1994. 30, 34
- [141] M. UENOHARA AND T. KANADE. **Use of Fourier and Karhunen-Loeve decomposition for fast pattern matching with a large set of templates**. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **19**(8):891 –898, aug 1997. 30, 34
- [142] S. OMACHI AND M. OMACHI. **Fast Template Matching With Polynomials**. *Image Processing, IEEE Transactions on*, **16**(8):2139 –2149, aug. 2007. 30, 34

REFERENCES

- [143] D.G. LOWE. **Object recognition from local scale-invariant features.** In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, **2**, pages 1150 –1157 vol.2, 1999. 30, 31, 35, 143
- [144] FENG ZHOU AND PETER KORNERUP. **A high-speed Hough transform Using CORDIC.** *Proceedings of International Conference of VLSI Digital Signal Processing*, pages 482–487, June 1995. 91, 92, 93
- [145] FENG ZHOU AND PETER KORNERUP. **A high speed Hough Transform using CORDIC.** Technical Report PP-1995-27, 1, 1995. 91, 92, 93
- [146] DIXON D. S. DENG AND HOSSAM ELGINDY. **High-speed parameterisable Hough Transform using reconfigurable hardware.** *Pan-Sydney Area Workshop on Visual Information Processing*, pages 51 – 57, May 2001. 91, 92, 93
- [147] J.H. PARK AND Y.S. KIM. **Decentralized variable structure control for active suspensions based on a full-car model.** In *Control Applications, 1998. Proceedings of the 1998 IEEE International Conference on*, **1**, pages 383 –387 vol.1, sep 1998. 130
- [148] M.R. STONE AND M.A. DEMETRIOU. **Modeling and simulation of vehicle ride and handling performance.** In *Intelligent Control, 2000. Proceedings of the 2000 IEEE International Symposium on*, pages 85 –90, 2000. 130
- [149] YI FU, H. LI, AND M. KAYE. **Design and stability analysis of a fuzzy controller for autonomous road following.** In *Intelligent Vehicles Symposium, 2009 IEEE*, pages 66 –71, june 2009. 130
- [150] BESTCOVERY. **Garmin nuvi 2460LMT GPS Review.** <http://www.bestcovery.com/garmin-nuvi-2460lmt-gps-vehicle-5-lcd>. [Online; accessed 01-Dec-2012]. 140
- [151] BESTCOVERY. **Best GPS.** <http://www.bestcovery.com/best-gps-overall#226830>. [Online; accessed 01-Dec-2012]. 140
- [152] C. PAPAGEORGIOU AND T. POGGIO. **A Trainable System for Object Detection.** In *International Journal of Computer Vision*, **38**, pages 15–33, 2000. 143

-
- [153] H. SCHNEIDERMAN AND T. KANADE. **A statistical method for 3D object detection applied to faces and cars.** In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, **1**, pages 746 –751 vol.1, 2000. 143
- [154] P. VIOLA AND M. JONES. **Rapid object detection using a boosted cascade of simple features.** In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, **1**, pages I-511 – I-518 vol.1, 2001. 143
- [155] BASTIAN LEIBE AND BERNT SCHIELE. **Scale-Invariant Object Categorization using a Scale-Adaptive Mean-Shift Search.** In *DAGM04 Pattern Recognition Symposium*, pages 145–153, 2004. 143
- [156] Y. FURUKAWA AND Y. SHINAGAWA. **Accurate and robust line segment extraction by analyzing distribution around peaks in hough space.** In *Computer Vision and Image Understanding*, **92**, pages I-1 – I-25, april 2003. x, xv, 100, 101, 111, 112, 119, 120, 121
- [157] M.-Y. CHERN AND Y.-H. LU. **Design and integration of parallel hough-transform chips for high-speed line detection.** In *Parallel and Distributed Systems, 2005. Proceedings. 11th International Conference on*, **2**, pages 42 –46 july 2005. 100
- [158] L. O’GORMAN AND S. A.C. **The converging squares algorithm: an efficient method for locating peaks in multidimensions.** In *PAMI, IEEE trans. on*, **6**, pages 280 –288, april 1984. 100
- [159] C.-T. CHOY, P.-K. SER, AND W.-C. SIU. **Peak detection in hough transform via self-organizing learning.** In *Circuits and Systems, 1995. ISCAS ’95., 1995 IEEE International Symposium on*, **1**, pages 139 –142, april 1995. 100
- [160] F. O GORMAN. AND M.B. CLOWES. **Finding Picture Edges Through Collinearity of Feature Points.** In *Computers, IEEE Transactions on*, 1976, **1**, pages I-449 – I-456 vol.C25, 1976. 100

REFERENCES

- [161] COMPUTATIONAL VISION AT CALTECH. **Caltech 101 database**. http://www.vision.caltech.edu/Image_Datasets/Caltech101/. [Online; accessed 25-Dec-2012]. 117
- [162] YIFAN HE AND ZIVKOVIC, Z. AND KLEIHORST, R. AND DANILIN, A. AND CORPORAAL, H.. **Real-time implementations of Hough Transform on SIMD architecture**. In *Distributed Smart Cameras, ACM/IEEE International Conference on, 1976*, **1**, pages I–8, 2008. 82
- [163] WANG, WEI AND HUANG, XINMING. **An FPGA co-processor for adaptive lane departure warning system**. In *Circuits and Systems (ISCAS), IEEE International Symposium on*, **1**, pages 1389–1383 vol.C25, 2013. 82

Declaration

I herewith declare that I have produced this paper without the prohibited assistance of third parties and without making use of aids other than those specified; notions taken over directly or indirectly from other sources have been identified as such. This paper has not previously been presented in identical or similar form to any other German or foreign examination board.

The thesis work was conducted from XXX to YYY under the supervision of PI at ZZZ.

CITY,