

Multi-Sensor Fusion Approach to Moving Object Detection and Tracking for Autonomous Driving

Submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Electrical and Computer Engineering

Hyunggi Cho

B.S., Mechanical Engineering, Sun Moon University

M.S., Electrical and Electronic Engineering, Yonsei University

M.S., Robotics, Carnegie Mellon University

Carnegie Mellon University

Pittsburgh, PA

July, 2014

I would like to dedicate this thesis to my loving parents ...

Acknowledgements

I am deeply grateful to Prof. Kumar for being a patient advisor and for constantly guiding me to right direction in my research. Without his insistence on analyzing things in the deepest level, his feedback on academic writing and, not least his providing of funding, this work would simply not have been possible. I would also like to thank Dr. Paul E. Rybski for being an advisor of my Master degree in robotics and introducing me to the CMU's great autonomous driving research team. Further, I would like to thank Prof. Raj Rajkumar for his endless passion for self-driving vehicles and his leadership for our SRX team. Finally, I thank Dr. Hernan Badino for his support and for providing wonderful feedback.

I really appreciate General Motors for providing funding over the six years of my graduate study at CMU. Not many PhD students could have this fantastic opportunity to consistently study one research problem (in my case, object detection) over their entire PhD period! I am also deeply thankful to colleagues in General Motors, Dr. Wende Zhang for his tremendous contribution to my bicycle detection and tracking project and Dr. Dan Levi for his amazing research ideas and in-depth feedback on the pedestrian detection project.

I am also thankful to every member of our SRX team. First of all, I want to thank Dr. John Dolan for his great leadership and Dr. Young-Woo Seo for his great comments on my multi-sensor tracking system. Secondly, I want to thank all my friends in our team, Jarrod Snider, Junsung Kim, Junqing Wei, JongHo Lee, Tianyu Gu, Wenda Xu and Alok Sharma. All the hard work we made together for the successful demos will be a good memory to all of us.

Finally, I save the last and greatest thanks for my lovely wife, Minjo. She has stood by me sacrificing many parts of her career developments. She is a wonderful mom to our kids and she is just a perfect woman to me. I would not be writing this if it were not for her support and love. Thank you!

Abstract

Autonomous vehicles, or self-driving vehicles, must be able to actively perceive and understand their immediate surroundings to operate safely in complex and dynamic traffic environments. However, correctly interpreting various sensor data and constructing a coherent world model for the traffic scene is a challenging task due to partial and noisy measurements from the sensors and the dynamic nature of the scene.

This thesis improves state-of-the-art moving object tracking with multiple sensors such as radars, LIDARs, and cameras for self-driving vehicles. It proposes improved approaches for vision-based object detection and multi-sensor object tracking. In addition, lane detection results are fused in the tracking system to exploit contextual interplay between lane markers and moving objects, especially moving vehicles.

Recognizing moving objects is essential to a perception system since the semantic information of object class can be utilized not only in a tracking system itself but also in a decision-making component. In this thesis, we thoroughly investigate the current state-of-the-art object detection methods and significantly improve the speed performance of one promising method called '*deformable part-based models*.' In addition, we improve pedestrian and vehicle detection accuracy by designing optimized object models for automotive applications. Finally, we achieve state-of-the-art pedestrian detection performance by adding motion features, that we refer to as '*inner motion features*.'

Furthermore, this thesis proposes a novel moving object detection and tracking system that uses improved motion and observation models for

active sensors (i.e., radars and LIDARs) and introduces a vision sensor. This cooperative fusion enables more accurate estimation of the kinematic properties (i.e., position and velocity) by radar and LIDAR sensors and new estimation of the geometric (i.e., size and volume) and semantic properties (i.e., object class) by cameras. Then, the semantic information of an object type is utilized for several internal sub-components of the tracking system.

Finally, we propose a holistic approach which leverages contextual cues to further improve the performance of our multi-sensor tracking system. The method exploits contextual interplay between moving objects and traffic environments such as lane markers and sidewalks.

All components proposed throughout this thesis were evaluated with challenging real-world data. Relevant sensor data were collected from 25 minutes of driving from the Carnegie Mellon's campus to the Pittsburgh's International Airport. Our experiments show that the holistic tracking approach, which integrates vision-based object detection, lane marker detection, and multi-sensor tracking, can offer a significant improvement over the conventional tracking system.

Contents

Contents	vi
List of Figures	x
List of Tables	xvi
1 Introduction	1
1.1 Motivation	2
1.2 Recent Progress on Autonomous Driving	2
1.3 Perception System for Self-Driving Vehicles	4
1.4 Automotive Sensors	5
1.5 Challenges for Moving Object Tracking	7
1.6 Thesis Statement	8
1.7 Summary of Contributions	9
1.8 Outline of the Dissertation	11
2 Related Work	14
2.1 Brief History of Autonomous Driving	14
2.2 Vision-Based Object Detection	15
2.3 Vision-Based Object Tracking	18
2.4 Multi-Sensor Fusion for Object Tracking	19
3 Vision-Based Object Detection with Deformable Part Models	21
3.1 Introduction	21
3.2 Real-Time Object Detection with DPM	23
3.2.1 Why DPM?	24

3.2.2	Review of the DPM and Star-Cascade Algorithm	24
3.2.3	Implementation Details	26
3.3	Motion-augmented DPM	29
3.3.1	Inner Motion	29
3.3.2	Weak Stabilization using Lucas-Kanade	30
3.3.3	Motion Feature Encoding	30
3.3.4	Integrating Motion into DPM	32
3.4	ROI Computation	32
3.4.1	Geometry	33
3.4.2	Formulation	34
3.5	Experiments	36
3.5.1	Model Design Parameters	37
3.5.2	Quantitative Evaluation with Caltech Benchmark	40
3.5.2.1	Pedestrian Detection	40
3.5.2.2	Vehicle Detection	44
3.5.3	Quantitative Evaluation of Performance with Inner Motion Features	47
3.5.4	Real-Time Onboard Evaluation	49
3.5.4.1	System Setup	50
3.5.4.2	Range Estimation	51
3.5.4.3	Detection Results	53
3.6	Summary	57
4	Vision-Based Multi-Object Tracking using a Rao-Blackwellized Particle Filter	58
4.1	Introduction	58
4.2	DPM as a Virtual Sensor	61
4.3	Single Object Tracking	62
4.3.1	Motion Models	62
4.3.2	Measurement Model	65
4.4	Extension to Multi-Object Tracking using RBMCDA Algorithm	67
4.4.1	Problem Statement	68
4.4.2	Particle Filter for Data Association	69

4.5	Experiments	71
4.5.1	Training DPMs for Urban Environments	72
4.5.2	Single Object Tracking Evaluation	73
4.5.3	Multi-Object Tracking Evaluation	74
4.6	Summary	76
5	Multi-Sensor Fusion for Moving Object Tracking	78
5.1	Introduction	79
5.2	Multi-Sensor Setup	80
5.3	Sensor Calibration	81
5.3.1	Synchronization	82
5.3.2	Camera and IBEO LUX Calibration	83
5.4	Sensor Characterization	84
5.4.1	Radar	84
5.4.2	LIDAR	84
5.4.3	Camera	86
5.5	Multi-Sensor Fusion	88
5.5.1	Sensor Fusion with Kalman filters	88
5.5.2	Tracking Models	90
5.5.3	Data Association	93
5.5.3.1	Camera	93
5.5.3.2	Radar	93
5.5.3.3	LIDAR	95
5.5.4	Movement Classification	95
5.6	Experiments	96
5.6.1	System Setup	96
5.6.2	Data Collection and Evaluation Methodology	99
5.6.3	Tracking Results	99
5.7	Summary	103
6	Toward a Holistic Approach to Moving Object Tracking	104
6.1	Introduction	105
6.2	Problem Formulation	107

6.3	Integration with Lane Detection	109
6.3.1	Lane Detection	109
6.3.2	Fusion with Lane Detection	110
6.4	Experiments	112
6.4.1	Qualitative Evaluation of Lane Detection System	113
6.4.2	Evaluation of Orientation Estimation	116
6.4.3	Evaluation on Trajectory Prediction	119
6.5	Summary	120
7	Conclusions and Perspectives	123
7.1	Key Contributions	124
7.2	Limitations	125
7.3	Future Work	126
Appendix A. Data Sets		128
.1	Bicycle Data Set	128
.2	Motorcycle Data Set	130
.3	Vehicle Data Set	132
References		135

List of Figures

1.1	Current self-driving vehicles.	3
1.2	High-level architecture for the perception system. Perception system usually provides four main types of information to the rest of system (i.e., Planning and Behavior). Those are static object map, moving object list, road structure, and traffic context.	5
1.3	Amount of information vs. cost for automotive sensors. A camera lies in a very interesting position in this diagram: low cost and high information.	6
1.4	Sample images showing the complexity of urban (upper row) and highway (lower row) traffic environments.	8
3.1	Illustration of a procedure for mutiscale object detection with an example of vehicle detection. It consists of first (a) creating a densely sampled image pyramid, (b) computing features at each scale, (c) performing classification at all possible locations, and finally (d) performing non-maximal suppression to generate the final set of bounding boxes. The main computational bottlenecks are feature computation for densely sampled image pyramid (HOG features in our case) and sliding-window classification (cascade detection with several parts in our case).	23

3.2	Typical camera configuration for automotive scenarios. We illustrate the algorithm with a case of vehicle detection. The yellow horizontal line represents the detected vanishing line. The green circle represents the location of the vanishing point tracked over frames. All green lines are the ones extracted from a perspective input image.	34
3.3	Illustration for the variation of HOG model size for each level of an image pyramid (left). Illustration for the search space for a level j of the image pyramid (right).	35
3.4	Pedestrian and vehicle detection results using the Caltech Benchmark in [33]: (a) Performance on unoccluded pedestrians over 80 pixels in height. (b) Performance on unoccluded pedestrians between 30-80 pixels in height. (c) Performance on pedestrians at least 50 pixels in height under no or partial occlusion. (d) Performance on unoccluded vehicles over 80 pixels in height. (e) Performance on unoccluded vehicles between 20-80 pixels in height. (f) Performance on vehicles at least 30 pixels in height under no or partial occlusion.	41
3.5	Qualitative detection results on the Caltech testset. The first and second row shows correct pedestrian detections in various scenarios. The third row shows typical false positives (indicated with red arrows).	42
3.6	Qualitative detection results on the Caltech testset (More examples).	43
3.7	Qualitative detection results on our vehicle dataset. The first and second row shows correct vehicle detections in various scenarios. The third row shows typical false positives (indicated with red arrows).	45
3.8	Qualitative detection results on our vehicle dataset (More examples).	46

3.9	Effect of inner motion features on pedestrian detection: (a) Performance on unoccluded pedestrians over 80 pixels in height. (b) Performance on unoccluded pedestrians between 30-80 pixels in height. (c) Performance on pedestrians at least 50 pixels in height under no or partial occlusion.	48
3.10	Our experimental vehicle and its sub-components. (a) Autonomous Cadillac SRX [109], (b) Forward-looking camera (monocular setup), (c) Computing cluster and control console.	50
3.11	Plot of \mathbf{ROI}_y as a function of d (Eq. 3.7). Based on this analysis, all models are designed and visualized here. (a) Normal-sized pedestrian model (96×40 with 8×8 HOG cell). (b) Small-sized pedestrian model (48×20 with 4×4 HOG cell). (c) Normal-sized vehicle model (48×48 with 8×8 HOG cell). (d) Small-sized vehicle model (16×16 with 4×4 HOG cell).	52
3.12	Empirical distribution of bounding box sizes for an SUV case. This statistics is calculated from 9,481 labeled bounding boxes of the SUV class. Measurements for (a) height and (b) width are well fit into the geometry model. (SUV width max : 2.0m, SUV width min : 1.8m, SUV height max : 2.0m, SUV height min : 1.7m, Camera height : 1.3m)	53
3.13	Typical snapshots from the online operation of our on-board pedestrian and vehicle detection system: (a) PPS and (b) FCW on our autonomous vehicle. (c) shows typical false positives for pedestrians (blue: pedestrians, yellow: vehicles, labeling: (class: distance in meter).	54
3.14	Typical snapshots from the online operation of our on-board pedestrian and vehicle detection system: (c) and (d) show pedestrian and vehicle detection results, respectively, in various scenarios. The purple horizontal line represents the detected vanishing line. The green circle represents the location of the vanishing point tracked over frames. All green lines are the ones extracted from a perspective input image.	55

LIST OF FIGURES

4.1	System block diagram for our vision-based multi-object detection and tracking system.	60
4.2	Illustration of our vision-based single object tracking process. . . .	63
4.3	Visualization of three major object models trained from our new urban datasets. For each model, the first, second, and third columns represent a root filter and part filters, and a deformation model, respectively.	72
4.4	Performance analysis on single object tracking. (a), (b), and (c) show typical raw detections obtained by our bicyclist detector and (d), (e), and (f) show the estimated path trajectory of the tracked bicyclists, respectively. The ellipses represent the 1, 2, 3-sigma confidence regions for the position estimate.	74
4.5	Qualitative multi-object tracking results on the Caltech test set. .	75
4.6	Qualitative multi-object tracking results on our vehicle data set. .	76
5.1	CMU’s new autonomous vehicle and its sensor configuration. (a) CMU’s new autonomous vehicle is designed to minimize alterations of a stock-car appearance while installing multiple sensors to maximize the sensing coverage. (b) Visualization of LIDAR measurement. LIDAR scans acquired from individual sensors are depicted in different color. (c) A horizontal field of view (HFOV) of sensing coverage, emphasizing the coverage around the vehicle.	81
5.2	(a) Definition of sensor coordinate systems as well as the vehicle coordinate system. (b) The mounting location of sensors on the vehicles.	83
5.3	Example images show raw sensor data as a function of distances. .	85
5.4	Example images show raw sensor data and extracted features as measurements. (a) An input scene. (b) Raw data from six radars. The data are used as features (called ‘point target’) for tracking directly. (c) Raw scan data from six LIDARs. “L” shaped features (called ‘edge target’) are extracted for tracking. (d) Bounding boxes from the vehicle detection system are used as features (called ‘vision target’) for tracking.	87

5.5	A diagram of our tracking system. Our system is mainly comprised of two layers: a sensor and a fusion layer. We enhance and improve the architecture of our earlier tracking system [29] by adding a vision sensor.	89
5.6	Illustration of data association methods for each sensor. (a) Camera, (b) LIDAR, and (c) Radar.	94
5.7	Our experimental vehicle and its sub-components. (a) Autonomous Cadillac SRX [109], (b) Display and control console, (c) Computing cluster.	97
5.8	Pixel height in image space as a function of distance d from the camera. Based on this analysis, all models are designed and visualized here. (a) Normal-sized pedestrian/bicyclist model (72×32 with 8×8 HOG cell). (b) Small-sized pedestrian/bicyclist model (36×16 with 4×4 HOG cell). (c) Normal-sized vehicle model (48×48 with 8×8 HOG cell). (d) Small-sized vehicle model (16×16 with 4×4 HOG cell).	98
5.9	Screenshot of our evaluation tool [66]	99
5.10	Typical tracking results for the qualitative evaluation. Tracking of a pedestrian (a) and a bicyclist (b), which was enabled by the vision recognition system. (c) Mirroring target issue (see text for the detail). (d) Tracking of a vehicle at far distance.	101
5.11	Typical tracking results for the qualitative evaluation (More examples). (e)~(h) Vehicle tracking results in various situations.	102
6.1	Contextual interactions between moving objects and traffic environments. Observe the relationship between a lane markers (l) and moving objects (o), especially vehicles. This relationship leads us to a simple and efficient probabilistic model for the traffic scene in an intuitive way.	105
6.2	Graphical model for the problem formulation. This simple probabilistic model captures the relationship between moving vehicles (o_i) and lane marker detections (l_i). Shaded nodes indicate observable variables whereas the unshaded nodes indicate hidden variables.	108

LIST OF FIGURES

6.3	Illustration of lane association method for each vehicle target. . .	111
6.4	Route for the evaluation. Four different straight road segments (Session1~Session4) were specified for a quantitative evaluation of orientation estimation.	113
6.5	Qualitative evaluation of the lane marker detection system.	114
6.6	Qualitative evaluation of the lane marker detection system (More examples).	115
6.7	Qualitative evaluation of vehicle orientation estimation. (a) Case of correct orientation estimation. (b) Case of incorrect orientation estimation. In each case, the upper (lower) figure shows a result without (with) lane marker fusion.	117
6.8	Qualitative evaluation of vehicle orientation estimation. (c) and (d) show more cases of incorrect orientation estimation.	118
6.9	Qualitative evaluation on vehicle trajectory prediction. In each case, the upper (lower) figure shows a result without (with) lane marker fusion.	121
6.10	Qualitative evaluation on vehicle trajectory prediction (More examples).	122
1	Some sample images from the bicycle data set.	129
2	Some sample images with ground truth bounding boxes.	129
3	Distribution of bicycle samples (left) and distribution of bottom lines of all bounding boxes (right).	130
4	Some sample images from the motorcycle data set.	131
5	Some sample images with ground truth bounding boxes.	131
6	Distribution of motorcycle samples (left) and distribution of bottom lines of all bounding boxes (right).	132
7	Some sample images from the vehicle data set.	133
8	Some sample images with ground truth bounding boxes.	134
9	Distribution of vehicle sub-type samples (left) and distribution of bottom lines of all bounding boxes (right).	134

List of Tables

3.1	Profiling Results for All Implementations (Unit:ms)	27
3.2	Different sampling schemes for model training	37
3.3	Different number of parts	38
3.4	Different number of scales per octave	39
3.5	System level detection performance of the proposed system.	53
4.1	Details of single bicycle tracking result.	74
4.2	Details of multi-object sequences used in the evaluation.	77
5.1	Installed sensors specifications and their primary usages.	82
5.2	Quantitative evaluation of our multi-sensor tracking system. Total Seconds - Session1: 900sec, Session2: 600sec. Total Objects - Session1: 1,762, Session2: 1,371	100
6.1	Variables used for the lane marker model and their meaning	112
6.2	Quantitative evaluation on orientation estimation (Unit: degree).	116
7.1	More examples of contextual cues and the possible challenges for integration.	127
2	Bounding box distribution.	130
3	Bounding box distribution.	132
4	Bounding box distribution for each vehicle sub-type.	133

Chapter 1

Introduction

Since Karl Benz introduced his first “*Motorwagen*” in 1885, the automotive industry has been a driving force for technological innovation and economic growth. In 2010, it was estimated that the number of vehicles over the world had risen to over one billion vehicles [92]. It is quite remarkable when we realize the fact that this dazzling expansion happened in just 125 years and driving became an essential part of our daily life. Now, in the early decades of the 21st century, the pace of innovation is becoming more dramatic and our society is on the verge of a new technological revolution: “self-driving” vehicles.

This chapter briefly discusses the motivation of autonomous driving technology and its recent progress. Then, it introduces the perception problem of self-driving vehicles, particularly, focusing on a problem of detecting and tracking of moving objects. We discuss the main challenges involved and briefly present our approach highlighting the contributions of the thesis. We begin in Section 1.1 and 1.2 with motivation and recent progress on autonomous driving. Section 1.3 and Section 1.4 present perception systems and currently available sensors for self-driving vehicles, respectively. Then, Section 1.5 introduces the problem of detecting and tracking of moving objects and discusses its main challenges. Finally, we conclude with a discussion of the major contributions and an outline of the structure of the dissertation in Section 1.7 and Section 1.8.

1.1 Motivation

A self-driving or fully autonomous vehicle suggests various positive effects to the society. First, it can improve the driving safety and save a huge number of human lives from vehicle accidents. According to [63], there were approximately 6 million vehicle crashes in 2010 leading to 32,788 human deaths and of the 6 million crashes, 93% are attributable to human error. Secondly, self-driving cars can save the millions of hours wasted in traffic jams because it can drive a little bit closer together on the road by relying on machine (i.e., sensors) precision rather than human's. On average, American commuter spends around 250 hours a year behind the wheel of a vehicle [107]. This time can be better used for more creative and productive work. Thirdly, it can provide a better quality of life to the disabled and senior citizens by offering greater mobility to them. Lastly, it also provides economic benefits to industry by enabling longer hours of vehicle operation because, with proper maintenance, machines never get tired. From these contexts, we believe that self-driving vehicles are now escaping from science fiction books and becoming a reality in the near future.

1.2 Recent Progress on Autonomous Driving

Fully autonomous driving through a busy city street has long been a dream to the intelligent vehicle and robotics community since the invention of automobiles. Traffic environments, especially urban traffic scene, pose lots of challenges in terms of perception, reasoning (i.e., decision making), and motion planning. Although not there yet, impressive progress has been achieved towards autonomous vehicles in the past few years, such as achievements from most teams that participated in the 2007 Urban Challenge [102, 60, 71, 69, 85], the Grand Cooperative Driving Challenge [91], and VisLab's Intercontinental Autonomous Challenge [18]. Yet another impressive work would be the 'Google Car' project. Since they revealed their robotic vehicles to the public for the first time [73] in 2010, they have reported safe autonomous driving more than 300,000 miles without a single accident under computer control [101]. With lessons from the Urban Challenge, Carnegie Mellon University has been developing a new autonomous



(a) CMU Boss [102]



(b) Stanford Junior [71]



(c) Victor Tango [85]



(d) MIT Talos [60]



(e) Cornell Skynet [69]



(f) KIT Annieway [55]



(g) Google Car [73, 101]



(h) VisLab [18]



(i) CMU Cadillac [109]



(j) Toyota [16]



(k) Nissan [94]



(l) Mercedes-Benz [26]

Figure 1.1: Current self-driving vehicles.

vehicle based on General Motors Cadillac SRX4 [109] since 2010. This new experimental vehicle has been used as a research platform for various aspects of autonomous driving technologies, ranging from reasoning, planning, and perception to V2V/V2I (Vehicle-to-Vehicle/Vehicle-to-Infrastructure) communication. In addition, major automotive manufacturers, including General Motors, Ford, Mercedes Benz, Volkswagen, Audi, Nissan, Toyota, BMW, and Volvo, are now participating in autonomous driving efforts and actively developing their own self-driving technology [16, 94, 26]. Figure 1.1 shows current self-driving vehicles.

1.3 Perception System for Self-Driving Vehicles

Autonomous vehicles that operate in urban environments must deal with a number of challenging perceptual problems. These include, but are not limited to, the detection and tracking of road shapes; the detection and avoidance of static obstacles; the detection, tracking, and prediction of other moving objects; recognition of traffic signs and signals; and situational reasoning for complex urban situations such as stop-and-go, queueing at traffic signals, merging into and out of moving traffic, and following precedence rules at intersections. A typical conceptual architecture for the perception system is shown in Figure 1.2.

Through the DARPA Urban Challenge [102], we learned that perception is one of the difficult challenges to accomplish fully autonomous urban driving. Indeed, the perception capability that was necessary to win the Urban Challenge, while impressive, is insufficient for a vehicle to operate on real roads and our primary focus has been on extending the perception system to deal with the complexity of real urban environments. This includes detection and tracking of various types of road participants such as different types of vehicles (e.g., sedan, SUV, bus, and truck) and vulnerable road users (e.g., pedestrians, bicyclists, and motorcyclists), as well as their classification. In addition, more general road structure (i.e., its width and curvature) and traffic context (i.e., traffic light/sign and workzone sign) estimation frameworks are also key components.

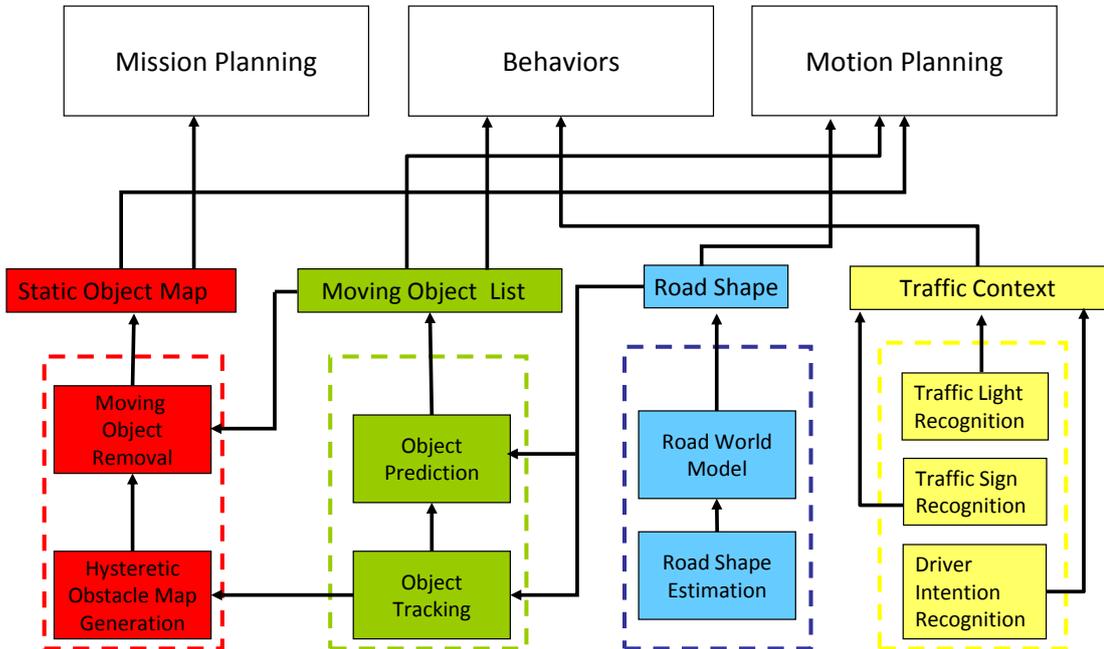


Figure 1.2: High-level architecture for the perception system. Perception system usually provides four main types of information to the rest of system (i.e., Planning and Behavior). Those are static object map, moving object list, road structure, and traffic context.

1.4 Automotive Sensors

Sensors are essential parts of autonomous vehicles, especially for the perception system. Due to their complementary nature, widely used current sensing modalities are radar, LIDAR, and camera.

Radar is primarily used for object detection and uses radio waves to determine the 2D position and velocity of objects. Because radar uses radio waves, it provides very reliable sensing capability even in adverse weather conditions and at longer operation range (e.g., long-range radar). Radar is already widely used in automotive industry for ADAS (Advanced Driver Assistance Systems) such as ACC (Adaptive Cruise Control) or FCW (Forward Collision Warning). Bosch LRR3 (Long-Range Radar) [1], Delphi ESR (Electronically Scanning Radar) [3] multi-range radars, and Continental’s ARS300 LRR (Long-Range Radar) [2] are widely used automotive grade sensors.

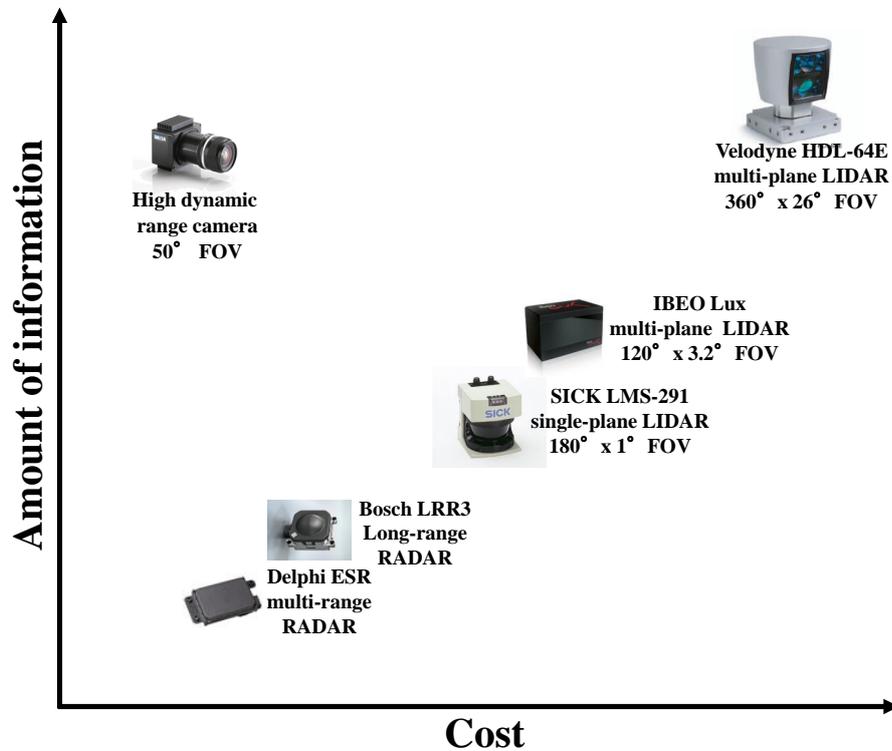


Figure 1.3: Amount of information vs. cost for automotive sensors. A camera lies in a very interesting position in this diagram: low cost and high information.

LIDAR, also known as laser rangefinder, is a remote sensing technology which measures distance by illuminating an object with a laser and analyzing the reflected light. LIDAR generally provides point cloud data, sometimes together with corresponding light intensity. In the market, there exists various types of LIDAR sensors: single-plane vs. multi-plane or indoor vs. outdoor, etc. LIDAR sensors have not been widely used in automotive industry mainly due to mechanically spinning parts and high cost. Currently, Velodyne HDL-64E [10] is known as the most advanced 3D LIDAR sensor and widely being used in current self-driving vehicles (see Figure 1.1). IBEO Lux [7] is a multi-layer (e.g., four or eight layers) LIDAR which targets at automotive applications. SICK LMS series [9] has been widely used in robotics mostly for the SLAM (Simultaneous Localization And Mapping) and moving object detection and tracking applications.

Cameras with computer vision algorithms have a number of benefits as well

as challenges that must be contended with in order to be useful. On one hand, cameras provide a high-resolution view of the scene compared to planar LIDAR or low-resolution scanning radar. Additionally, features such as color, texture, shape, and contours can all be extracted from vision systems which are unavailable to those other sensors. Another practical benefit of cameras is the relatively low cost of the sensor itself compared to LIDAR and radar. On the other hand, because the FOV (Field Of View) of vision systems subtends a large area, objects of interest must be extracted from potentially complex backgrounds before they can be processed. Variations in lighting, object size, shape, and so forth mean that vision systems may be able to recognize an object in one set of conditions but may fail to recognize them in other situations. For instance, a well-performing pedestrian detection algorithm at the day time might not work well at the night time. Figure 1.3 shows the relationship between the sensing power and cost for most representative sensors for autonomous vehicles and it can be seen that the cameras offer high information content at relatively low cost.

1.5 Challenges for Moving Object Tracking

Distinguishing between static and moving entities and tracking moving objects reliably from the traffic scene is one important task among the many challenging perception problems mentioned before. As shown in Figure 1.4, urban traffic scene poses lots of challenges in terms of moving object tracking. First, various types of objects such as pedestrians, bicyclists, motorcyclists, and vehicles should be tracked in real-time along with their classification. Classification of moving objects is critical to higher layer components (i.e. behavioral module) for intelligent reasoning. Secondly, the nature of the urban traffic scene is highly dynamic. Moving objects appear only for a short period of time in the sensor's field of view. A tracking system has to initialize a track as fast as possible when an object enters the sensor's FOV. Finally, each sensor (with different sensing modality) has limited capability to observe the scene so that it can only provide noisy measurements about some aspects of the scene. A tracking system that reasonably solves all these difficulties gives an autonomous vehicle the ability to understand complex scenarios of urban driving.



Figure 1.4: Sample images showing the complexity of urban (upper row) and highway (lower row) traffic environments.

1.6 Thesis Statement

In this thesis, we investigate the problem of detecting and tracking of moving objects for autonomous vehicles to operate safely in busy traffic environments. This includes both urban and highway environments. To this end, we first propose a practical multi-sensor configuration for the perception system. Currently, the majority of autonomous vehicles (e.g., Google’s self-driving car) seems to utilize a high-definition LIDAR sensor such as Velodyne HDL-64 [10], which is almost always installed on the roof of a vehicle. We believe that this bulky sensor setup on the roof would be problematic when we start to regard autonomous driving as a part of our daily life. Secondly, to compensate for the lack of such a high-definition LIDAR, we investigate vision sensor’s capability in terms of object

detection, tracking, and classification. This is mainly because current automotive grade LIDAR sensors provide enough measurements for position and velocity estimation of objects, but not enough for reliable recognition or classification of objects. We believe that, as discussed in Section 1.4, multiple vision sensors together with automotive grade LIDAR sensors could replace the role of such a high definition LIDAR sensor. Thirdly and more importantly, we propose a holistic approach for a multi-sensor, multi-object tracking system. The meaning of ‘holistic’ here is taking into account contextual information from vision algorithms (e.g., object classification and lane marker detection) to generate a coherent tracking result for traffic environments. We integrate vision-based object detection, road geometry information from lane marker detection (possibly from a prior map, e.g., Road Network Definition File (RNDF)), and multi-sensor tracking system in a probabilistic model and propose an efficient procedure to perform real-time estimation with the model.

1.7 Summary of Contributions

This thesis studies the problem of detecting and tracking of moving objects for self-driving vehicles. It makes four main contributions to this goal:

- The first contribution of this thesis is an improvement of a state-of-the-art object detection method called ‘Deformable Part-based Models (DPM)’ [41] in terms of speed and accuracy. This method was selected due to its superior detection performance and the potential for real-time operation based on our thorough investigation of the state-of-the-art object detection methods. However, multiscale object detection is a computationally intensive task. The lack of scale invariance property of a model used in detection requires a detector to generate a densely sampled image pyramid (corresponding feature pyramid) and to run its classifier on it. For example, the original MATLAB implementation of the DPM method runs at around 1 frame per second on a modern PC. Most recent approaches attack two aspects: either accelerating feature computation (such as with feature approximation in nearby scales [31]) or speeding up sliding-window classification with object

models (such as with building a cascade classifier [40]). In this thesis, our approach to overcome the speed challenge is two-pronged: (1) complete C implementation of the baseline method with a multi-threading technique based on multi-cores and (2) an algorithm to define region of interest (ROI) based on known scene geometry. We also improve its detection accuracy by designing object models (e.g., pedestrian and vehicle) optimized for automotive applications. This effort allows us to run one model at the speed of 14 Hz with the state-of-the-art detection performance. Furthermore, for the pedestrian case, we achieved a significant improvement on detection rate by adding motion features (we call ‘*inner motion features*’) to the DPM.

- The second contribution is a vision-based multi-object tracking method incorporating a Rao-Blackwellized Particle Filter which runs a particle filter for data association and an EKF for each object tracking. The original theory for this framework was developed by Särkkä et al. and named as ‘RBM-CDA’ [98]. We applied the algorithm to the problem of vision-based object tracking by improving the measurement likelihood computation which takes advantage of rich appearance information from images.
- The third contribution is an introduction of a novel sensor fusion system which exploits high-level semantic information from vision-based object detection with intermediate features from radar and LIDAR sensors. This object class information is utilized as a key factor in several sub-components of the tracking system such as model selection, data association, and movement classification. A more appropriate tracking model (either a box model or a point model) is selected depending on the object class information and quality of the sensor data. In addition, when a LIDAR’s edge feature is associated with an object being tracked with a box model, the class information is used for pruning unnecessary interpretation of the edge feature for vehicles. Finally, a different parameter set can be used for movement classification based on an object type.
- The final contribution of this thesis is a holistic approach for a multi-sensor, multi-object tracking system. With the contributions we made for the

multi-sensor object tracking, what we want to exploit additionally are contextual interactions between moving objects and traffic environments. In other words, we try to understand a moving object tracking problem in the whole context of a traffic scene. The main context cue we investigated for moving object tracking was a road structure, specifically road lane markers. Lane marker information from a proprietary lane detection system is exploited to improve tracking performance of our multi-sensor tracking system. The fusion brings two benefits to the tracking system. It not only improves orientation estimation of moving vehicles by fusing the tangential heading angle of the associated lane marker but also enables prediction of future trajectories of the moving vehicles.

1.8 Outline of the Dissertation

This section gives an overview on the organization of the following chapters. We briefly summarize each chapter.

- **Chapter 2** describes the state-of-the-art in autonomous driving, vision-based object detection and tracking, multi-sensor tracking system, and holistic traffic scene understanding.
- **Chapter 3** presents our onboard vision system based on the deformable part-based models for real-time pedestrian, bicyclist, and vehicle detection. We review the detection part of the original algorithm and explain how we re-implement the algorithm for the real-time operation. This chapter also discusses the use of motion features to improve detection performance for pedestrians. Especially, motion occurring from each part, what we call ‘inner motion’, is exploited. In addition, a new ROI (Region Of Interest) algorithm was proposed for further speedup for the system. Finally, the vision system consisting of several components is evaluated qualitatively and quantitatively.
- **Chapter 4** presents a monocular vision based multi-object tracking framework for autonomous vehicles based on a detection system we developed in

Chapter 3 and a tracking method using a Rao-Blackwellized Monte Carlo Data Association (RBMCD) algorithm. To reliably detect major traffic participants such as vehicles, bicyclists, and pedestrians under a variety of circumstances, corresponding deformable part-based models are designed and trained using the system from Chapter 3. These robust object detectors provide a series of measurements (i.e., bounding boxes) in the context of recursive Bayesian filtering. Secondly, to exploit the unique characteristics of objects' motion dynamics, one of two motion models, either constant velocity (CV) model or simplified bicycle model (SB), is selectively used based on the object class information from the detectors. For each motion model, an extended Kalman filter (EKF) is used to estimate the position and velocity of an object in the vehicle coordinate frame. Finally, a single object tracking method using an EKF is extended to that of multiple object tracking by incorporating a Rao-Blackwellized Particle Filter which runs a particle filter for a data association and an EKF for each object tracking. We demonstrate the effectiveness of this tracking-by-detection framework through a series of experiments run on a new object dataset captured from a vehicle-mounted camera.

- **Chapter 5** presents our new moving object detection and tracking system that extends and improves our earlier system used for the 2007 DARPA Urban Challenge. We revised our earlier motion and observation models for active sensors (i.e., radars and LIDARs) and introduced a vision sensor. In the new system, the vision module detects pedestrians, bicyclists, and vehicles to generate corresponding vision targets. Our system utilizes this visual recognition information to improve a tracking model selection, data association, and movement classification of our earlier system. Through the test data of actual driving, we demonstrate the improvement and performance gain of our new tracking system.
- **Chapter 6** presents a holistic approach for the moving object tracking system we developed in Chapter 5. For lane detection, we used MobilEye's proprietary lane detection system [8]. Following the very basic fact of 'cars follow roads', the idea of fusing the vehicle's orientation estimate with

tangential lane directions was formulated as a MAP estimation problem. MobilEye's lane marker detection system was qualitatively evaluated on different scenarios. Then, the fusion with lane markers was quantitatively evaluated using four different log sessions. Finally, trajectory prediction module was also properly evaluated. Our experiments show that the holistic approach significantly improves the estimation of vehicles orientation and prediction of vehicles future trajectories.

- **Chapter 7** summarises our approach and our key results, and provides a discussion of the advantages and limitations of the proposed system. It also provides some suggested directions for future research in this area.

Chapter 2

Related Work

This chapter reviews the state-of-the-art in autonomous driving systems with particular attention to environment perception. This includes object detection and tracking in computer vision, and multi-sensor fusion. Section 2.1 provides a brief history of autonomous driving technology. Section 2.2 and Section review, respectively, the state-of-the-art in object detection and object tracking in computer vision. Finally, Section 2.4 summarizes most relevant previous works in multi-sensor fusion based object tracking in mobile robotics.

2.1 Brief History of Autonomous Driving

The whole story about the history of the development of autonomous driving technology is beyond of the scope of this thesis. For detailed contents, we refer the reader to the relevant Wikipedia page [6]. Here, we only summarize the main events in autonomous driving history from a perception perspective.

In the 1980s, a pioneer Ernst Dickmanns and his team in collaboration with Daimler successfully demonstrated the first autonomous driving based on cameras with a Mercedes-Benz van in highway environments without traffic. This success subsequently initiated the European Commission funded EUREKA Prometheus Project on autonomous vehicles (1987~1995). In 1995 the team demonstrated semi-autonomous driving in real traffic from Munich in Germany to Odense in Denmark at speeds up to 175 km/h, with human intervention for about 5%

of the distance. Around the same time, the CMU Navlab team achieved 98.2% autonomous driving with manual longitudinal control using the RALPH (Rapidly Adapting Lateral Position Handler) system [81]. Right after that, the research group of the University of Parma led by Alberto Broggi launched the ARGO Project [19] and demonstrated 94% autonomous driving on a journey of 1,900 km over six days.

The 2005 DARPA Grand Challenge [100, 103] and the 2007 DARPA Urban Challenge [102, 60, 71, 69, 85] offered researchers with unique opportunities to improve and demonstrate autonomous driving technologies. These events were milestones in that they provided opportunities for reevaluating the status of the relevant technologies and for regaining the public attention on self-driving car development. Since then, the related technologies have been significantly improved. Industry and academia have reported notable achievements including: autonomous driving of more than 300,000 miles in daily driving contexts [101], intercontinental autonomous driving [18], a self-driving car with a stock-car appearance [109], and many more. Such developments and demonstrations increase the possibility of self-driving cars in near future.

Similarly to the participants of the DARPA Urban Challenge, the Google driver-less car [101] is equipped with a Velodyne 3D laser scanner for perception and requires manually annotated maps at lane-level accuracy for path planning. Furthermore, its precise localization system is based on registering depth and reflectance measurements with respect to a 3D map, which is recorded a-priori.

2.2 Vision-Based Object Detection

Object detection in static images and videos has been one of the hottest topics in the computer vision and pattern recognition research communities. Due to application domains of concern in this dissertation, we mainly focus on pedestrian and vehicle detection. In addition, we only focus on research using a monocular camera in the visible spectrum. Thus, we omit work related to the use of infrared cameras and stereo vision.

Pedestrian Detection: For the earlier work on this topic, please refer to the surveys of Gavrila [46] and Li et al. [61]. For more comprehensive surveys,

including the most recent research efforts in the field, please refer to [45], [37]. Dollár et al. [33] focuses primarily on the pedestrian detection problem and performs an extensive evaluation of the majority of the state-of-the-art detector algorithms. Gerónimo et al. [49] focuses on pedestrian protection systems for advanced driver assistance systems which utilize tracking, scene geometry, and stereo systems. We review only the important advances for pedestrian detection.

Historically, one of the first pioneering pedestrian detection efforts was the work of Papageorgiou et al. [78] which used Haar wavelet features in combination with a polynomial Support Vector Machine (SVM). They also introduced the first generation pedestrian dataset, known as the ‘MIT Pedestrian Dataset’. Inspired by their work, Viola and Jones [104] brought several important ideas into this field including the use of a new machine learning algorithm (AdaBoost) for automatic feature selection, the use of a cascade structure classifier for efficient detection, and finally the use of an integral image for fast feature computation. Later, the authors demonstrated how to incorporate space-time information into their simple Haar-like wavelet features for moving people detection [105].

The next breakthrough in the pedestrian detection technology occurred in a feature domain. Dalal and Triggs [27] demonstrated excellent performance in detecting a human in a static image using dense histogram of oriented gradient (HOG) features with linear SVM. They also introduced the second generation pedestrian dataset, called the ‘INRIA Person Dataset.’ Currently, HOG is considered to be the most discriminative single feature and is used in nearly all modern detectors in some form [33].

Detectors that improve upon the performance of HOG utilize either a fusion of multiple features or part-based approaches. Wojek and Schiele [111] exploit several combinations of multiple features such as Haar-like features, shapelets, shape context, and HOG features. This approach is extended by Walk et al. [106] by adding local color self-similarity and motion features. Wang et al. [108] combined a texture descriptor based on local binary patterns (LBP) with HOG. Recently, Dollár et al. [32] provided a simple and uniform framework for integrating multiple feature types including LUV color channels, grayscale, and gradient magnitude quantized by orientation. That group also implemented a near real-time [31] and real-time [30] versions of the algorithm which makes this method

suitable for automotive applications.

Parts-based approaches have gained popularity mainly because they can handle the varying appearances of pedestrians (due to clothing, pose, and occlusion) and can provide a more flexible model for pedestrian detection. Mohan et al. [78] take this approach to divide the human body into four parts: head, legs, left arm, and right arm. Each part detector is trained using a polynomial SVM whose outputs are fed into a final classifier after checking geometric plausibility. Mikolajczyk et al. [68] model humans as assemblies of parts that are represented by SIFT-like orientation features. Felzenszwalb et al. [41] demonstrated that their deformable part-based model (DPM) human detector can outperform many of existing current single-template-based detectors [105, 27, 108]. Based on a variation of HOG features, they introduce a latent SVM formulation for training a part-based model from overall bounding box information without part location labels. Bar-Hillel et al. [13] introduced a new approach for learning part-based human detection through a process called ‘feature synthesis.’

Pedestrian detection algorithms have an obvious role in automotive applications due to their potential for improving safety systems. In this case, the design criterion for a detector might be very different as real-time operation is just as important as high detection accuracy. Shashua et al. [88] proposed a part-based representation in a fixed configuration for pedestrian detection. The authors used 13 overlapping parts with HOG-like features and ridge regression to learn a classifier for each part. Gavrila and Munder [47] proposed a pipeline using Chamfer matching and several image based verification steps for a stereo camera setup.

Vehicle Detection: Sun et al. [97] provides a comprehensive survey on the topic. Historically, according to [97], researchers in automotive field used low-level features to detect vehicles in image streams. Such low-level features included strong vertical edge pairs, horizontal shadow patterns cast by a leading vehicle, and textures. However, discriminative power from these low-level features has not been sufficient for real-world automotive applications. Similar to research in pedestrian detection, object classification methods with more sophisticated features became more widely used in the field [50, 59, 72, 99]. Han et al. [50] applied HOG-SVM combination and Moutarde et al. [72] applied Haar-AdaBoost combination to both vehicle and pedestrian detection. Leibe et al. [59]

deployed a set of five single-view vehicle detectors using ‘*implicit shape models*’ in combination with a real-time structure-from-motion (SfM) system. Takeuchi et al. [99] applied ‘deformable part-based model’ [41] to vehicle detection and combined its bounding box output with a particle filter-based tracking.

2.3 Vision-Based Object Tracking

Mathematical foundation of majority of vision-based multi-object tracking approaches is originated or, at least, inspired by the classical multi-target tracking (MTT) theory developed for military applications [84, 14] by following its structural decomposition of the problem, i.e., dynamic models, observation models, and data association. Among these, the issue of resolving the data association lies at the heart of the solution for the MTT problem and, hence, many popular methods such as Multiple Hypothesis Tracking (MHT) [84], Joint Probabilistic Data Association (JPDA) [14], and other Sequential Monte Carlo (SMC)-based methods [34] have been developed.

While the MTT theory is largely based on radar point tracking scenarios (i.e., missile or aircraft tracking), vision-based tracking introduces very interesting additional aspects to the problem arising from the unique characteristics of vision sensors, namely, cameras. First, a camera provides rich appearance information in contrast to a radar sensor which usually provides a point position and velocity measurement. Secondly, a camera provides projected 2D images of the real 3D world, where measurements from a radar sensor are measured in the world coordinate. Regarding the first issue, appearance information in images has been largely exploited. There are vision-specific algorithms such as Mean-Shift [25] and Lucas-Kanade [62] for object tracking, hence named as ‘visual tracking’, and sometimes those methods are used as a measurement for observation models or an additional clue for the data association in the context of Bayesian filtering. In recent years, with advances of appearance-based object detection algorithms [78, 105, 27, 41, 32], ‘tracking-by-detection’ has become a promising candidate for multi-object tracking [47, 59, 38, 99, 22]. For the second issue, since a measurement is essentially a 2D mapping of the 3D world, state variables associated with a dynamic model can be formulated either in 3D world coordinates [47, 59, 38, 22]

or in 2D image space [77, 99]. Here, based on these two perspectives, we summarize the most relevant previous works for multi-object tracking on a moving platform.

In [47], given the detection output, Gavrilu and Munder used an α - β tracker [15] to track multiple pedestrians in the vehicle coordinates. For data association, they used the Hungarian method [57]. Liebe et al. [59] and Ess et al. [38] introduced a two-stage hypothesize-and-test framework in which an over-complete set of trajectory candidates is generated from an object detector in each time step and is pruned to a consistent subset using statistical model selection. Okuma et al. [77] tracks a varying number of hockey players in videos using the combination of Adaboost object detection and particle filtering, where they used Hue-Saturation-Value (HSV) color histograms for computing measurement likelihood. Takeuchi et al. [99] used the DPM framework (same as our approach) for detecting vehicles and a particle filter for tracking, where they used both a probability from the detector and intensity correlation for computing measurement likelihood.

2.4 Multi-Sensor Fusion for Object Tracking

Detection and tracking of moving objects is a core task in mobile robotics and as well as in the field of intelligent vehicles. Due to such a critical role, this subject has been extensively studied for the past decades. Since a comprehensive literature survey of this topic is beyond the scope of this paper (we refer to [67, 83] for such surveys), here we review only the earlier work on multi-sensor fusion for moving object detection and tracking, which are relevant to our work.

The Navlab group at Carnegie Mellon University has a long history of development of autonomous vehicles and advanced driver assistance systems. For the Navlab 11, one of the latest developments, they proposed a high-level fusion approach for object tracking using cameras and LIDARs [11]. In fact, our feature extraction algorithm for LIDAR is motivated by their method [67]. Another interesting effort was the work of Stiller et al. [95], where they used radar, LIDAR, and stereo vision for obstacle detection and tracking. Although they did not provide quantitative results of the system, it brought researchers' attention to the multi-sensor fusion approach.

Since then, an approach of fusing LIDAR measurements with vision sensors' outputs has gained popularity for vehicle tracking [64, 70] and pedestrian tracking [83, 93]. Monteiro et al. [70] used a single-layer LIDAR and a monocular camera to detect, track, and classify objects. For fast detection and tracking, a LIDAR was used and generated regions of interest (ROIs) to a vision module. For the classification of objects, two classifiers, a Gaussian Mixture Model (GMM) classifier for a LIDAR and an Adaboost classifier for a camera, are applied. A sum decision rule was used to combine both outputs. Mählich et al. [64] focused on a 'cross-calibration' method between these two sensors while showing vehicle tracking. Premebida et al. [83] used a multi-layer LIDAR and a monocular camera for pedestrian detection. They exploited several features for each sensor measurements and classification algorithms for better accuracy. Spinello and Siegwart [93] also utilized a multi-layer LIDAR for detecting hypotheses for pedestrians and then a vision-based pedestrian detector was applied for verification. A Bayesian decomposed expression was used as a reasoning fusion rule.

The 2007 DARPA Urban Challenge provided researchers with a unique opportunity to develop and test the multi-sensor based systems [60, 71, 69]. Due to the practical nature of the competition, high-level fusion approaches were widely exploited. In particular, the Stanford [71] and MIT [60] teams developed a similar object tracking system which utilized a set of LIDAR sensors as primary sensors. Their systems first removed irrelevant measurements, such as laser scans from the ground and from vertical structures, and then fitted geometric primitives (e.g., 2D rectangles) to the remaining measurements to, using Bayesian filters, estimate objects' position, velocity, and size. Similarly, the Cornell team [69] used a Rao-Blackwellized particle filter for moving object tracking, where a data association problem is solved by a particle filter and a state estimation problem is solved by an extended Kalman filter.

Most of the teams developed their own tracking systems to effectively fuse sensor measurements in different modalities. However, due to reliability issue and computational cost, a tracking system based on vision-sensor was not extensively studied for the competition.

Chapter 3

Vision-Based Object Detection with Deformable Part Models

This chapter presents our onboard real-time object recognition system intended for self-driving vehicles as well as for use in automotive active safety applications. For reliable and real-time performance in challenging real world conditions, we exploit a state-of-the-art object detection method called ‘deformable part-based model’ (DPM) [41, 40] and significantly improve its speed and detection performance. Section 3.1 discuss the overview and main contributions of the proposed system. Technical review of the DPM and details of our C implementation are described in Section 3.2. Section 3.3 presents how we add motion features into the DPM and its performance gain is analyzed. Section 3.4 derives the Region of Interest (ROI) computation algorithm and Section 3.5 describes experimental results using the system. Section 3.6 concludes the chapter.

3.1 Introduction

Real-time onboard pedestrian and vehicle detection from a moving vehicle is a challenging task, especially when using a monocular camera as the sole sensor. Although a number of approaches have been proposed, there are still a number of aspects of this problem that are challenging and can be considered to be as of yet “unsolved.” That being said, the accuracy and computational performance

of these detectors are steadily improving [78, 105, 27, 112, 41, 32].

Our efforts to overcome these difficulties of vision systems primarily focus on a) exploiting a rich object model which considers the deformation of parts for reliable detection, b) developing an efficient ROI computing algorithm for real-time detection, and c) exploiting motion features for better detection. Specifically, we make use of the cascade version [40] of the DPM [41] introduced by Felzenszwalb et al., which is considered as one of the most successful methods for general object detection. Our contributions in this chapter are as follows.

The first main contribution is a C implementation of the DPM method suitable for real-time operation. Achieving real-time detection speed is by no means trivial especially for most of the state-of-the-art detectors which usually use a sliding window approach. Specifically, we reimplement the original detection system using C/C++ with parallelism techniques based on multi-cores and achieved real-time detection rate of 14fps on an Intel Core i7 computer when one model is applied to 640×480 images.

The second contribution is a simple, but powerful ROI computation algorithm for increased speedup by using the known camera calibration and a vanishing point tracker. Usually, the availability of camera information is not assumed in general object detection applications, but for automotive applications where the camera is mounted in a known position in the vehicle, this information can be very useful when used with vanishing point tracking. By doing so, we are able to not only accelerate our detection process by searching only relevant image regions, but we are also able to improve detection accuracy by suppressing a number of potential false positives in irrelevant image regions.

The third contribution is our approach to designing and building deformable part-based pedestrian and vehicle models for automotive applications. We improve the detection rate of the baseline detector by training multiresolution models, where a normal sized model with 8×8 cell size is built for normal distances and a small sized model with 4×4 cell size is built for longer distances. The actual size for each model is designed based on scene geometry analysis and these models show a superior performance when detecting far-scale pedestrians and vehicles.

Our final contribution is a quantitative evaluation of our framework using challenging real world datasets. First, following the same series of experiments

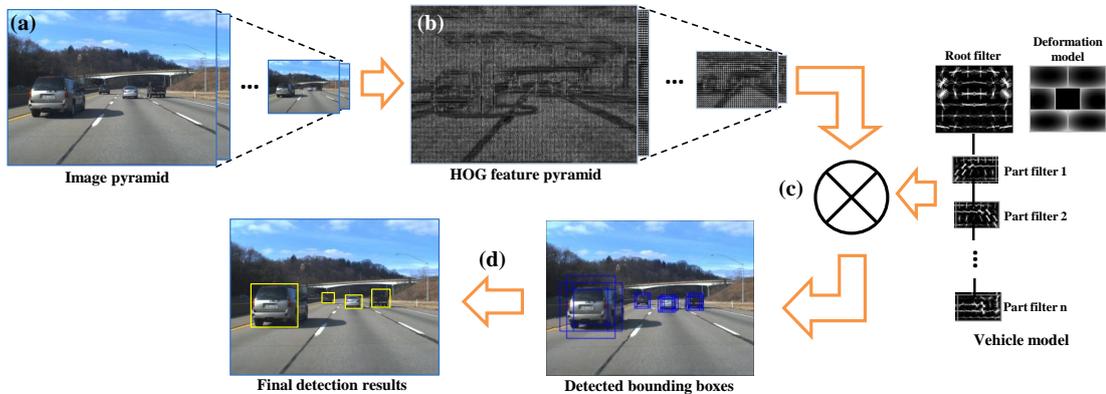


Figure 3.1: Illustration of a procedure for multiscale object detection with an example of vehicle detection. It consists of first (a) creating a densely sampled image pyramid, (b) computing features at each scale, (c) performing classification at all possible locations, and finally (d) performing non-maximal suppression to generate the final set of bounding boxes. The main computational bottlenecks are feature computation for densely sampled image pyramid (HOG features in our case) and sliding-window classification (cascade detection with several parts in our case).

as in [20], we seek to identify values of key design parameters such as the optimal number of parts for the pedestrian and vehicle models, the optimal number of scales per octave for multiscale object detection. Second, with the insights from the first set of experiments, we quantitatively evaluate detection performance for each class of objects using the Caltech benchmark [33]. Lastly, we deploy the proposed detection system on our autonomous vehicle and demonstrate active safety functionalities (i.e., Pedestrian Collision Warning (PCW) and Forward Collision Warning (FCW)) in real-time in real-world scenarios.

3.2 Real-Time Object Detection with DPM

This section discusses the main reasons why we chose the DPM method and reviews its model and detection process and then describes the important details of our implementation that were key to its fast performance.

3.2.1 Why DPM?

We performed a broad survey of the state-of-the-art object detectors and ended up with the methods in [108, 40, 31, 13] as candidates for our implementation. We opted to implement a detector based on the work of Felzenszwalb et al. [41, 40] for the following reasons. First, this method provides an elegant mechanism for handling a wide range of intra-class variability (e.g., multiple views of a vehicle) by having multiple submodels (so-called “components”). Furthermore, the DPM exploits dynamic configurations of its parts which allows for a wider variation (e.g., various poses of pedestrians) in object appearances to be accepted. The importance of this aspect is well illustrated in a recent survey [33], where the DPM method shows better performance than the other detectors which use even multiple features when high-resolution of appearance of the objects is provided. The reason turns out to be the structural richness of the DPM, which considers the deformation of parts. Recent improvements in consumer camera technology mean that low-cost high-resolution cameras are available for use in the automotive domain. Secondly, this method has a well-designed learning technique called “latent SVM” which not only can handle a large training set effectively, but also can learn a part-based model without requiring information about exact part labels. Finally, this approach utilizes an efficient detection mechanism called the star-cascade algorithm [40] which makes it efficient and potentially suitable for real-time applications.

3.2.2 Review of the DPM and Star-Cascade Algorithm

We primarily review the detection part of the model since we are interested in applying the DPMs to real-time applications. For the training part, we refer the readers to [41]. The DPM consists of three main parts: a root filter, part filters, and deformation models. For example, note the vehicle model shown in Figure 3.1. A root filter is a HOG feature template designed to capture an object’s global shape and part filters are smaller HOG feature templates trying to capture details of object’s parts. It is important to note that these part filter’s HOG features should be computed at twice the resolution in a feature pyramid compared to that of root filter [41]. A deformation model associated with each part filter is a

cost map measuring the deviation of the part from its ideal location relative to the root.

We now describe the detection process mathematically. A DPM for an object with n parts is defined by a $(n + 2)$ -tuple $(F_0, P_1, \dots, P_n, b)$ where F_0 is the root filter, P_i is the i -th part model and b is a bias term. Each part model is defined by a 3-tuple (F_i, v_i, d_i) where F_i is a filter for the i -th part, v_i is a two-dimensional vector specifying coefficients of a quadratic placement of the part relative to the root location, and d_i is a four-dimensional vector specifying coefficients of a quadratic function defining a deformation cost for each possible placement of the part relative to the anchor location. An object hypothesis specifies the location of each filter in the model in a feature pyramid, $z = (p_0, \dots, p_n)$, where $p_i = (x_i, y_i, l_i)$ specifies the position (x_i, y_i) and level l_i of the i -th filter in the feature pyramid H . Then, the score of a hypothesis is given by the scores of each filter at their respective locations minus a deformation cost that depends on the relative position of each part with respect to the root, plus the bias, i.e.,

$$\begin{aligned} \text{score}(p_0, \dots, p_n) = \\ \sum_{i=0}^n F'_i \cdot \phi(H, p_i) - \sum_{i=1}^n d_i \cdot \phi_d(dx_i, dy_i) + b \end{aligned} \quad (3.1)$$

where F'_i denotes vectors from the i -th filter F_i and $\phi(H, p_i)$ denotes vectors from the feature pyramid at p_i . $(dx_i, dy_i) = (x_i, y_i) - (2(x_0, y_0) + v_i)$ corresponds to the displacement of the i -th part relative to its anchor position and $\phi_d(dx, dy) = (dx, dy, dx^2, dy^2)$ are deformation features. With this object hypothesis definition, an overall score for each root location is defined as:

$$\text{score}(p_0) = \max_{p_1, \dots, p_n} \text{score}(p_0, \dots, p_n) \quad (3.2)$$

To detect objects in an image, we can compute this overall score for each root location with a sliding-window method and detect objects by thresholding the scores. Indeed, since the interaction between parts is not considered explicitly in

the model, the score can be factored as follows.

$$\begin{aligned} score(p_0) &= F'_0 \cdot \phi(H, p_0) \\ &+ \sum_{i=1}^n \max_{p_i} (F'_i \cdot \phi(H, p_i) - d_i \cdot \phi_d(dx_i, dy_i)) + b \end{aligned} \quad (3.3)$$

The detection algorithm in [41] computes this score, for each level of the feature pyramid, by performing the cross-correlation between all part filters and the feature map, and the generalized distance transform [42] for the max operation in the second term of the Eq. 3.1. For the 6 parts model, for instance, this is going to require 7 cross-correlation and 6 distance transform operations for one level and this set of operations should be repeated for the entire feature pyramid. Although an efficient algorithm for the generalized distance transform [42] is utilized, the speed performance of the detection is not sufficient for real-time applications.

The star-cascade algorithm [40] increases detection speed by more than one order of magnitude compared to its baseline [41] by building a cascade classifier from the star-structured DPM. The basic idea is to use a subset of the parts progressively for pruning low scoring hypotheses, rather than performing all the operations (above mentioned) in advance. In this case, as shown in the Figure 3.1, each part simply constitutes each stage of the cascade classifier. Building such a cascade model corresponds to finding a set of thresholds such that it can achieve the same detection rate as the baseline detector.

3.2.3 Implementation Details

Although the star-cascade algorithm opens the possibility of its use in real-time applications, it is still too slow to be used in real applications (i.e., takes 682ms for the pedestrian model to process one frame on a 2.67GHz Intel Core i7 920 CPU). For this practical challenge, we have engineered this algorithm in an attempt to optimize it to be the core of a real-time object detection for automotive applications. To understand where our efforts would need to be applied in the implementation of this algorithm, we profiled the performance of the original MATLAB/C based detectors. These detectors included the `voc-release3` [43] and `voc-release4` [44] with a star-cascade algorithm [40]. The profiling

Table 3.1: Profiling Results for All Implementations (Unit:ms)

Algorithm Name	Computer I		Computer II	
	Intel Core2 Duo P8800@2.66GHz 2GB RAM		Intel Core i7 2920XM@2.5GHz 16GB RAM	
	Matlab star-cascade	C star-cascade	Matlab star-cascade	C star-cascade
HOG Feature Computation	1145	300	840	80
Sliding Window Classifier	560	320	300	100
Non-Maximal Suppression	24	10	24	5

was performed using two evaluation computers that included an Intel Core2 Duo P8800@2.66GHz with 2GB RAM, labeled computer I, and an Intel Core i7 2920XM@2.5GHz with 16GB RAM, labeled computer II. For 640×480 images and 10 scales per octave, `voc-release3` (1 component with 6 parts, multi-threaded version) demonstrated a performance of 0.5 fps and `voc-release4` algorithm with a star-cascade algorithm (1 component with 8 parts, single-threaded version) demonstrated a performance of 0.6 fps on computer I. The details of the profiling result is shown in Table 3.1. As can be seen in this Table, the two main bottlenecks are the computation of HOG features for densely sampled image pyramids and the sliding-window classification.

To tackle this, we re-implemented the algorithm (originally based in MATLAB/C) using C/C++ and optimized several subsystems in our previous work [20]. Basically, we re-implemented `voc-release3` [43] of the system with the star-cascade algorithm using C/C++ and parallelized (assuming a multi-core architecture) most of the computationally intensive functions such as HOG feature computation and cascade detection. Our implementation follows a standard procedure for object detection as illustrated in Figure 3.1. The key details to this

implementation are described below:

Feature computation: Given an input image, the first step of the pipeline is to build a densely sampled image pyramid and compute the corresponding feature pyramid. For an image pyramid, we employed the image resize function of the OpenCV 2.0 library. For the HOG feature pyramid, which was the first computational bottleneck, we refactored the algorithm to make use of the `pthread` library. This was possible because the computational process to generate each level of the pyramid is independent of all the others. This solution allowed us to speed the algorithm up by one order of magnitude.

Sliding-window classification: For the `voc-release3` algorithm, as discussed before, a large number of cross-correlations is the main bottleneck in practice. For this, the original method provides a parallelized correlation scheme with a numerical linear algebra enhanced function. We ported their MEX functions into our implementation. Also, `voc-release4` with a star-cascade algorithm provides an efficient way for classification by evaluating parts in a cascaded fashion with a sequence of thresholds. For implementation of this idea, `voc-release4` uses $2(n+1)$ models for a model with $n+1$ parts, where the first $n+1$ models are obtained by sequentially adding parts with simplified appearance models (PCA version of original HOG feature) for faster computation and second $n+1$ models are obtained by sequentially adding parts with its full feature. Our implementation is slightly different in that we just use $n+1$ cascade models with full HOG feature for ease of implementation and we parallelize the process using `pthread` library. By doing this, we achieve 2X-4X speed improvement.

Non-maximal suppression: After sliding-window classification, we usually get multiple overlapping bounding boxes from detections in nearby locations and scales. Non-maximal suppression (NMS) is used for eliminating those repeated detections. Currently, two dominant NMS approaches have been widely used: mean shift mode estimation introduced in [27] and pairwise max suppression introduced in [41]. We implemented the pairwise max suppression scheme due to its simplicity and efficiency.

3.3 Motion-augmented DPM

Motion information can be very useful for object detection in videos. In particular, pedestrians are the best example for exploiting the motion cue since they not only tend to be in motion but also they exhibit relative motion of their limbs with respect to their center of body. Whether the motion occurs from the body boundary or “inside the body”, it can be characteristic and discriminative, and thus, can be used as a complementary feature along with appearance features (e.g., HOG features in our case). The goal of this section is to investigate how to exploit motion features to improve pedestrian detection performance.

3.3.1 Inner Motion

Thanks to its uniqueness, motion has been widely used in many areas including detection [105, 28, 112, 106, 79], tracking [89, 113], and action recognition [39, 108]. Depending on the origin of the motion, each method has a different assumption and strategy to extract motion features. Thus, we want to categorize motion observed in videos first and then discuss what types of motion we are trying to extract. For this issue, we found the discussion in [79] particularly interesting. In [79], the authors categorized image motion into three types. The first is camera-centric motion which is the motion induced by the movement of the camera itself with respect to the world. The second is object-centric motion which is the motion induced by the movement of the object with respect to the world. Finally, the part-centric motion or, what we call, ‘*inner motion*’ is the motion induced by the relative movement of object parts with respect to the center of object. Having this categorization in mind, the large body of methods [105, 80] based on background subtraction tries to deal with object-centric and part-centric motion since the camera is stationary. Secondly, there are some techniques which compute optical flow in an object-centric coordinate frame [36]. Those methods try to encode both camera- and part-centric motion. Finally, the simplest approach is to directly compute motion features on raw video where all types of motion are superimposed. Most approaches try to remove camera motion by looking at differences of flow [28, 112, 106]. Recently, Park et al. [79] proposed an interesting technique to remove camera- and object-centric motion while pre-

serving the part-centric motion (i.e., inner motion). They proposed a two-stage approach, where they first attempt to stabilize both camera- and object-centric motion by using coarse-scale optical flow to align a sequence of images, and then they use temporal difference to capture the part-centric motion that remains after weak stabilization. The authors hypothesized that the majority of useful motion information is contained in part-centric motion and verified their hypothesis with various sets of experiments.

Following the method in [79], we also try to extract inner motion features, but unlike the method, we want to integrate the inner motion features into the DPM framework. We discuss how to stabilize videos to compensate both camera and object motion using the Lucas-Kanade method [62], how to encode the motion features, and how to integrate the inner motion features into the DPM framework in the next subsections.

3.3.2 Weak Stabilization using Lucas-Kanade

The first step to extract motion features is to weakly stabilize input images to remove both camera- and object-centric motion while preserving the inner motion. The method in [79] achieves this by estimating coarse-scale optical flow to align a sequence of frames. To compute optical flow field for the purpose, they applied the Lucas-Kanade method with a large window radius σ which roughly corresponds to the size of pedestrians in images (i.e., 16 to 64 pixels). By doing so, since the window radius σ controls the scale of the flow, it can preserve fine inner motion.

We denote the computed flow field from frame J_t to frame J_{t-1} as $W_{t,t-1}$. $J_{t-1,t}$ is the frame J_{t-1} warped to frame J_t using the inverse of the flow field $W_{t,t-1}$. We denote an image patch defined by the detection window (i.e., 64×32) from the warped image as $I_{t-1,t}$. Following the practice in [79], when stabilizing across multiple frames, we compute the global motion $W_{t,t-n}$ by progressively warping and summing pairwise flow fields.

3.3.3 Motion Feature Encoding

The second step to extract motion features is to encode motion features from the weakly stabilized frames. Since rough camera and object motion are already re-

moved by weak stabilization, we simply take a temporal gradient which is defined as

$$D^\sigma = I_t - I_{t-1,t} \quad (3.4)$$

where σ is the scale used for Lucas-Kanade method. In many scenarios, since the amount of motion observed between consecutive two frames may be quite small, we consider a simple approach of computing multiple frame differences between the current frame and $k = n/m$ other frames spaced apart temporally by m frames from $t - m$ to $t - n$. We refer to m as the *frame skip* and n as the *frame span*.

$$D^\sigma(n, m) = \begin{bmatrix} I_t - I_{t-1m,t} \\ I_t - I_{t-2m,t} \\ \vdots \\ I_t - I_{t-km,t} \end{bmatrix} \quad (3.5)$$

Following the best result from [79], we use $D^{16}(8, 4)$ for all the experiments. It is important to note that this decision should be based on the original video frame rate. The parameters $D^{16}(8, 4)$ were chosen based on the fact that Caltech Pedestrian data was recorded at 30 frames per second. As Eq. 3.5 indicates, our temporal difference features are computed using the signed temporal gradient. We also tried the absolute value of the temporal gradient and we found that the signed temporal gradient performed better. Next, to add a small amount of spatial invariance, all of the features are pooled over a $c \times c$ sized rectangular window. To make integration with HOG features easy, the same size as static HOG cell size (i.e., 4×4) was used. Finally, we applied a similar L1 block normalization as in [79] to make motion features more robust to varying lighting and background texture, but we applied the normalization only over the 2×2 spatial blocks rather than over spatio-temporal blocks.

3.3.4 Integrating Motion into DPM

In the previous subsections, we discuss how to extract inner motion features based on a) coarse-scale optical flow and b) feature encoding on temporal gradient of weakly stabilized frames. Now, we explain how we integrate the inner motion features into the DPM framework.

Let us review the score function (i.e., Eq. 3.1) for an object hypothesis in the DPM framework. Since we are considering adding inner motion features to each part filter as well as a root filter, it is straightforward to compute a new score function, which is given by

$$\text{score}(p_0, \dots, p_n) = \sum_{i=0}^n M'_i \cdot \phi_m(H_m, p_i) + \sum_{i=0}^n F'_i \cdot \phi_a(H_a, p_i) - \sum_{i=1}^n d_i \cdot \phi_d(dx_i, dy_i) + b \quad (3.6)$$

where M'_i denotes vectors from the i -th inner motion filter M_i and $\phi_m(H_m, p_i)$ denotes vectors from the motion feature pyramid at p_i . Note that we assume the same location for each part for extracting both HOG features and inner motion features. Therefore, by augmenting motion features with HOG features, we can train deformable part-based models using the latent SVM formulation as in [41]. In our case, we let the latent SVM training process find the best model parameters by utilizing not only the appearance features but also the inner motion features.

3.4 ROI Computation

A multiscale object detection is a computationally intensive task. The fundamental reason for this is a lack of scale invariance property of a model used in detection. This requires a detector to generate a densely sampled image pyramid (and corresponding feature pyramid) and to run its classifier on it. Most recent approaches attack this problem in two ways: either accelerating feature computation (such as with feature approximation for nearby scales [31]) or speeding up sliding-window classification with object models (such as with building a cascade classifier [104, 40, 31]). There is yet another interesting approach which is orthogonal to those approaches: the use of scene geometry.

Narrowing down search space by using constraints on the scene geometry is another way to improve overall performance because it can not only reduce the detection time but it can also reduce the false positive rate by only processing an image area which is geometrically valid for object detection. Sudowe and Leibe [96] address this issue and provide a general and practical formula for extracting a safe search space. In this thesis, we propose a rather simple algorithm for computing safe and efficient ROIs assisted by vanishing point tracking.

3.4.1 Geometry

Consider a typical camera configuration for automotive scenarios where an object (vehicle in this case) lies at a certain distance (d), as shown in Figure 3.2. Here, we use pinhole camera model and assume 1) flat ground plane and 2) parallel optical axis to the ground plane (i.e., zero pitch angle of a camera) for ease of formulation. However, these assumptions occasionally may not hold due to the vehicle’s sudden ego-motion and uneven road surface. To deal with such cases, we dynamically estimate the position of a vanishing line by tracking vanishing points using the method in [87]. The vanishing point tracker detects vanishing points by extracting lines from a perspective image. We believe this approach is better than that of using lane markings to find vanishing points since one can almost always extract numerous lines from scene structure (e.g., buildings, bridges, jersey barriers, and curbs) while lane markings are sometimes not available.

From this setting, given the vanishing line estimated via a vanishing point tracking, we can easily compute a pixel height (h_g) from the vanishing line to the bottom line of an object as well as a pixel height (\mathbf{ROI}_y) in the image plane for the object. This is expressed by:

$$h_g = \frac{Y_c \times f_p}{d}, \quad \mathbf{ROI}_y = h_g - h_t = \frac{OH}{Y_c} \times h_g \quad (3.7)$$

where Y_c and OH are camera height and physical object height in meters, respectively. f_p is a focal length expressed in pixels and h_t is defined similarly as h_g . We model variability in object sizes by assuming a maximum and minimum size of an object OH_{min}, OH_{max} , which corresponds to the max and min size of

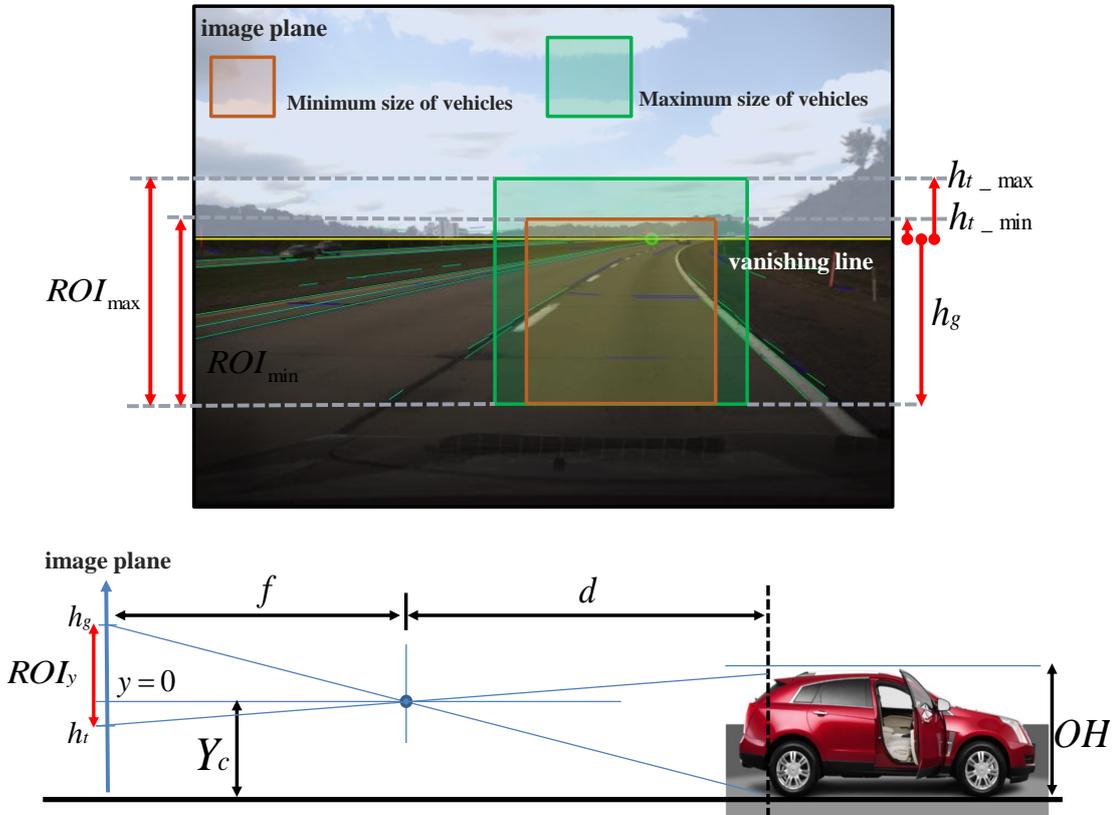


Figure 3.2: Typical camera configuration for automotive scenarios. We illustrate the algorithm with a case of vehicle detection. The yellow horizontal line represents the detected vanishing line. The green circle represents the location of the vanishing point tracked over frames. All green lines are the ones extracted from a perspective input image.

the ROI, \mathbf{ROI}_{min} and \mathbf{ROI}_{max} . As illustrated in Figure 3.2, this relationship between the \mathbf{ROI}_y and h_g suggests the derivation of a simple formula for finding ROIs from an image pyramid. We discuss this derivation in the following.

3.4.2 Formulation

Let us consider a J -level image pyramid and an object model as illustrated in Figure 3.3. We can think of the model as a root filter of the DPM. Level 0 corresponds to an input image and the following levels corresponds to downscaled images. Note that we illustrate HOG model size to be increasing while the image

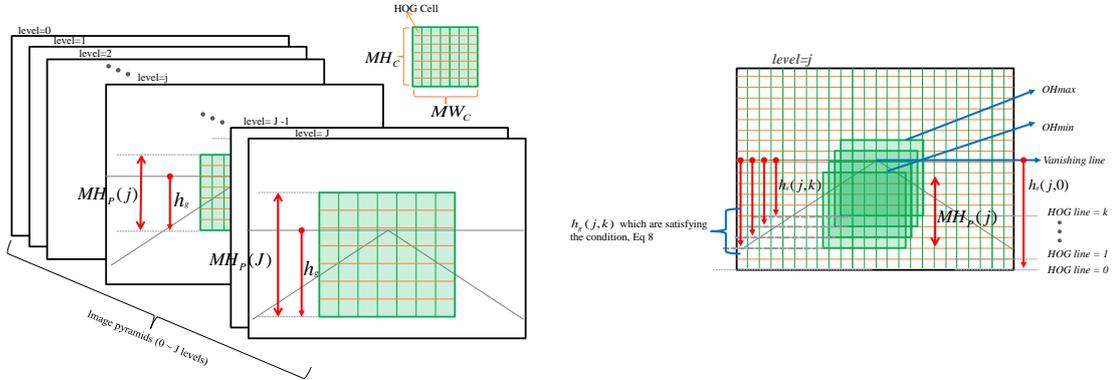


Figure 3.3: Illustration for the variation of HOG model size for each level of an image pyramid (left). Illustration for the search space for a level j of the image pyramid (right).

size remains the same as we go down the feature pyramid. This representation is chosen for better understanding of our algorithm. MH_c and MW_c are model height and width in cells, respectively. MH_p is model height in pixels, so it can be expressed as $MH_p = CH_p \times MH_c$ where CH_p is the number of pixels for one HOG cell (usually 8). Now we want to define MH_p as a function of the level since the coverage of fixed model size changes for different pyramid levels:

$$MH_p(j) = \Delta P(j) \times CH_p \times MH_c, j = 0, 1, \dots, J \quad (3.8)$$

where $\Delta P(j) = 2^{\frac{j}{\lambda}}$, a relative pixel size indicating how many pixels in the original input image would correspond to one pixel in pyramid level j and λ is the number of levels in an octave. Looking back at Eq. 3.7, the next quantity we want to determine is the $h_g(j)$, i.e., how the pixel height (h_g) from the vanishing line to the bottom line of an object is expressed for a level j . In this formulation, we want to parameterize the quantity with one more variable, k , since what we are eventually interested is a search space in the HOG feature pyramid.

$$h_g(j, k) = h_g(0, 0) - k \times CH_p \times \Delta P(j) \quad (3.9)$$

where k is an integer index indicating the order of HOG cell line beginning from the bottom to top. In other words, the physical meaning of $h_g(j, k)$ is the pixel

height (h_g) from the vanishing line to the k -th HOG cell line in level j of the feature pyramid. This is illustrated in the right-hand side of Figure 3.3. Finally, the search space for the level j can be obtained from the k 's which satisfy the following relation (starting line) and the model pixel size (i.e., $MH_p(j)$) at that level (end line):

$$MH_p(j) = \frac{OH}{Y_c} \times h_g(j, k) \quad (3.10)$$

Note that all quantities except k are constants, so that computing k is trivial. Eq. 3.10 is exact but it is formulated based on several assumptions such as the road's evenness and the zero pitch angle of the camera, which may not be met perfectly. Although we alleviate these conditions by continuously updating the horizontal vanishing line, we still model variances of the vanishing line by introducing an error margin $e(j)$. The final equation we use is as follows:

$$MH_p(j) \times \frac{Y_c}{OH_{max}} - e_u(j) < h_g(j, k) < MH_p(j) \times \frac{Y_c}{OH_{min}} + e_l(j) \quad (3.11)$$

where $e_u(j)$ and $e_l(j)$ are upper and lower error margins for level j , respectively. With this explicit modeling of error sources and the vanishing line estimation, our real-world experiments (details in Section 3.5-C) verify that the algorithm generates reliable and efficient ROIs for object detection. With this ROI scheme we achieve a 5X speed increase whereas our previous method [20] showed 3X increase.

3.5 Experiments

To evaluate our object detection system, we analyzed its performance on various real-world datasets. To evaluate pedestrian detection accuracy, we used the Caltech Pedestrian Dataset [33] which is the largest publicly available pedestrian dataset at this time. The Caltech dataset corresponds to approximately 2.3 hours of video (640×480 resolution) captured at 30fps and is segmented into 11 sessions, 6 of which are used for training (S0~S5) and the remaining sessions are used for testing (S6~S10). The dataset offers an opportunity to exploit many different aspects of model training thanks to its large number of positive samples. For

Table 3.2: Different sampling schemes for model training

Sampling Scheme	No. of Pos. Samples	LAMR (%)
All frames	65,570	56
Every 10th	6,557	56
Every 20th	3,261	55
Every 30th	2,198	54
Every 40th	1,629	55
Every 50th	1,280	56
Every 60th	1,088	58

details of the dataset and benchmark criterion, please refer to [33]. For a vehicle detection performance evaluation, we collected a new vehicle dataset based on the analysis of vehicle collision statistics [76]. The dataset was captured at 3fps following the lessons in [20] and consists of three sessions (S0~S2) for training and two sessions (S3~S4) for testing.

First, we performed a set of experiments to identify the key design parameters of the DPM for the pedestrian and vehicle class without our ROI mechanism. Secondly, using the insights learned from the first set of experiments, we quantitatively evaluate detection performance for each class of objects using the Caltech Benchmark with the ROI mechanism. Finally, for a qualitative real-world evaluation, we deploy the detection system to our experimental vehicle and demonstrate real-time detection for pedestrians and vehicles on a 43 minute sequence from various complex environments. For the first and second experiments, we used the same protocol used in [33], which upscale the input images by a factor of 2 (thus, a final resolution of 1280×960) for fair comparison. For a real deployment of our system, however, we use the original size of images (640×480) with two multiresolution models as detailed in Section 3.5-C.

3.5.1 Model Design Parameters

Number of Training Samples: The Caltech dataset contains approximately 2,300 unique pedestrians out of approximately 347,000 total instances of pedes-

Table 3.3: Different number of parts

No. of Parts	LAMR for Ped.(%)	LAMR for Veh.(%)
2	58	20
3	55	19
4	55	18
5	56	18
6	54	18
7	56	18
8	56	18

trians. This high number of bounding boxes is mainly because the dataset was recorded at 30fps. Once a pedestrian appears in the field of view, he or she tends to be captured for at least a couple of seconds. Because of this high level of redundancy in the sample images, we had to first determine the best sampling scheme to obtain a statistically valid set of training images. We trained 7 models with a standard set of parameters (1 component with 6 parts) where each model is trained with a training set (S0~S5) consisting of images selected from the dataset at a different sampling frequency. We trained models for each of the following down-sampling factor: 1, 10, 20, 30, 40, 50, and 60. We ran each model on the test set (S6~S10) and evaluated them using the same evaluation code provided by [33] (ver. 3.0.0). The results of these experiments are shown in Table 3.2, where we use log-average miss rate (LAMR) to represent detector performance by a single reference value. The LAMR is computed by averaging miss rate at nine False Positives Per Image (FPPI) rates evenly spaced in log-space in the range 10^{-2} to 10^0 according to [33]. Although there is no large difference in performance between the settings, using every 30th training images (i.e., one training image per one second) in this case gives us the best performance. We decided to use this setting for the pedestrian class throughout the remainder of our experiments. This result has implication in the generation of ground truth data as annotating every 30th frame of a dataset is far less expensive than having to annotate every frame. For the vehicle model, as mentioned before, we collected the dataset at 3fps and did not perform this experiment.

Table 3.4: Different number of scales per octave

Scales / Octave	No. of Levels	LAMR (%)
2	10	61
4	19	56
6	28	56
8	37	55
10	46	54
12	56	54
14	65	53

Number of Parts: The standard number of parts for the DPM is 6 for `voc-release3` [43]. However, the optimal number of parts indeed depends on the variability of an object class and may be significantly different between classes. Our second experiment was designed to identify the optimal number of parts required for the pedestrian and vehicle models that would be used for the automotive application. Once again, we trained 7 models with different number of parts (2~8) using each training set and tested them on each testing set. As shown in Table 3.3, for the pedestrian case, using 6 parts yields the best performance while its variance is not so large depending on the number of parts. The reason is that most of the pedestrians (84%) in the Caltech dataset are smaller than 80 pixels in height and large number of parts does not necessarily help for better detection in such low resolution images. Interestingly, the performance of vehicle models shows a very low variance on the number of parts. This seems reasonable given the fact that vehicles are basically rigid bodies and we trained a rear-view vehicle model for our FCW application. In general, when selecting the number of parts for `voc-release3`, it might be better to use smaller number of parts for a faster detection rate. However, for the star-cascade algorithm using 6 to 8 parts seems reasonable due to the cascaded structure of its classifier. To balance efficiency and accuracy, we decided to use 6 parts for our evaluations.

Number of Scales Per Octave: Typically, modern object detectors use two or three octaves and sample 8-14 scales per octave for accurate detection. Since the computational requirements for feature computation in this image pyramid

can be significant, new methods of handling this issue must be identified. Recently, Dollár et al. [31] proposed a technique to avoid constructing such a feature pyramid by approximating feature responses at certain scales by computing feature responses at a single scale. They demonstrated that such an approximation is accurate within an entire scale octave. While the method is applicable to our case, here, we are interested in primarily looking at performance differences depending on different number of scales and looking for a specific optimal solution for our configuration. We tested 7 different settings and the results are shown in Table 3.4. We found that even 4 scales per octave shows a marked improvement over 2 scales per octave in accuracy. We decided to use 4 scales per octave as a trade-off between accuracy and speed performance.

3.5.2 Quantitative Evaluation with Caltech Benchmark

In this subsection, we quantitatively evaluate the performance of our implementation of the DPM for both pedestrian and vehicle class using the Caltech benchmark. This is possible because the benchmark provides a flexible way to evaluate detectors under various conditions on multiple datasets.

3.5.2.1 Pedestrian Detection

We used the Caltech Pedestrian Dataset for training a pedestrian model. We trained a model using the training set sampled at every 30th frame and filtered by a height condition (i.e., $40 < h < 400$ pixels), resulting in 2,198 positive samples. Note that we intentionally excluded small-sized bounding boxes (i.e., $h < 40$ pixels) since our aim was training a model with 80 pixels in height. We evaluated the performance of our implementation on the Caltech testing data. Although the framework provides detection results from sixteen pre-trained detectors (as of January 2012) to help provide consistent comparison with other state-of-the-art detectors, we intentionally selected 8 of the most relevant top-performing detectors to increase the clarity of the resulting graphs. The criterion we used to select the algorithms was detection accuracy and run-time. One exception is the “MultiFtr+Motion” algorithm, which exhibits very slow detection speed (reported as more than 30 seconds for a 640×480 image in [33]). But we included the method

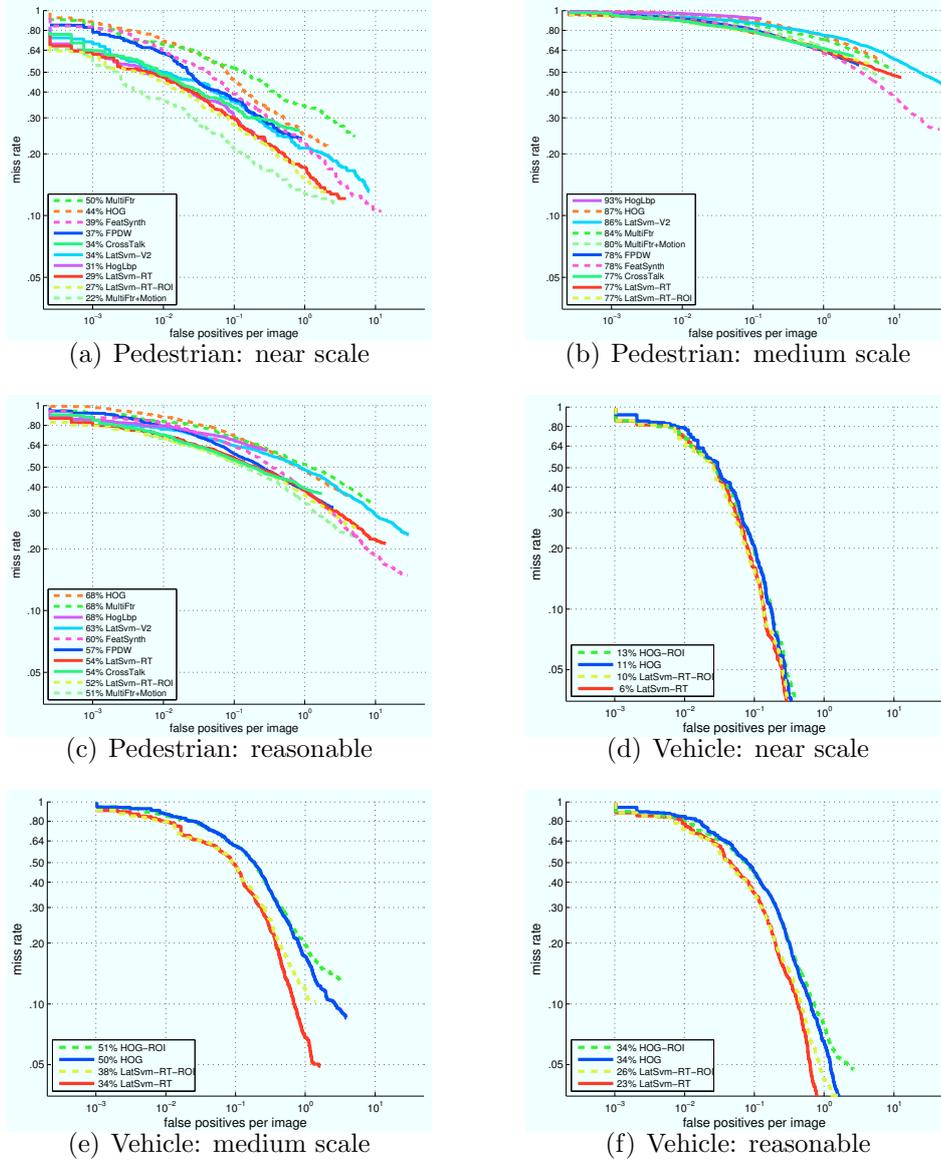


Figure 3.4: Pedestrian and vehicle detection results using the Caltech Benchmark in [33]: (a) Performance on unoccluded pedestrians over 80 pixels in height. (b) Performance on unoccluded pedestrians between 30-80 pixels in height. (c) Performance on pedestrians at least 50 pixels in height under no or partial occlusion. (d) Performance on unoccluded vehicles over 80 pixels in height. (e) Performance on unoccluded vehicles between 20-80 pixels in height. (f) Performance on vehicles at least 30 pixels in height under no or partial occlusion.

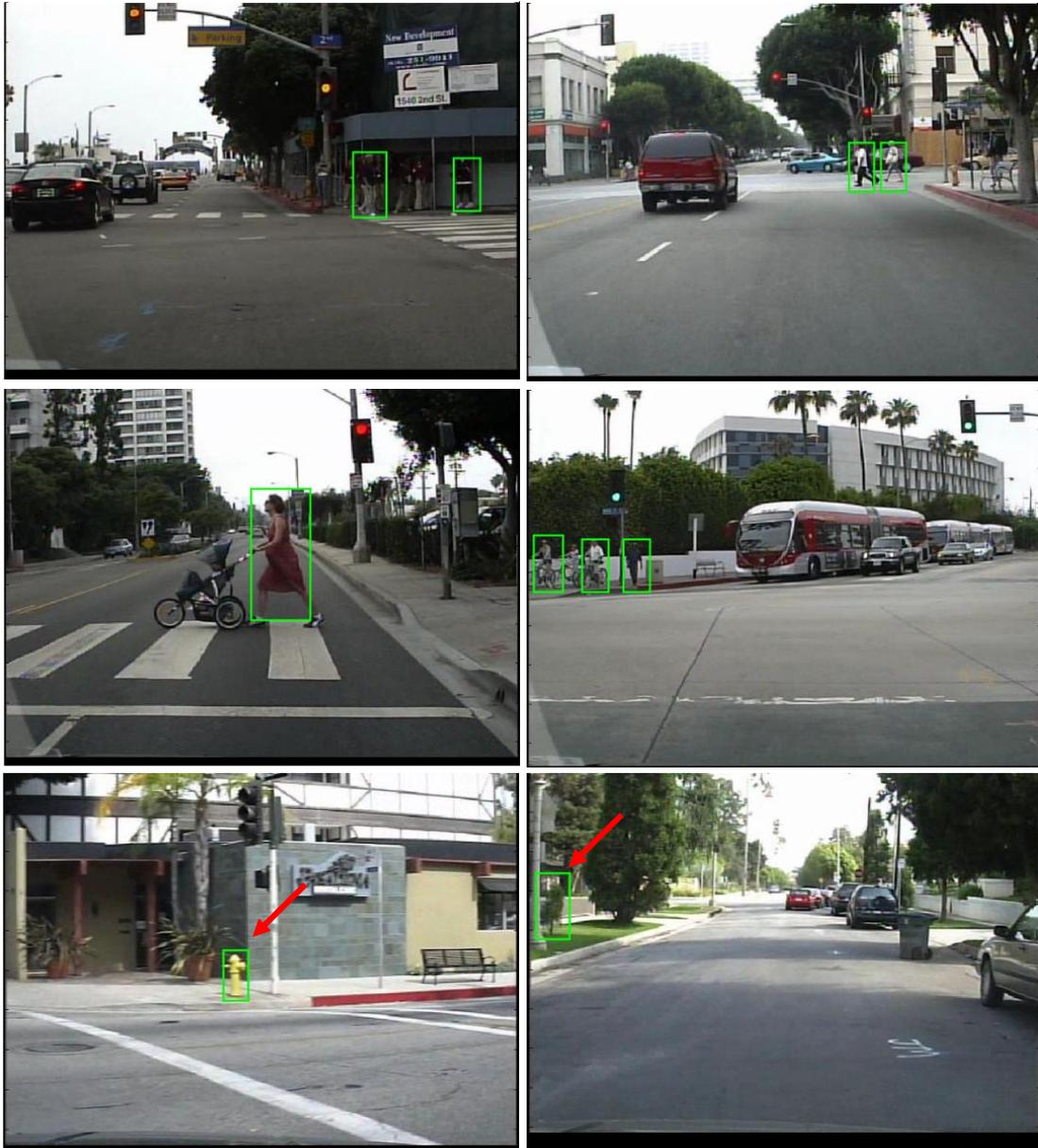


Figure 3.5: Qualitative detection results on the Caltech testset. The first and second row shows correct pedestrian detections in various scenarios. The third row shows typical false positives (indicated with red arrows).

since it is the only method which uses motion features in our list and also because it shows the best performance in most cases.

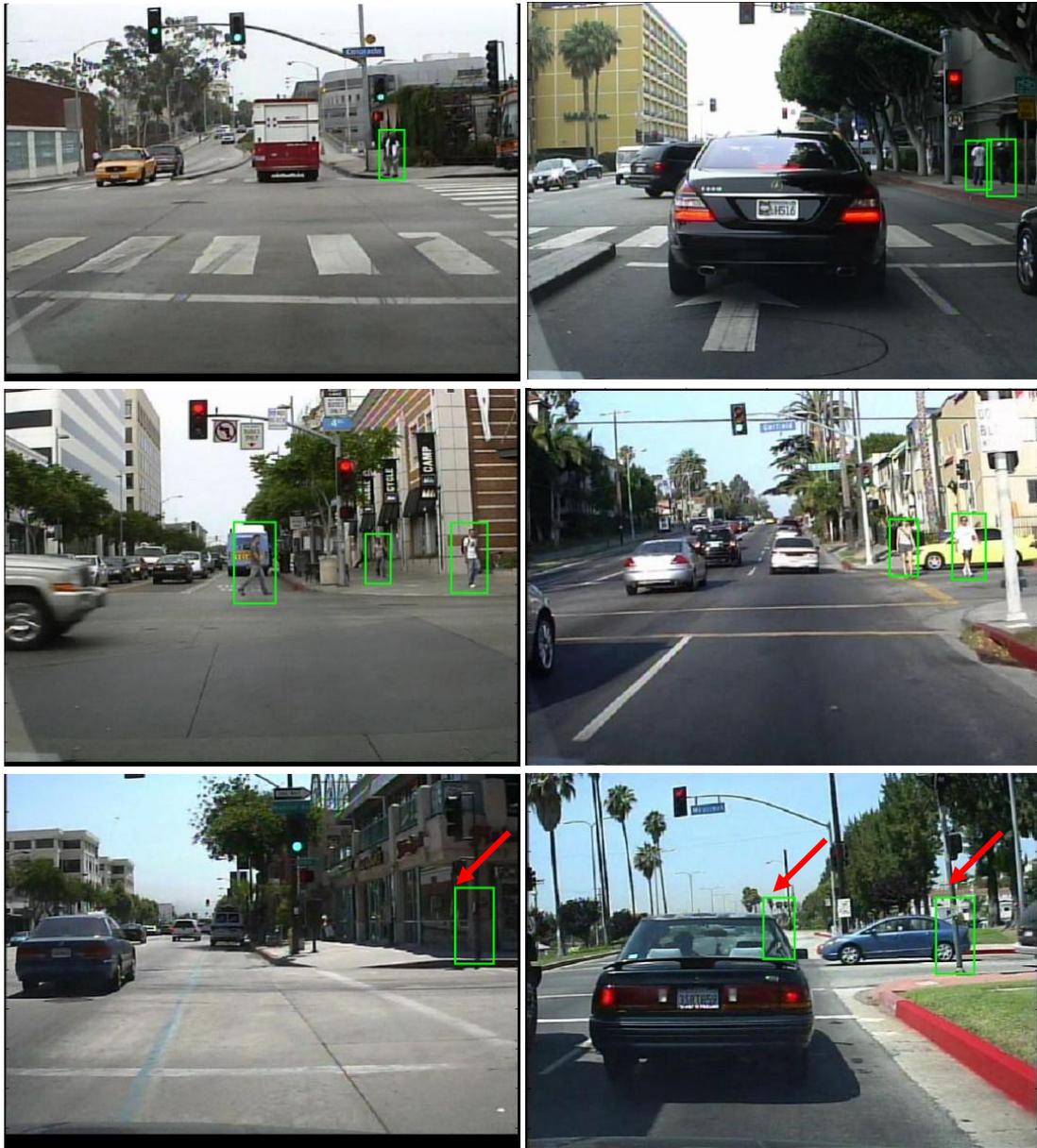


Figure 3.6: Qualitative detection results on the Caltech testset (More examples).

We named our implementation “LatSvm-RT” without our ROI algorithm as in [20] and “LatSvm-RT-ROI” with the ROI algorithm. Following the naming convention of the Caltech benchmark we described the other detectors as follows: HOG [27], HogLbp [108], MultiFtr [111], LatSvm-V2 [41], FeatSynth [13],

FPDW [31], CrossTalk [30], MultiFtr+Motion [106]. All detectors except MultiFtr+Motion [106] (which used their own dataset called ‘TUD-MP’) were trained with the “INRIA Person Dataset [27].” Note that the LatSvm-V2 is our baseline detector, `voc-release3` and our effort in this thesis is to develop its real-time implementation. Our implementation shows better performance compared to the baseline mainly because we used models trained with the Caltech training set and also the ROI algorithm.

These facts are illustrated in the results shown in Figure 3.4(a)~3.4(c). Following the example in [33], we plot miss rate vs. false positives per image (FPPI) and use the LAMR as a single comparison value for the overall performance. The entries are ordered in the legend from the worst LAMR to the best. In general, our detector shows very good performance. Results for near and medium scale unoccluded pedestrians, corresponding to heights of at least 80 pixels and 30-80 pixels, respectively, are shown in Figure 3.4(a) and Figure 3.4(b). In each case, our detector with the ROI algorithm shows the second best and the best performance with a LAMR of 27% and 72%, respectively. Figure 3.4(c) shows performance on pedestrians over 50 pixels tall under no or partial occlusion. Here our detector shows second best performance with a LAMR of 52%. Obviously, we can see that our ROI algorithm is also contributing to better detection performance. Typical qualitative results achieved on the Caltech testset are shown in Figure 3.5 and 3.6.

3.5.2.2 Vehicle Detection

We collected a new dataset (recorded at 3fps with 1280×960 resolution) for training vehicle models. We decided to record high-definition images for future usage, but for the low-resolution purpose, we annotated fully visible (rear-view) vehicles which are larger than 16×16 pixels for all sessions (S0~S4). We trained two rear view vehicle models using the training data (S0~S2), as visualized in Figure 3.11, one a normal size model for short- and normal-distance and the other one a small size model for long-distance. We use 4,484 and 558 positive samples for the normal-distance vehicle model and long-distance vehicle model, respectively. Trucks and buses are excluded for the training although they are

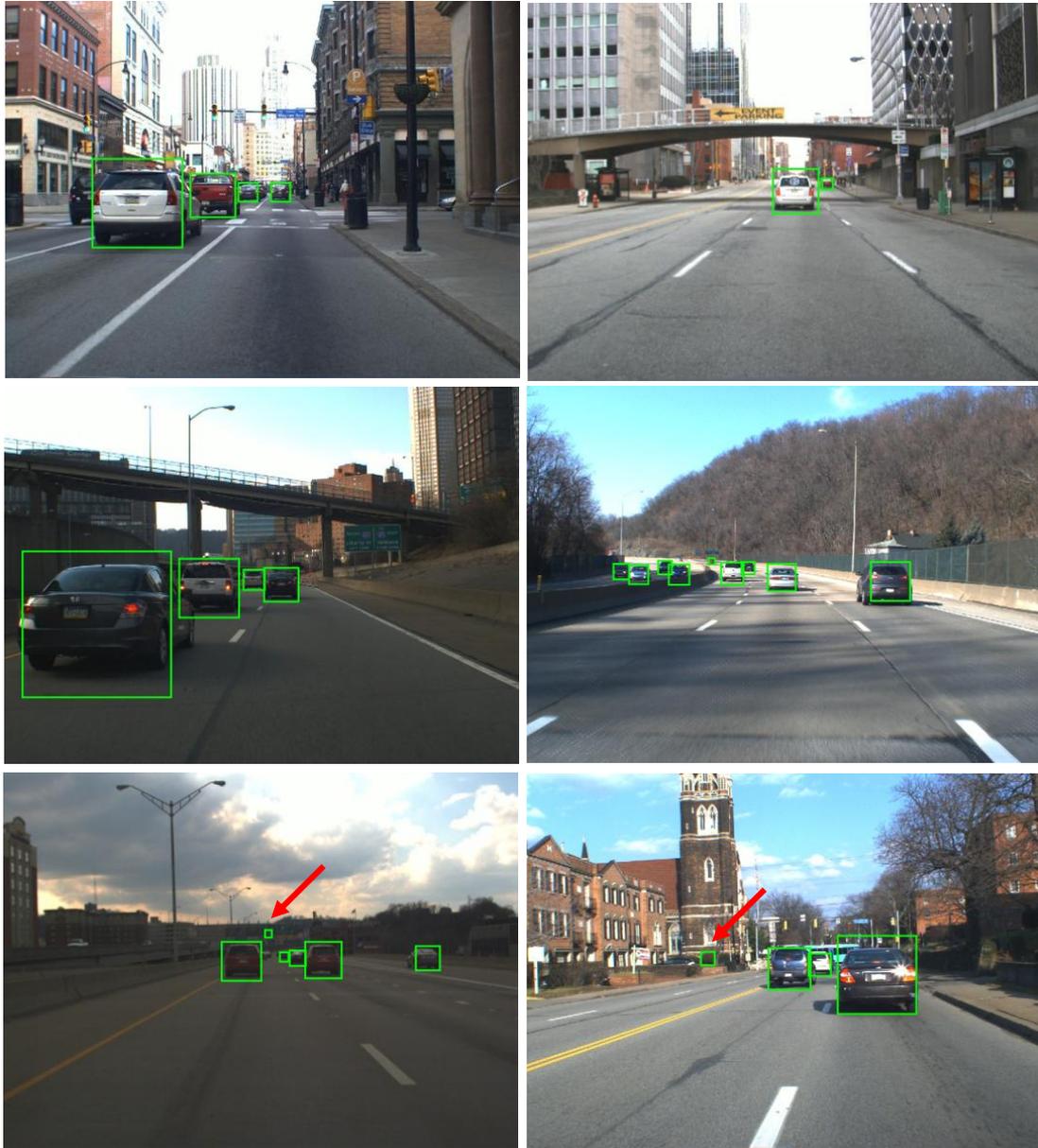


Figure 3.7: Qualitative detection results on our vehicle dataset. The first and second row shows correct vehicle detections in various scenarios. The third row shows typical false positives (indicated with red arrows).

labeled. We evaluated their performance on the testing data (S3~S4). To this end, we run both models on the entire images with (or without) the ROI algorithm

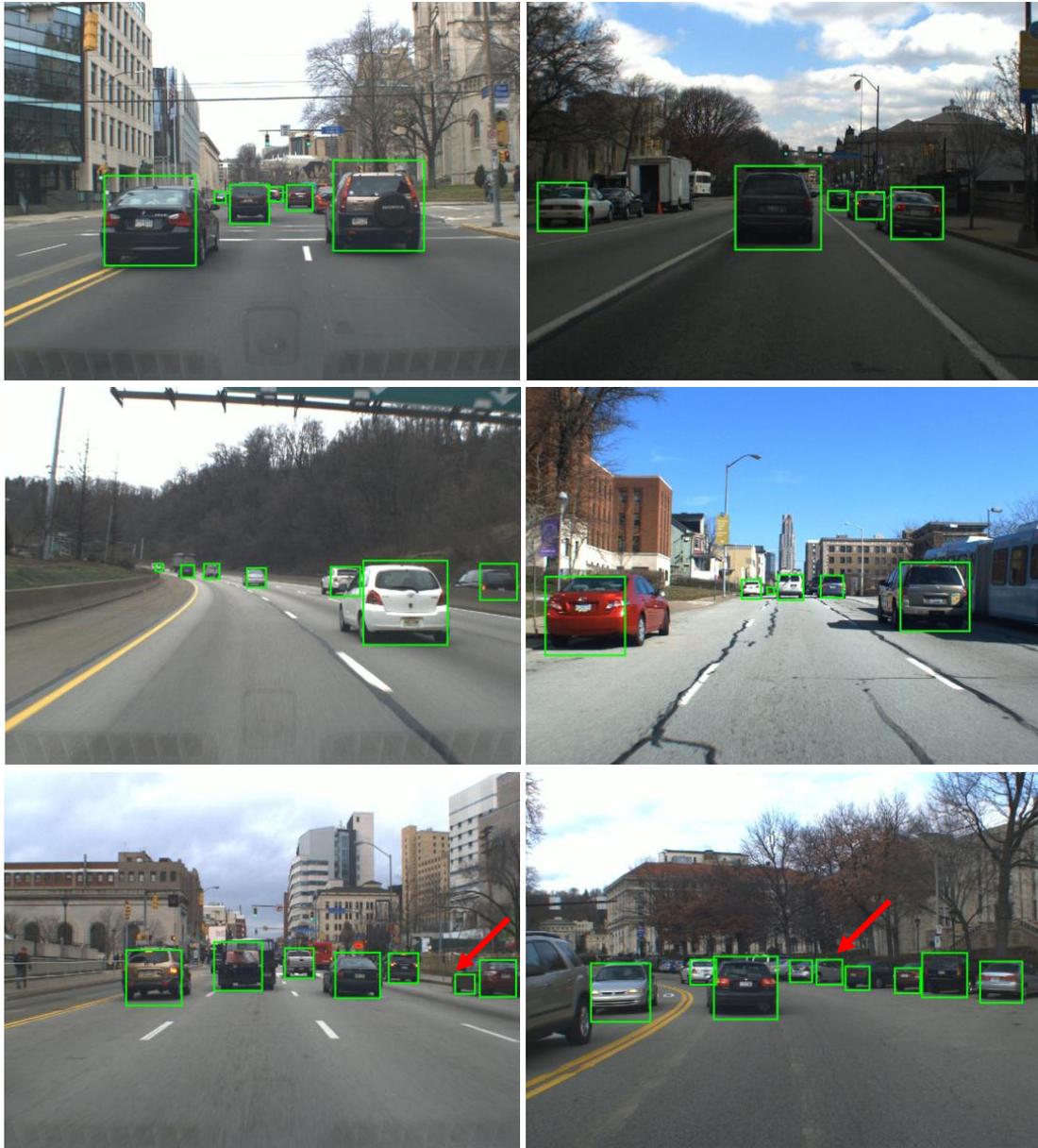


Figure 3.8: Qualitative detection results on our vehicle dataset (More examples).

and then merge detections from both detectors. For comparison, we also trained the standard HOG models using the DPM code of `voc-release3` and applied them with (or without) our ROI algorithm. For the evaluation, we used the same scenarios as the pedestrian case but with different height criteria for each

scenario. Since detecting small vehicles (in images) is important we changed the height criteria from 30~80 to 20~80 for the ‘*medium scale*’ scenario and from >50 to >30 for the ‘*reasonable*’ scenario.

As can be seen in Figure 3.4(e)~3.4(f), our vehicle models exhibit much better performance compared to the pedestrian case. For ‘*near scale*’ and ‘*reasonable*’ scenarios, our models show a perfect detection rate with 1 FPPI. We claim that this is a very meaningful result since most of the false positives in this condition come from the front view of oncoming vehicles, which might not be false positives in our applications. Furthermore, we can verify that even the standard HOG vehicle models also perform very well. We think that the results are reasonable considering that vehicles are in general rigid body objects, thus, less deformable. For vehicle models, the advantage of the ROI algorithm in terms of detection accuracy turns out to be low compared to the pedestrian case. Typical qualitative results achieved on our test set are shown in Figure 3.7 and 3.8.

3.5.3 Quantitative Evaluation of Performance with Inner Motion Features

In this subsection, we quantitatively evaluate the effect of inner motion features on pedestrian detection. For this purpose, we also used the Caltech Pedestrian Dataset. This time, however, we followed the convention used in [79] for fair comparison with them. We trained one-component deformable part-based model with a size of 56×28 and a 4×4 cell size. We use the training set sampled at every 30th frame, filtered by a height condition (i.e., $50 < h < 400$ pixels), and excluded occluded pedestrians, resulting in 1,458 positive samples. Note that because of the reduced model size with 4×4 cell size we ran our detectors on the original input images rather than $(2\times)$ upscaled images. This model gives us 43% log-average miss rate in the ‘*reasonable*’ scenario which outperforms the prior state-of-the-art method called ‘CrossTalk’ [30] by 11%. We also point out that this new trained DPM significantly improves its previous performance (specified as ‘LatSVM-V2’ in Figure 3.9(c)) by 20%, which is somewhat surprising to us. For comparison purpose, we also trained a standard HOG-SVM model [27] with the same training set using the DPM code `voc-release3` [43]. In the same scenario,

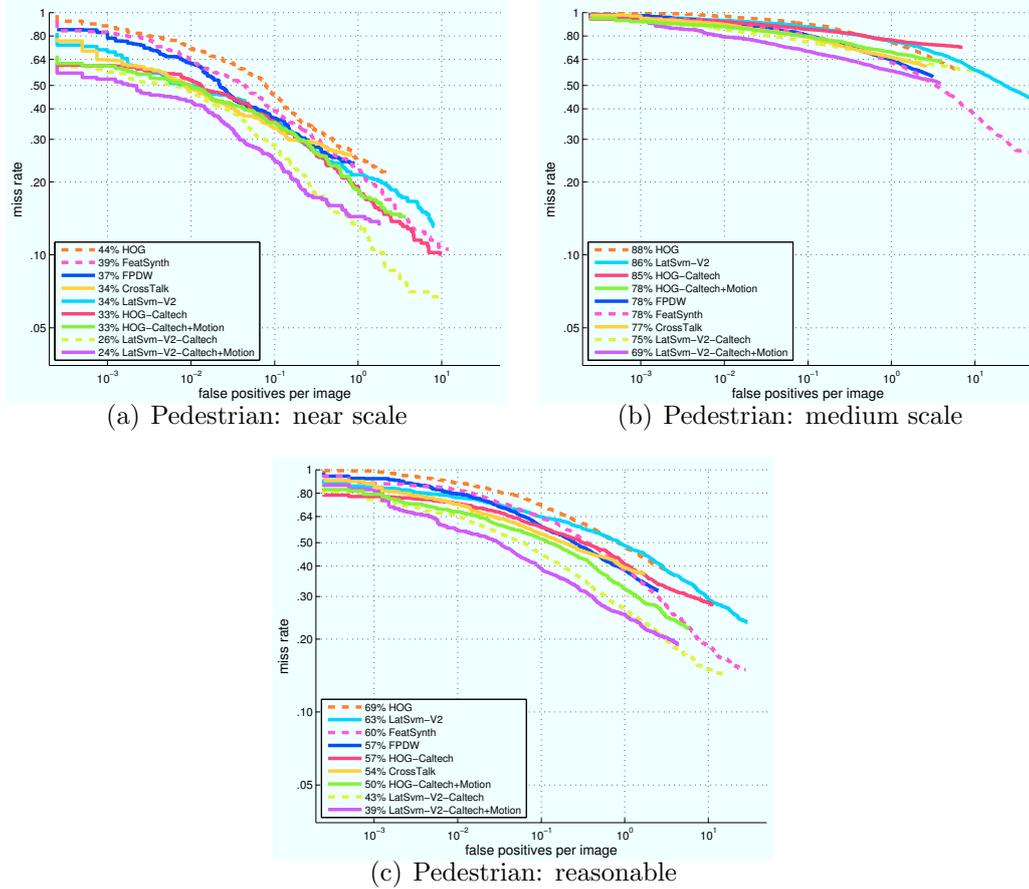


Figure 3.9: Effect of inner motion features on pedestrian detection: (a) Performance on unoccluded pedestrians over 80 pixels in height. (b) Performance on unoccluded pedestrians between 30-80 pixels in height. (c) Performance on pedestrians at least 50 pixels in height under no or partial occlusion.

this model shows 57% of log-average miss rate where the original method in [79] reported 46%. We believe that this difference is due to the fact that we only use spatial (2×2) block normalization where the original method uses spatial-temporal ($2 \times 2 \times 2$) block normalization.

For the inner motion features, we implemented the $D^{16}(8, 4)$ as we explained in section 3.3 with a set of parameters reported as the best trade-off between time complexity and performance. We use the intensities of gray scale images for computing the signed temporal gradient (original method used a Luminance channel of LUV color model). We encode inner motion features after applying

4×4 spatial aggregation and L1 block normalization. We trained a DPM and HOG-SVM model with our inner motion features using the same training set we used for the baseline models. For the HOG-SVM model case, we obtain 50% log-average miss rate and get 7% improvement over the baseline model which shows 57% log-average miss rate in the ‘*reasonable*’ scenario. For the DPM case, it shows 4% improvement from 43% to 39%. From this result, we can conclude that the inner motion features give consistent positive effect to the overall detection against possible textureless regions on images. Detailed results on ‘*near scale*’, ‘*medium scale*’, and ‘*reasonable*’ scenarios are shown in Figure 3.9(a)~3.9(c).

3.5.4 Real-Time Onboard Evaluation

All quantitative results from the previous subsection show state-of-the-art object detection performance in the computer vision field. However, there is a huge performance gap between the current performance and what is required for automotive safety applications. For example, for the pedestrian case, the false positive rate should be less than 1 per 3 hours of driving with a detection rate of above 95% [88], but our result reports a 83% detection rate around 1 FPPI for a ‘*near scale*’ case (see Figure 3.4(a)). Just considering only the false positive rate, assuming 10Hz frame rate, we will see around 108,000 false positives for 3 hours, which is far from the requirement. Does this mean that the performance of most of current state-of-the-art object detection systems is significantly insufficient for real-world applications? This subsection describes our efforts to find out the answers for this question. We integrated our object detection system into our experimental vehicle to see how it performs in challenging real-world scenarios. Our experimental vehicle is indeed a new research vehicle platform developed at Carnegie Mellon University to conduct autonomous driving research [109]. While the vehicle is designed to meet the requirements of general autonomous driving with multiple sensors such as radar, LIDAR, and cameras, here we are focusing on vision-based object detection functionalities for automotive active safety applications.



Figure 3.10: Our experimental vehicle and its sub-components. (a) Autonomous Cadillac SRX [109], (b) Forward-looking camera (monocular setup), (c) Computing cluster and control console.

3.5.4.1 System Setup

The experimental vehicle is equipped with a forward-looking camera (PointGrey FL3-GE-50S5C-C, 45° VFOV) and four mini-ITX form-factor computers (Core 2 Extreme QX9300@2.53GHz, 8GB RAM). The forward-looking camera is interfaced with computers via a Gigabit Ethernet (i.e., GigE). 640×480 resolution of images are fed into all computers at 8Hz and our detection system runs on each of two computers with a different model (i.e., pedestrian and vehicle). The vehicle and described sub-components are shown in Figure 3.10.

Model size design: For deciding a pedestrian and vehicle model size, we analyzed the object’s pixel height in the image plane, as a function of distance using the Eq. 3.7 and a known camera parameter ($f_p=557.73$). From this analysis, we trained a 96×40 pixel-sized model to detect pedestrians reliably up to 10m. For the range from 10m to 20m, we trained a 48×20 pixel-sized model with HOG cell size of 4×4 . Similarly, we trained two vehicle models, one a 48×48 sized model for normal-distance (i.e., 5m~22m) and the other a 16×16 sized model with HOG cell size of 4×4 for long distances (i.e., 22m~55m). We only trained rear-view vehicle models for efficiency reasons. All models and corresponding reasoning are shown in Figure 3.11.

Software parameters: After a large amount of unit testing, we set the optimal parameters for each module of our detection system. For detection, we trained DPMs with 6 parts using the `voc-release3` training tool [43] and then built cascade models from them for the star-cascade algorithm with the modification explained in Section 3.2.3. We run a normal size model with the

ROI method and a small size model on a pre-defined ROI. We use 4 scales per octave (i.e., $\lambda=4$), resulting in 14 image pyramid levels for 640×480 images. For the ROI algorithm, we use constant value 3 and 2 for all $e_u(j)$ and $e_l(j)$, respectively for implementation convenience. We also use the following definitions: $OH_{ped}=1.7m \pm 0.3m$ and $OH_{veh}=2.0m \pm 0.5m$. For the range estimation module, finally, we use the following parameter set a) $w_{hg}=0.7$, $w_{BBH}=0.15$, $w_{BBW}=0.15$, $OH_{veh}=1.5m$, and $OW_{veh}=1.9m$ for the vehicle model for short- and normal-distance and b) $w_{hg}=0.5$, $w_{BBH}=0.2$, $w_{BBW}=0.3$, $OH_{veh}=2.0m$, and $OW_{veh}=2.0m$ for the vehicle model for long-distance and finally c) $w_{hg}=0.7$, $w_{BBH}=0.15$, $w_{BBW}=0.15$, $OH_{ped}=1.7m$, and $OW_{ped}=0.5m$ for the pedestrian model. The rationale for these parameters results from data statistics analysis on our dataset.

3.5.4.2 Range Estimation

Accurate range estimation from a detected bounding box is the most important final step for automotive applications. With sufficiently accurate performance, it could be used as an input into an Adaptive Cruise Control (ACC) system as well as an input to the Forward Collision Warning (FCW) system. In general, it is difficult to get an accurate range estimation from a monocular camera setting. Usually, we can obtain a rough range estimation from Eq. 3.7 with some assumptions (e.g., flat ground and accurate localization of bounding boxes). We evaluated these approaches and determined that they could work reasonably well for pedestrian models due to shorter working range but they are too error prone for the vehicle model due to the longer distance at which the vehicles must be detected (i.e., 55m vs. 20m) and error sensitivity from various violations of above assumptions is too severe for distant vehicles (i.e., above 40m). To address this problem, we propose a new formula for better range estimation which considers a bounding box size as well as its bottom position. From the camera geometry illustrated in Figure 3.2, the relationship between a bounding box size (i.e., width or height) and distance (d) can be easily obtained. For verification of this geometry model, we empirically analyzed a number of labeled bounding boxes of various object classes. A case study for an SUV (Sports Utility Vehicle) class is

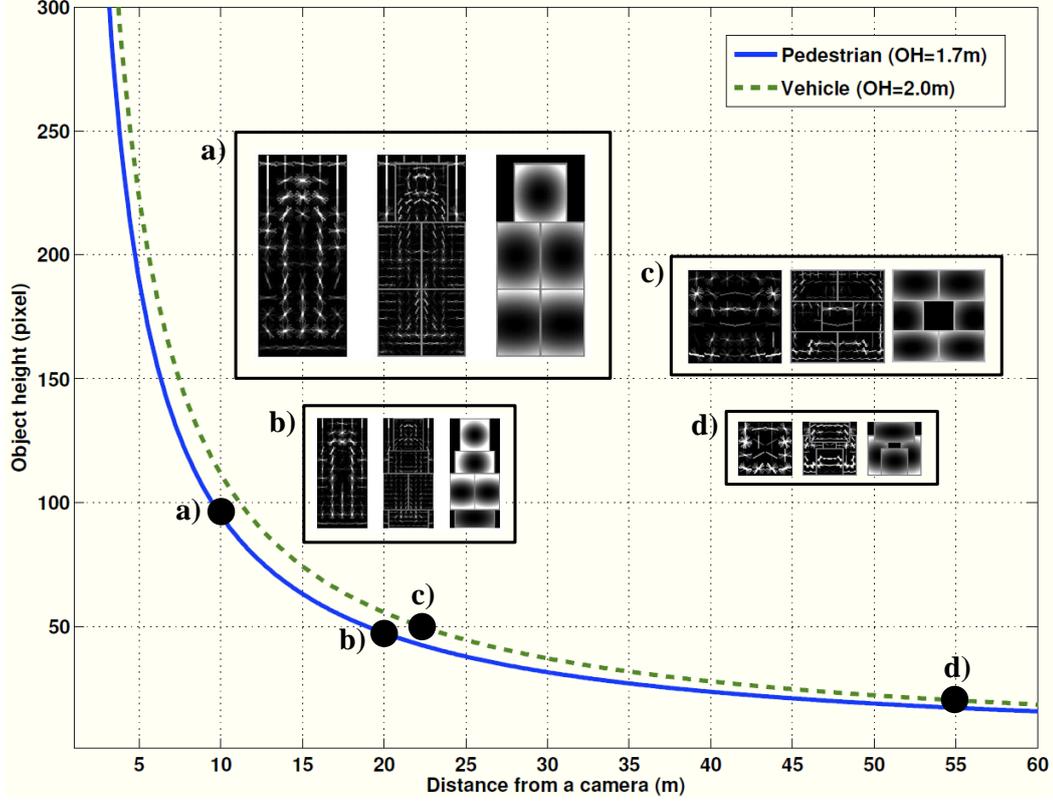


Figure 3.11: Plot of \mathbf{ROI}_y as a function of d (Eq. 3.7). Based on this analysis, all models are designed and visualized here. (a) Normal-sized pedestrian model (96×40 with 8×8 HOG cell). (b) Small-sized pedestrian model (48×20 with 4×4 HOG cell). (c) Normal-sized vehicle model (48×48 with 8×8 HOG cell). (d) Small-sized vehicle model (16×16 with 4×4 HOG cell).

shown in Figure 3.12(a) and 3.12(b). Final estimate is the convex combination of those three separate estimates. From this, we have a new formula for a range estimation (d):

$$d = w_{h_g} \left(\frac{Y_c \times f_p}{h_g} \right) + w_{BB_H} \left(\frac{OH \times f_p}{BB_H} \right) + w_{BB_W} \left(\frac{OW \times f_p}{BB_W} \right) \quad (3.12)$$

where w_{h_g} , w_{BB_H} , and w_{BB_W} are weighting factors for camera geometry, height and width of bounding box terms, respectively and $w_{h_g} + w_{BB_H} + w_{BB_W} = 1$. OH and OW are the physical height and the width of an object, respectively.

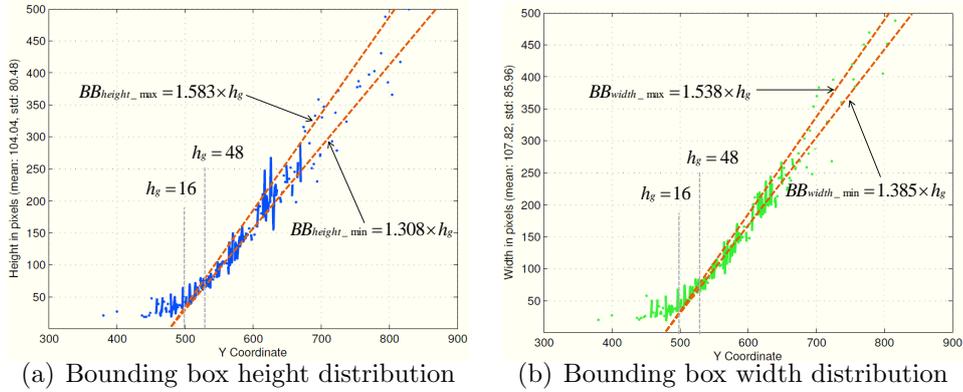


Figure 3.12: Empirical distribution of bounding box sizes for an SUV case. This statistics is calculated from 9,481 labeled bounding boxes of the SUV class. Measurements for (a) height and (b) width are well fit into the geometry model. (SUV width max : 2.0m, SUV width min : 1.8m, SUV height max : 2.0m, SUV height min : 1.7m, Camera height : 1.3m)

Class	Detection Rate (%)	False Positives/hour
Pedestrian	91%	25
Vehicle	94%	14

Table 3.5: System level detection performance of the proposed system.

3.5.4.3 Detection Results

To obtain a realistic performance assessment, we drove our experimental vehicle from the Cranberry Township to Pittsburgh international airport. This course involves various challenging environments such as urban streets and highways for vehicle detection and busy pedestrian zones at the airport departure area for pedestrian detection. This test sequence corresponds to 43 minutes of daytime driving in various places under normal weather conditions. In general, our system shows a promising performance for pedestrian detection and a remarkable performance for vehicle detection except in some extreme cases such as significant egomotion and dynamic scene change. We manually evaluated the performance for each model. However, for the quantitative performance evaluation, it is required to manually label each of the frames in the entire data set. Because this is

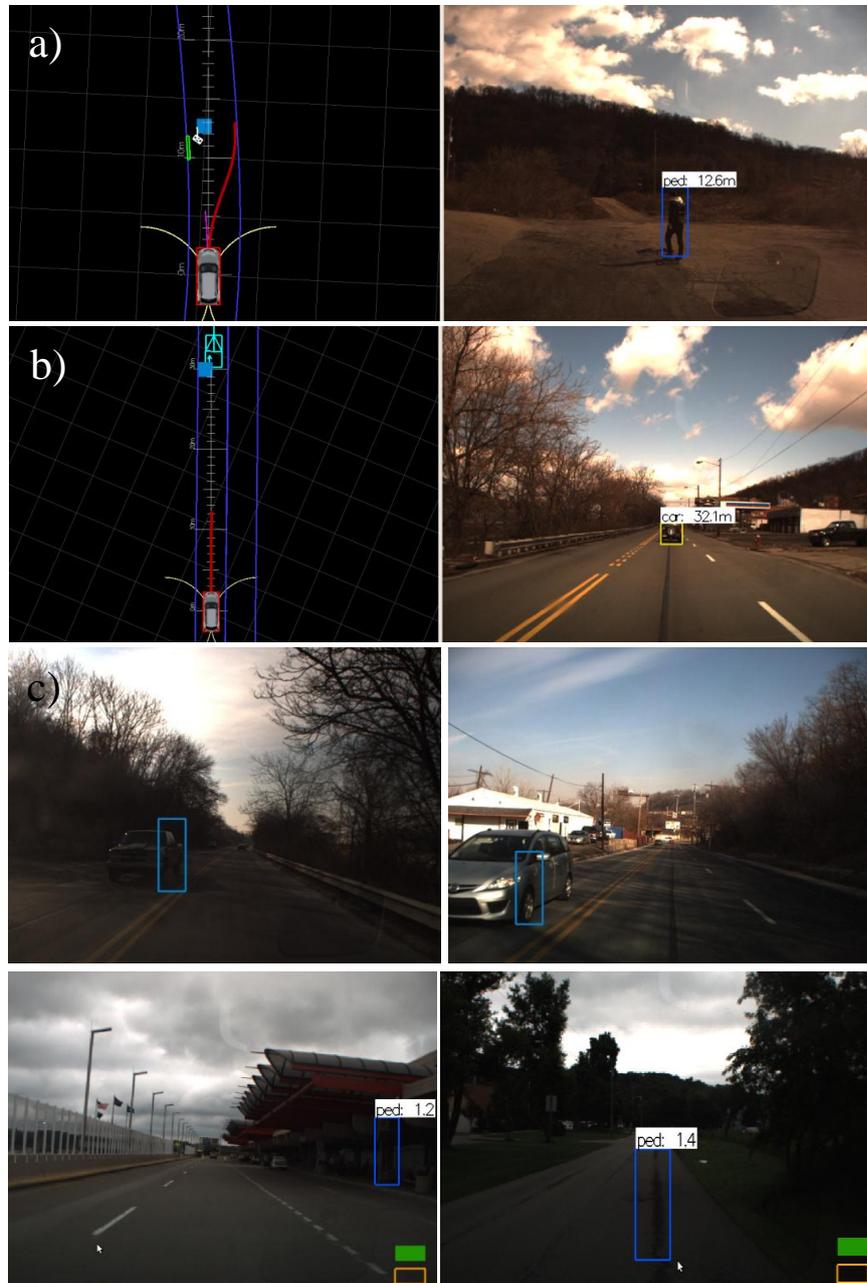


Figure 3.13: Typical snapshots from the online operation of our on-board pedestrian and vehicle detection system: (a) PPS and (b) FCW on our autonomous vehicle. (c) shows typical false positives for pedestrians (blue: pedestrians, yellow: vehicles, labeling: (class: distance in meter)).



Figure 3.14: Typical snapshots from the online operation of our on-board pedestrian and vehicle detection system: (c) and (d) show pedestrian and vehicle detection results, respectively, in various scenarios. The purple horizontal line represents the detected vanishing line. The green circle represents the location of the vanishing point tracked over frames. All green lines are the ones extracted from a perspective input image.

labor-intensive and error-prone, we evaluated the performance differently. In particular, we had human annotators, using our visualization tool, go over the data second-by-second. Following the scheme used in [88], we counted the number of correctly (and incorrectly) detected bounding boxes. When we evaluate detection rate, we only consider inward moving and in-path stationary pedestrians up to 20m for pedestrian detection. For vehicle detection, we only consider vehicles on the current, left, and right lane up to 55m with respect to the ego-vehicle. In addition, since our vehicle models are designed specifically for Sedan and SUV types and not for trucks and buses, we evaluated the detection performance without considering truck and bus samples. The manual evaluation of our system, as shown in the Table 3.5, indicates that we achieve 91% detection rate with 25 false positives per hour for a pedestrian case, 94% detection rate without truck and bus samples with 14 false positives per hour for a vehicle case. We believe that this is quite remarkable performance given the fact that it is achieved from only a detection system, i.e., single-frame classification. It is well known that a reasonable performance for either PCW or FCW systems can be achieved via a system level integration of a detection system together with other sub-modules such as tracking and multi-frame verification [49], [88]. Putting all these facts together, we claim that the performance level of state-of-the-art object detection system indeed is getting closer to a level of field requirement.

With this performance, we demonstrated pedestrian avoidance and forward collision warning functionalities as shown in Figure 3.13 (a) and (b). Typical false positives are shown in Figure 3.13 (c). The most frequent false positive for a pedestrian class was from leading or oncoming vehicle’s boundary, which can be alleviated by looking at the vehicle detection results or by adding motion features [79]. For a vehicle class, rectangular-shaped road traffic signs were primary source. Figure 3.14 (d) and (e) show typical detection results for pedestrians and vehicles from our system, respectively. Training and test data for our vehicle model as well as all result videos are available on our project website¹.

¹<http://users.ece.cmu.edu/~hyunggic/inAction.html>

3.6 Summary

This chapter presented a well-integrated on-board object detection system intended for self-driving vehicles as well as for potential use in automotive active safety applications. Besides a real-time implementation of an object detection method, other missing pieces (i.e., ROI computation and range estimation modules) are integrated for the final consistent performance. First, we engineered one of the top performing state-of-the-art object detection method called ‘deformable part-based model’ (DPM) for real-time operation. Secondly, to further speed up the detection process we introduced a simple, but powerful ROI computation algorithm assisted by a vanishing point tracking. Third, quantitative evaluation on two object categories (i.e., pedestrians and vehicles) was conducted using the Caltech Pedestrian Dataset as well as the vehicle dataset we collected. Finally, with the proposed system, we qualitatively demonstrated its feasibility for automotive applications by driving our experimental vehicle through various challenging real-world scenarios for pedestrians and vehicles. Although the results we obtained from the qualitative evaluation still are not reaching a level of so-called “field requirement”, we believe that our work represents meaningful progress of on-board object detection system for autonomous vehicles.

Chapter 4

Vision-Based Multi-Object Tracking using a Rao-Blackwellized Particle Filter

Previous chapter presented our onboard real-time object detection system based on the state-of-the-art object detector called DPM. Based on reliable detection from that system, this chapter develops a monocular vision based multi-object tracking framework for autonomous vehicles based on a tracking method using a Rao-Blackwellized Monte Carlo Data Association (RBMCD) algorithm. Section 4.1 provides an overview of the tracking problem of urban objects and main contributions of the proposed system. Section 4.2 discusses how we treat the detection results from the DPM detector as a set of measurements for object tracking. Technical details for single object tracking and its extension to multi-object tracking using the RBMCD algorithm are described in Section 4.3 and Section 4.4, respectively. We describe experimental results using the framework in Section 4.5 and conclude in Section 4.6.

4.1 Introduction

The automotive industry is increasingly interested in adding more intelligence to cars and trucks with the ultimate goal of developing fully autonomous automo-

bile traffic [17, 107]. To this end one of the most important research areas to address is that of automated perception systems that will allow the vehicle to perceive its immediate environment and make decisions that enhance the safety of vehicle occupants as well as the safety of persons around it [45, 63]. Although such a perception system that can, in real time, gather enough information to do a complete scene analysis is our ultimate goal to pursue using a multi-sensor approach (chapter 5 and chapter 6), in this chapter, we focus on the problem of identifying and extracting specific aspects of interest in the scene. In particular, we are interested in the problem of detecting and tracking major traffic participants such as other vehicles as well as the class of objects called vulnerable road users (VRUs) [45] which includes bicyclists and pedestrians.

In general, bicyclists and pedestrians are the most vulnerable of the class of VRUs due to the lack of any real protection against collisions. Especially, bicyclists move at speeds similar to a slow moving vehicle and, by law, must share the road with vehicles in most urban environments [74]. This puts them at particular risk for suffering life-threatening accidents. However, pedestrians and bicyclists are particularly challenging for an on-board vision system to track due to the fact that pedestrians are highly non-rigid objects and bicyclists travel at higher speeds in close proximity to vehicles. In addition, tracking vehicles is relatively easier than tracking pedestrians since vehicles are quite rigid-body objects. However, tracking vehicles is also challenging because vehicles come in many different shapes depending on their sub-categories, e.g., sedan, SUV, truck, and bus. High variation in shape leads to difficulty in reliable object detection and therefore, high risk of tracking failure. On top of this, the need for tracking multiple instances of different object classes poses additional challenges to the problem. In the case of pedestrians, up to 15 pedestrians, depending on the complexity of the scene, can appear in the image space and need to be tracked reliably.

To overcome these challenges of automotive vision systems, this thesis focuses on a) building a general and powerful object detector which considers the deformation of object parts (previous chapter) and b) developing a multi-object tracking method using a Rao-Blackwellized Particle Filter (current chapter). Specifically, for the first part, we make use of our real-time C implementation [20, 24] of the

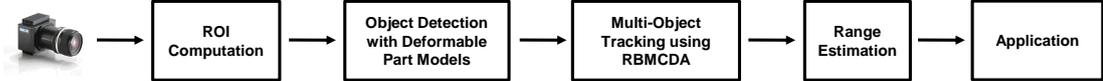


Figure 4.1: System block diagram for our vision-based multi-object detection and tracking system.

‘deformable part-based models’ (DPM) [41, 40] for reliable detection of vehicles, pedestrians, and bicyclists. For the second part, we apply the Rao-Blackwellized Monte Carlo Data Association (RBMCD) algorithm [98] to a vision-based multi-object tracking problem with an Interacting Multiple Model (IMM) filter [22]. In this chapter, we focus on the second part (i.e., tracking part) of the problem and improve our previous work. New contributions of this chapter are as follows.

The first contribution is that given three major object models (i.e., vehicle, pedestrian, and bicyclist) trained, we exploit the unique characteristics of each object’s motion. Clearly, each of the three objects has its own motion kinematics and constraints. For example, a pedestrian can move in any directions whereas the motions of a vehicle or bicycle is confined by non-holonomic constraint. Therefore, we choose two motion models: a constant velocity (CV) model [15] for pedestrians and a simplified bicycle model [54] for vehicles and bicyclists. Given the object semantic information from the detectors, a corresponding motion model is selectively chosen. For each motion model, an extended Kalman filter (EKF) is used to estimate the position and orientation of an object in the vehicle coordinates, hence 3D localization is possible via a proper uncertainty propagation (see Section 4.3).

The second contribution is the extension of the single object tracking method to that of multi-object tracking by incorporating a Rao-Blackwellized Particle Filter which runs a particle filter for data association and an EKF for each object tracking. The original theory for this framework was developed by Särkkä et al. and named as ‘RBMCD’ [98]. We applied the algorithm to the problem of vision-based object tracking by improving the measurement likelihood computation which takes advantage of the rich appearance information in images.

The final contribution of this chapter is a quantitative evaluation of our tracking framework using a challenging real world dataset. We collected new urban

object (e.g., vehicle, pedestrian, and bicyclist) datasets based on the accident statistics [74, 75]. For each object class, tracking performance for single object was analyzed in terms of position and velocity estimation accuracy and the performance of data association algorithm was evaluated qualitatively on multiple video sequences. Figure 4.1 shows a system block diagram of our design.

4.2 DPM as a Virtual Sensor

Multi-object tracking from a moving vehicle is generally a challenging task especially when using a single video camera as its sole sensor. Given a sequence of images, continuous and reliable object detection is challenging due to several facts including, for example, that a pedestrian can exhibit large amount of articulation, a bicycle can present dramatic appearance changes according to camera viewpoint, and finally, vehicles can display high intra-class variability (e.g., sedan vs. truck). The common solution for the problem of bicycle and vehicle cases is to build multiple view-based detectors to overcome dramatic appearance changes and the problem of a pedestrian case can be addressed by training a part-based model. Rather than trying to capture a global pattern of an object with one global template, part-based models focus on parts of an object and, in consequence, provide more flexible and robust representations.

The physical sensor we are using is a monocular video camera which generates a sequence of images at a certain rate. At the arrival of a new image from the camera, our object detectors are applied and generate a set of bounding boxes if there are objects of interest. Thus, from a tracking perspective, the object detectors are considered as virtual sensors which generate a sequence of measurements. The measurement set at time step k can be expressed by:

$$\begin{aligned} \mathbf{O}_k &= [\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_n] \\ \mathbf{d}_i &= [u \ v \ w \ h \ c]^T \quad i = 1, \dots, n \end{aligned} \tag{4.1}$$

where \mathbf{d}_i indicates a detected bounding box (u, v, w, h, c) , where $(u, v)^T$ is the center of the bottom line (shown as a red and green dot in Figure 4.2) and $(w, h)^T$ are the width and height of the bounding box. c indicates a category of

the detected object. This measurement set is fed into the update process of a Bayesian filter.

4.3 Single Object Tracking

With a set of measurements (i.e., bounding boxes) from our object detector, a well-defined tracking framework should be used to fuse the information from an object’s motion model and real measurements. Because the DPM detector is a time demanding module, we are interested in incorporating an algorithm with lower complexity and providing some indication about its uncertainty for tracking. For this reason, we chose to apply an extended Kalman filter (EKF) to our framework. Specifically, we employ two motion models to better describe the motion of a detected object. In addition, as a measurement model, a standard pinhole camera model is linearized with a flat ground assumption and used in the EKF update process. As shown in Figure 4.2, core idea of our tracking method is that we assume an object can be seen as a point mass moving on the $x - z$ plane in the vehicle coordinates. Another key fact is that we track the relative motion of objects in the vehicle coordinate system and thus, state variables are defined in the same coordinate system. We discuss technical details of both a motion model set and a measurement model in the next subsections.

4.3.1 Motion Models

As discussed in Section 4.1, some object classes such as bicycles and vehicles have their own unique kinematics. Thus, at a first glance, it seems natural to use an object’s kinematics to model the real motion of the object accurately. However, it becomes a fuzzy situation once we consider the fact that the measurement in our case is a rough bounding box in the image space. From the sequence of these measurements, estimating all state variables (e.g., yaw angle and yaw rate) of a complex model might be challenging. We have carried out a comprehensive set of experiments to find the best solution for this issue. From our previous work, we have learned that a bicycle can be seen as a moving mass and can be tracked reasonably well using a constant velocity model in [21] and then exploited a more

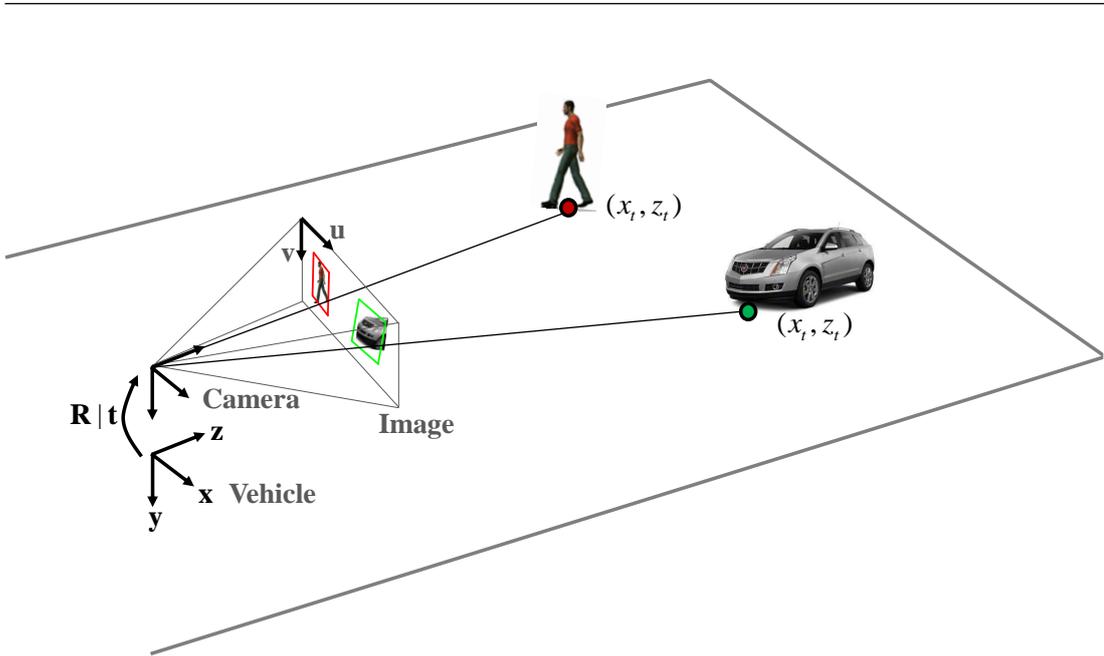


Figure 4.2: Illustration of our vision-based single object tracking process.

complicated motion model based on simplified bicycle kinematics and improved tracking performance by fusing the information from the two motion models [22]. For this purpose, we used a well-known Interacting Multiple Model (IMM) filter. In this work, we no longer use the IMM filter for fusion of two motion models, instead, we use a suitable motion model for each class of objects. Specifically, we employ a constant velocity (CV) model [15] for pedestrians and a simplified bicycle (SB) model [54] for vehicles and bicyclists. Since both motion models belong to a class of point models and a 2D bounding box in the image space is used as a measurement, we chose to use a midpoint of the bottom line of the 2D bounding box as the representative point. Based on a flat ground assumption, the point can move according to its dynamics only in the ground plane (i.e., $x - z$ plane).

First, in a CV model, the state of this moving point at time step k is expressed as:

$$\mathbf{x}_k = [x_k \quad z_k \quad \dot{x}_k \quad \dot{z}_k]^T \quad (4.2)$$

and the continuous-time state equation for this CV model [15] can be modeled

as a linear, time-invariant system:

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \mathbf{w}(t) \quad (4.3)$$

where $\mathbf{w}(t)$ is a continuous-time white Gaussian noise process with a power spectral density \mathbf{Q}_{cv} . A discrete model of this state-space equation is used for the Kalman filter.

Secondly, in a SB model, the state of the moving point is expressed by:

$$\mathbf{x}_k = [x_k \quad z_k \quad \psi_k \quad v_k \quad \omega_k \quad a_k]^T \quad (4.4)$$

where ψ_k , v_k , ω_k , and a_k are the yaw angle, forward velocity, yaw rate, and acceleration, respectively. The orientation of the forward velocity and acceleration vectors are defined with respect to the yaw angle [20]. The continuous-time state equation for this SB model [54] assuming a constant acceleration and constant yaw rate is given by:

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} v \cos(\psi) \\ v \sin(\psi) \\ \omega \\ a \\ 0 \\ 0 \end{bmatrix} \quad (4.5)$$

A discrete model of this state-space equation can be obtained by integration of the upper differential equation over one sampling period T and expressed by:

$$\begin{aligned} \hat{\mathbf{x}}_{k|k-1} &= \mathbf{f}[\hat{\mathbf{x}}_{k-1|k-1}] \\ &= \hat{\mathbf{x}}_{k-1|k-1} + \int_0^T \dot{\mathbf{x}}(\tau) d\tau \end{aligned} \quad (4.6)$$

The expression $\int_0^T \dot{\mathbf{x}}(\tau) d\tau$ in Equation 4.6 is represented in matrix form as:

$$\begin{bmatrix} \frac{v+aT}{\omega} SW + \frac{a}{\omega^2} CW - \frac{v}{\omega} \sin(\psi) - \frac{a}{\omega^2} \cos(\psi) \\ -\frac{v+aT}{\omega} CW + \frac{a}{\omega^2} SW + \frac{v}{\omega} \cos(\psi) - \frac{a}{\omega^2} \sin(\psi) \\ \omega T \\ aT \\ 0 \\ 0 \end{bmatrix} \quad (4.7)$$

where $SW = \sin(\psi + \omega T)$ and $CW = \cos(\psi + \omega T)$. The noise covariance matrix \mathbf{Q}_{sb} of this discrete-time process can be computed using the direct discrete method [15] as $\Gamma D Q_c D^T \Gamma^T$ where the noise gain matrix Γ is the Jacobian of Equation 4.6, Q_c is the noise covariance matrix for the continuous-time process, and D is a mapping matrix.

4.3.2 Measurement Model

Since a measurement is a set of bounding boxes in the image space and the tracking process itself is executed in the state-space (i.e., in the vehicle coordinate), a measurement model should be able to map the state variable \mathbf{x}_k into its measurement space (i.e., in the image coordinate). To facilitate this mapping, we use only one representative point on the bounding box, which is the center (i.e., $(u, v)^T$) of the bottom line of a 2D bounding box as the measurement. The measurement at time step k in the context of EKF can be expressed by:

$$\mathbf{z}_k = [u_k \ v_k]^T \quad (4.8)$$

Then, in general, the nonlinear mapping of the state space into the measurement space is given by:

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k, k) + \mathbf{v}_k \quad (4.9)$$

where \mathbf{v}_k is the measurement noise at time step k and can be determined by analyzing the statistics of detection results empirically. The nonlinear mapping

of the state space into the measurement space of the video camera is given by a perspective projection equation:

$$\begin{aligned}
\begin{bmatrix} u_k \\ v_k \\ 1 \end{bmatrix} &= \begin{bmatrix} f_u & 0 & u_c \\ 0 & f_v & v_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R} & | & \mathbf{t} \end{bmatrix} \begin{bmatrix} x_k \\ 0 \\ z_k \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \end{bmatrix} \begin{bmatrix} x_k \\ 0 \\ z_k \\ 1 \end{bmatrix} \tag{4.10}
\end{aligned}$$

where f_u and f_v are the focal lengths in pixels in terms of u and v direction, respectively, and (u_c, v_c) is the optical center of a camera. \mathbf{R} is a rotation matrix and \mathbf{t} is a translation vector for extrinsic parameters. The parameters P_{ij} are the corresponding entries of the final perspective projection matrix. Note that we use a flat ground assumption (i.e., $y_k = 0$). Then the measurement function \mathbf{h} is expressed by:

$$\begin{bmatrix} \frac{P_{11}x_k + P_{13}z_k + P_{14}}{P_{31}x_k + P_{33}z_k + P_{34}} & \frac{P_{21}x_k + P_{23}z_k + P_{24}}{P_{31}x_k + P_{33}z_k + P_{34}} \end{bmatrix}^T \tag{4.11}$$

Corresponding Jacobian, i.e., $(\frac{\partial \mathbf{h}}{\partial \mathbf{x}})$ should be properly computed and used in the EKF update process.

We also need to compute its inverse function and a corresponding covariance matrix to initialize a new track with a detected 2D bounding box. Given a measurement of $[u_k \ v_k]^T$ we can obtain $[x_k \ z_k]^T$ by solving the linear system $\mathbf{A} [x_k \ z_k]^T = \mathbf{b}$ with:

$$\mathbf{A} = \begin{bmatrix} u_k P_{31} - P_{11} & u_k P_{33} - P_{13} \\ v_k P_{31} - P_{21} & v_k P_{33} - P_{23} \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} P_{14} - u_k P_{34} \\ P_{24} - v_k P_{34} \end{bmatrix}$$

The covariance of $[x_k \ z_k]^T$ can be approximated using error propagation [48]:

$$\Sigma_{(\mathbf{x}_k, \mathbf{z}_k)} = \mathbf{J} \text{diag}(\sigma_u^2, \sigma_v^2) \mathbf{J}^T \quad (4.12)$$

where the Jacobian \mathbf{J} is given by:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial}{\partial u} \mathbf{A}^{-1} \mathbf{b} & \frac{\partial}{\partial v} \mathbf{A}^{-1} \mathbf{b} \end{bmatrix} \quad (4.13)$$

with $\partial(\mathbf{A}^{-1} \mathbf{b}) = \mathbf{A}^{-1}(\partial \mathbf{b} - \partial \mathbf{A} \mathbf{A}^{-1} \mathbf{b})$ [48].

4.4 Extension to Multi-Object Tracking using RBMCDA Algorithm

In the vision-based single object tracking method discussed in the previous section, we assumed correct data association between a measurement and an already initialized track. This method can be extended to a multiple object tracking framework by incorporating a proper data association technique. In this thesis, we attempt to achieve the goal by using a Rao-Blackwellized Particle Filter (RBPF) [34, 86]. Since we have multiple measurements and no information about how many objects exist at a certain time step, the scope of the problem is quite broad, including solving a data association problem, estimating the number of objects, and tracking each object. The RBPF framework provides a mathematical tool to handle this daunting task. The core idea of the RBPF is to break down a large state estimation problem into two smaller problems, one that can be solved by an analytical solution and the other one that can be solved by a particle filter, hence providing better results than what could be obtained from a pure particle filter solution.

Särkkä et al. [98] applied an RBPF framework to a multi-target tracking problem and named their algorithm Rao-Blackwellized Monte Carlo Data Association (RBMCD). The algorithm relies on a Bayesian factorization to separate the posterior into two parts: 1) an estimation of the number of objects and data association problem and 2) a single target tracking problem. The RBPF solves

the first problem via a particle filter and then, with known number of objects and data associations, tracks each object using an optimal filter such as EKF and IMM filter. Although they showed promising performance on multiple synthetic datasets, its performance on real sensor systems, especially on vision systems, was not fully demonstrated. Therefore, in this work, we apply the algorithm to our vision-based multi-object tracking problem. Next, we briefly describe the problem formulation and how we applied the RBMCDA algorithm.

4.4.1 Problem Statement

The goal of multi-target tracking (MTT) problem is to estimate the state of targets given all the measurements obtained so far. In other words, we want to compute the posterior distribution $p(\mathbf{x}_k|\mathbf{z}_{1:k})$ recursively. In our case, \mathbf{x}_k contains the state variables of T objects, i.e., $\mathbf{x}_k = [\mathbf{x}_{k,1}, \dots, \mathbf{x}_{k,T}]^T$ and $\mathbf{z}_{1:k}$ denotes all the measurements up to time step k , i.e., $\mathbf{z}_{1:k} = \{\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_k\}$. This MTT problem inherently contains other sub-problems such as data association and estimation of target’s initialization and termination. The goal of data association and target’s birth and death estimation is, respectively, to find out the correct correspondences between current tracks and incoming new measurements and to detect new target’s initialization and old target’s termination. Considering those issues explicitly as latent variables λ_k , the whole problem can be formulated by the following decomposition of the posterior:

$$p(\mathbf{x}_k, \lambda_k|\mathbf{z}_{1:k}) = p(\mathbf{x}_k|\mathbf{z}_{1:k}, \lambda_k)p(\lambda_k|\mathbf{z}_{1:k}) \quad (4.14)$$

where λ_k denotes the latent variable which contains the visibility indicator \mathbf{e}_k and the data association indicator c_k at the time step k , i.e., $\lambda_k = \{\mathbf{e}_k, c_k\}$. It is important to understand that this factorization is always true, but only applicable to a problem when there is certain structure within the state variables. In this case, for example, the state variable is a mixture of state variables of T objects, a data association indicator that indicates which measurement corresponds to which track, and a visibility indicator which controls target’s initialization and termination. Once the second term in the right hand side of Eq. 4.14 is determined, the first term can be easily solved by our single object tracking method

described in the previous section. The RBMCDA algorithm solves the second term using a particle filter by analyzing only one measurement at a time assuming that at most one target can terminate at any time step. In addition, we reduce the complexity of the problem one step further to take advantage of the fact that our virtual sensor is a vision-based detector. Since the DPM object detector usually provides very accurate measurements (in terms of detection rate and bounding box localization), we can avoid target's birth and death estimation sub-problems from the whole problem. Instead, we initialize a track whenever we have a new measurement which could not be associated to one of existing tracks, and verify the track after M consecutive measurements, and terminate a track if there is no longer valid measurements for the track for N consecutive time steps. Therefore, the latent variable in our case would be, simply, $\lambda_k = c_k$.

4.4.2 Particle Filter for Data Association

Using a particle filter, the quantity we want to compute now is the distribution of $p(c_k | \mathbf{z}_{1:k})$, i.e., the probability of the current data association indicator, which has value $c_k = 0$ for clutter and $c_k = j$ for target $j \in \{1, \dots, T\}$, given all the measurements up to time step k . Note that we treat each measurement received at time step k sequentially not jointly. Then, conditioned on the data associations c_k the targets will remain independent during tracking. Due to this operational property of the RBPF, computation can be simplified in such a way that an individual target can be processed separately in each particle. The RBMCDA algorithm consists of a set of N particles, where each of particle i at time step k is defined as:

$$\{c_{k-m+1:k}^{(i)}, \mathbf{m}_{k,1}^{(i)}, \mathbf{P}_{k,1}^{(i)}, \dots, \mathbf{m}_{k,j}^{(i)}, \mathbf{P}_{k,j}^{(i)}, \dots, \mathbf{m}_{k,T}^{(i)}, \mathbf{P}_{k,T}^{(i)}, w_k^{(i)}\} \quad (4.15)$$

where $c_{k-m+1:k}^{(i)}$ are the data association indicators of time steps $k - m + 1, \dots, k$, with which an m th order Markov model can be constructed as a data association prior model. $\mathbf{m}_{k,j}^{(i)}, \mathbf{P}_{k,j}^{(i)}$ are the mean and covariance of the target j , and they are conditioned on the data association history $c_{1:k}^{(i)}$. $w_k^{(i)}$ is the importance weight of the particle. With this definition, the particle filter for data association updates as follows [98]:

-
1. Sample a new association from the optimal importance distribution:

$$c_k^{(i)} \sim p(c_k^{(i)} | \mathbf{z}_{1:k}, c_{1:k-1}^{(i)}). \quad (4.16)$$

2. Calculate new unnormalized weights as

$$\begin{aligned} w_k^{*(i)} &\propto w_{k-1}^{*(i)} \\ &\times \frac{p(\mathbf{z}_k | c_k^{(i)}, \mathbf{z}_{1:k-1}, c_{1:k-1}^{(i)}) p(c_k^{(i)} | c_{1:k-1}^{(i)})}{p(c_k^{(i)} | \mathbf{z}_{1:k}, c_{1:k-1}^{(i)})} \end{aligned} \quad (4.17)$$

3. Normalize the weights to sum to unity as

$$w_k^{(i)} = \frac{w_k^{*(i)}}{\sum_{j=1}^N (w_k^{*(j)})^2} \quad (4.18)$$

4. Estimate the effective number of particles as

$$n_{eff} \approx \frac{1}{\sum_{j=1}^N (w_k^{(j)})^2} \quad (4.19)$$

If the effective number of particles is less than a predefined threshold, perform resampling. This is why this version of a particle filter is also known as the Sequential Importance Resampling (SIR) filter [86].

To implement the above SIR filter, three main ingredients in Equation 4.17 should be constructed for each particle i :

- The *predictive* probability $p(c_k | c_{1:k-1}^{(i)})$ which gives the prior probability of the data association given the old data associations. Since we use m th order Markov model, that would be $p(c_k | c_{k-m:k-1}^{(i)})$ in our case.
- The *measurement likelihood* $p(\mathbf{z}_k | c_k, \mathbf{z}_{1:k-1}, c_{1:k-1}^{(i)})$ which gives the likelihood of a measurement conditioned on all previous measurements, previous data associations and the current data association. Whereas the RBMCDA algorithm suggested using the Kalman filter measurement likelihood evaluation

based on a geometric clue, in this work, we make use of an image-based appearance cue as well as a geometric cue from a bounding box. Our measurement likelihood is given by

$$\begin{aligned}
& p(\mathbf{z}_k | c_k, \mathbf{z}_{1:k-1}, c_{1:k-1}^{(i)}) \\
&= \begin{cases} 1/V & \text{if } c_k = 0 \\ \Gamma(\mathbf{d}_k, \mathbf{d}_j) & \text{if } c_k = j \end{cases} \quad (4.20)
\end{aligned}$$

with a definition of $\Gamma(\mathbf{d}_k, \mathbf{d}_j)$

$$\left(1 - \frac{\text{box}(\mathbf{d}_k) \cap \text{box}(\mathbf{d}_j)}{\text{box}(\mathbf{d}_k) \cup \text{box}(\mathbf{d}_j)}\right) \times (1 - \text{xcorr}(\mathbf{d}_k, \mathbf{d}_j)) \quad (4.21)$$

where \mathbf{d}_k and \mathbf{d}_j are two object detections (defined in Equation 4.8) associated to the measurements of \mathbf{z}_k and j 's target, respectively. $\text{box}(\cdot)$ represents the bounding box that belongs to a detected object and $\text{xcorr}(\cdot, \cdot)$ returns the maximum of the normalized cross-correlation of two detections.

- The *optimal importance distribution* $p(c_k | \mathbf{z}_{1:k}, c_{1:k-1}^{(i)})$ is very important for the efficiency of a SIR filter. The posterior distribution of c_k can be calculated using Bayes' rule:

$$\begin{aligned}
& p(c_k | \mathbf{z}_{1:k}, c_{1:k-1}^{(i)}) \\
& \propto p(\mathbf{z}_k | c_k, \mathbf{z}_{1:k-1}, c_{1:k-1}^{(i)}) p(c_k | c_{k-m:k-1}^{(i)}) \quad (4.22)
\end{aligned}$$

Note that we already computed the second and third terms in the equation. Therefore we can sample from the optimal importance distribution directly.

4.5 Experiments

We quantitatively evaluated our vision-based multi-object tracking system using various challenging real world datasets. To train three major object models, we collected a large amount of video data of pedestrians, bicyclists, and vehicles based on the accident statistics [74, 75]. As for the object tracking evaluation in a real application context, we collected six additional video sequences. To

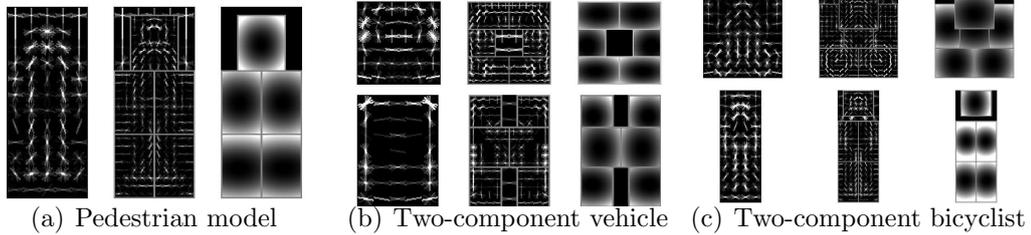


Figure 4.3: Visualization of three major object models trained from our new urban datasets. For each model, the first, second, and third columns represent a root filter and part filters, and a deformation model, respectively.

evaluate tracking performance of the single object tracking method, first three video sequences (i.e., ‘Seq1’~‘Seq3’, recorded at 13fps with 320×240) of a bicyclist case were analyzed in terms of position estimation. The reason we take bicyclist class is that bicyclists are particularly challenging for an onboard vision system to track due to the fact that they are as unprotected as pedestrians but travel at higher speeds in very close proximity to vehicles. In addition, there are three video sequences, one for each object class, for multi-object scenarios. Some important statistics of the video sequences are summarized in Table 4.1.

4.5.1 Training DPMs for Urban Environments

Pedestrian: We used Caltech Pedestrian Dataset [33] to train a pedestrian model. The Caltech dataset corresponds to approximately 2.3 hours of video (640×480 resolution) captured at 30fps and is segmented into 11 sessions, 6 of which are used for training (S0~S5) and the rest sessions are used for testing (S6~S10). We trained a model using the training set sampled at every 30th frame and filtered by a height condition (i.e., $40 < h < 400$ pixels), resulting in 2,198 positive samples.

Bicyclist: We collected a new bicyclist dataset (captured at 3fps with 1280×960 resolution) in various places and weather conditions. It consists of two training sets (S0~S1) and two testing sets (S2~S3), each with one or two short video sequence files. Considering the bicycle accident statistics [53], most of samples correspond to rear view and side view of samples of bicyclists. We trained two-component bicyclist models using 2,845 and 427 positive samples for rear and

side view, respectively.

Vehicle: We also collected a new vehicle dataset (captured at 3fps with 1280×960 resolution) in various places and weather conditions. It consists of three training sets (S0~S2) and two testing sets (S3~S4), each with one or two short video sequence files. Since we target Forward Collision Warning (FCW) application, most of samples corresponds to rear view samples of vehicles such as sedan, SUV, truck, and bus. The dataset was very carefully collected based on our investigation about vehicle accident statistics [75]. We trained two rear view vehicle models using the training data (S0~S2), one for sedan and SUV sub-class and one for bus and truck sub-class. We use 10,320 and 689 positive samples for the sedan-SUV vehicle model and truck-bus vehicle model, respectively. Through experiments, we found that those combinations offer a good trade-off between detection accuracy and running time. All trained models are displayed in Figure 4.3(a)~4.3(c).

4.5.2 Single Object Tracking Evaluation

For the performance of single object tracking, as partially shown in Figure 4.4(a)~4.4(f), the proposed EKF based tracking method successfully tracks a bicyclist except for the case in which the bicyclist is shown beyond the effective (or working) distance of the DPM detector, which is around 10m for the setting (i.e., 320×240 resolution). In ‘Seq. 1’, a bicyclist approaches to the ego-vehicle (stationary) and quickly turns right to avoid the vehicle. In ‘Seq. 2’, a bicyclist comes across the road from right to left while the ego-vehicle moves forward. In ‘Seq. 3’, a bicyclist comes across the road in front of the ego-vehicle (stationary) and makes a turn toward the vehicle so that the left side and frontal view of the bicyclist are seen and tracked.

More detailed analyses for each sequence are provided by plotting filtered state variables of the tracker at each time step. In our case, that would be the positions in x and z coordinates of the bicyclist. For instance, Figure 4.4 (d)~(f) show bird-eye views of the bicyclist position estimation in x and z coordinates for the ‘Seq1’~‘Seq3’, respectively. We also have listed the root mean square errors (RMSE) of position estimates in the last column in Table 4.1.

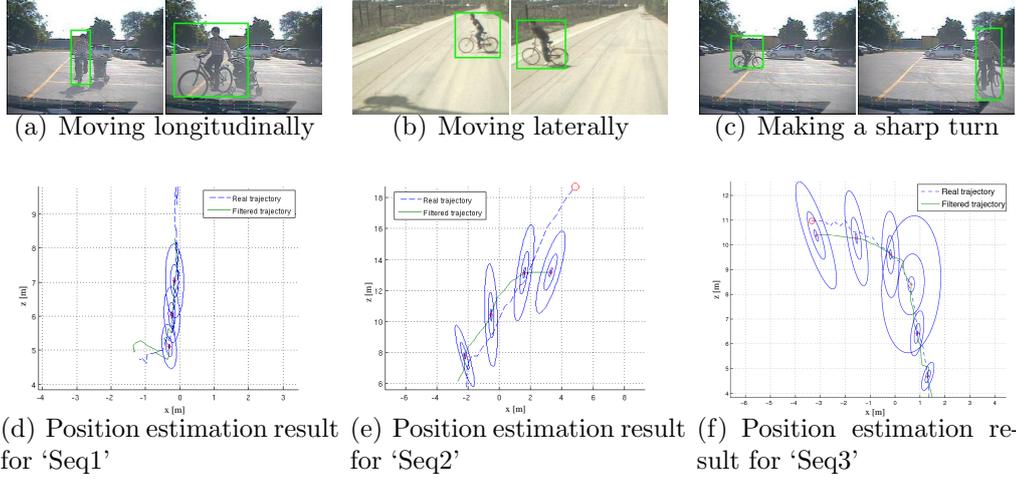


Figure 4.4: Performance analysis on single object tracking. (a), (b), and (c) show typical raw detections obtained by our bicyclist detector and (d), (e), and (f) show the estimated path trajectory of the tracked bicyclists, respectively. The ellipses represent the 1, 2, 3-sigma confidence regions for the position estimate.

Table 4.1: Details of single bicycle tracking result.

Seq. No.	Ego-vehicle	Bicyclist	RMSE(m)
‘Seq1’	stationary	longitudinally	1.17
‘Seq2’	moving	laterally	4.34
‘Seq3’	stationary	sharp turn	2.35

4.5.3 Multi-Object Tracking Evaluation

For multi-object tracking, the RBMCDA algorithm applied to vision-based scenarios shows very promising performance. We applied the multi-object tracker on two challenging data sets, one for pedestrians and the other one for vehicles. Details on the test sequences are shown in the Table 4.2. In both cases, the tracker successfully tracks multiple pedestrians and vehicles reliably against a large amount of motion of the ego-vehicle and target objects.

In the ‘Seq4’ case shown in Figure 4.5, the tracker can track three pedestrians on the sidewalk distance up to 26m. As the ego-vehicle approaches to a crosswalk, the tracker starts to track two pedestrians trying to walk across the crosswalk.



Figure 4.5: Qualitative multi-object tracking results on the Caltech test set.

Thanks to the image appearance features as well as geometric features used for data association, the RBMCDA algorithm can track the further pedestrians well even when there is quite amount of occlusion. The second row of Figure 4.5 shows this result. Finally, the third row of the figure shows how the tracker can quickly catch up the previous tracks when tracked pedestrians are completely occluded by a passing car for a short period of time (i.e., one or two seconds).

In the ‘Seq5’ case shown in Figure 4.6, the tracker can easily track multiple vehicles on the road. Selected tracking result images are also shown in the first

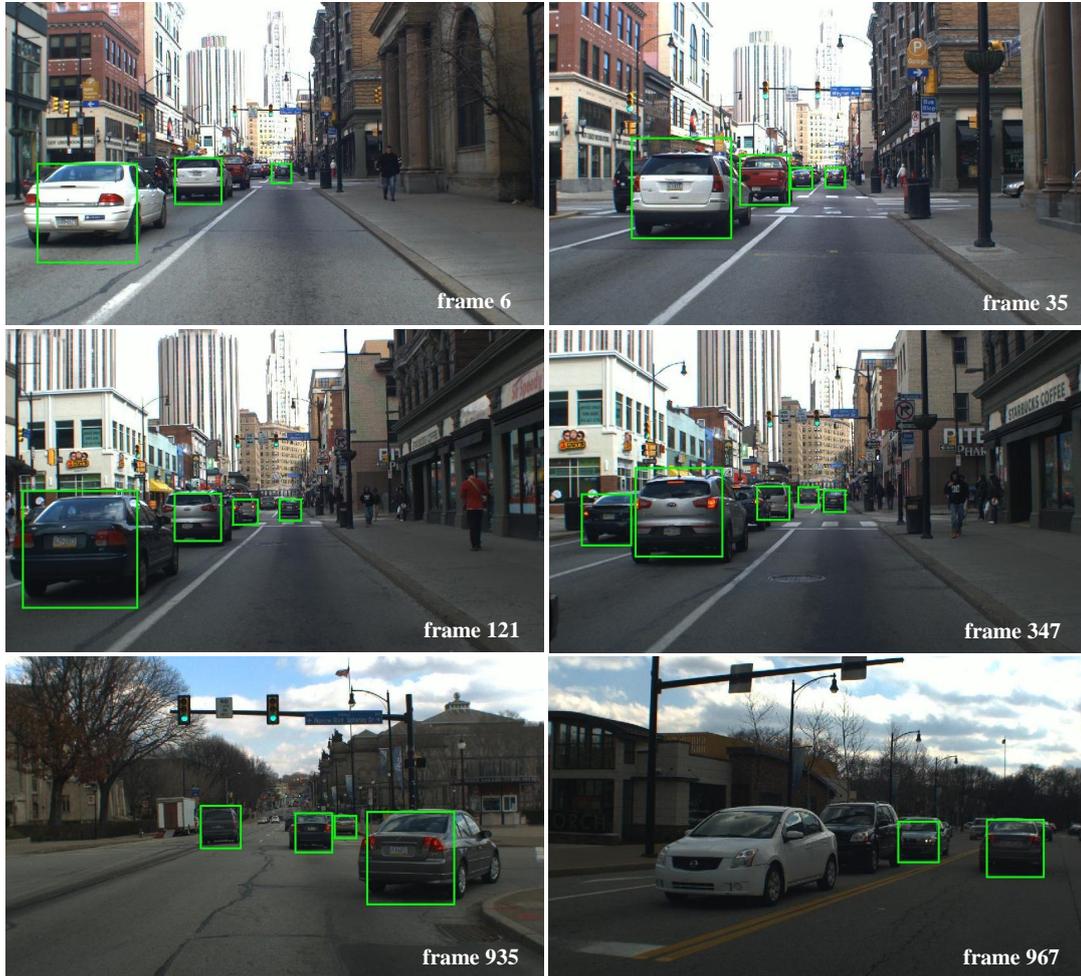


Figure 4.6: Qualitative multi-object tracking results on our vehicle data set.

and second rows in Figure 4.6. In the third row, our tracker manages to track a leading vehicle despite significant turning motion and dynamic viewpoint changes of the vehicle.

4.6 Summary

This chapter presented a vision-based multi-object tracking method for autonomous vehicles using a deformable part-based models (DPMs) and the Rao-Blackwellized Monte Carlo Data Association (RBMCD) algorithm. To robustly detect three

Table 4.2: Details of multi-object sequences used in the evaluation.

Seq. No.	Resolution	Frame No.	No. of Objects
‘Seq4 (ped.)’	640×480	492	8
‘Seq5 (vehicle)’	1280×960	1,015	5

major urban objects, we used one of the most successful state-of-the-art object detectors [41, 40]. This robust object detector provides a series of measurements (i.e., bounding boxes) to the tracking framework. We defined two different motion models to describe the kinematics of objects. For each motion model, an extended Kalman filter (EKF) is used to estimate the position and velocity of an object in the vehicle coordinate system. Finally, we show how we extend our single object tracking method to allow it to track multiple objects by incorporating a Rao-Blackwellized Particle Filter to solve the data association problem. This complementary approach allows our system to effectively track an urban object even when it changes orientation (and thus appearance) in the image. We have shown several experiments that illustrate the effectiveness of each component of the proposed method.

Chapter 5

Multi-Sensor Fusion for Moving Object Tracking

This chapter presents our new moving object detection and tracking system that extends and improves our earlier system used for the 2007 DARPA Urban Challenge. The new system uses revised motion and observation models for active sensors (i.e., radars and LIDARs) and introduces a vision sensor. In the new system, the vision system we developed in Chapter 3 detects pedestrians, bicyclists, and vehicles to generate corresponding vision targets. The new tracking system utilizes this visual recognition information to improve a tracking model selection, data association, and movement classification of our earlier system. Section 5.1 discuss the overview and main contributions of the proposed system. Section 5.2 and Section 5.3 detail the configuration and calibration of multiple sensors in different modalities, respectively. Section 5.4 discusses the characteristics of each sensing modality. Section 5.5 describes interactions between our vision sensor based object detection system and active sensor based object tracking system. Section 5.6 discusses the experimental results and the findings. Section 5.7 summarizes this chapter.

5.1 Introduction

The 2005 DARPA Grand Challenge and the 2007 DARPA Urban Challenge offered researchers with unique opportunities to demonstrate the state-of-the-art in the autonomous driving technologies [102, 60, 71, 69, 85]. These events were milestones in that they provided opportunities of reevaluating the status of the relevant technologies and of regaining the public attention on self-driving car development. Since then, the relevant technologies have advanced significantly. Industry and academia have reported notable achievements including: autonomous driving more than 300,000 miles in daily driving contexts [101], intercontinental autonomous driving [18], a self-driving car with a stock-car appearance [109], and many more. Such developments and demonstrations have increased possibility of self-driving cars in near future.

After the DARPA Urban Challenge, Carnegie Mellon University started a new effort to advance the findings of the DARPA Urban Challenge and developed a new autonomous vehicle [109] to fill the gap between the experimental robotic vehicles and consumer cars. Among these efforts, this chapter details our perception system, particularly, a new moving objects detection and tracking system. The Urban Challenge was held in a simplified, urban driving setup where restricted vehicle interactions occurred and no pedestrians, bicyclists, motorcyclists, traffic lights, GPS dropouts appeared. However, to be deployed in real-world driving environments, autonomous driving vehicles must be capable of safely interacting with nearby pedestrians and vehicles. The prerequisite to safe interactions with nearby objects is reliable detection and tracking of moving objects.

To this end, we first propose a practical multi-sensor configuration for the perception system. Currently, majority of autonomous vehicles (e.g., Google’s self-driving car) seems to utilize a high definition LIDAR sensor such as Velodyne HDL-64 [10], which is almost always installed on the roof of a vehicle. We believe that this bulky sensor setup on the roof would be problematic when we start to regard autonomous driving as a part of our daily life.

Secondly, to compensate for the lack of such a high definition LIDAR, we investigate vision sensor’s capability in terms of object detection, tracking, and classification. This is mainly because knowledge of moving objects’ classes (e.g.,

car, pedestrian, bicyclists, etc.) is greatly helpful to reliably track them and derive a better inference about driving contexts and current automotive grade LIDAR sensors provide enough measurements for position and velocity estimation of objects, but not enough for reliable recognition or classification of objects. We believe that, as discussed in Section 1.4, multiple vision sensors together with automotive grade LIDAR sensors could replace the role of such a high definition LIDAR sensor.

Finally, we exploit high-level semantic information from vision-based object detection with intermediate features from radar and LIDAR sensors. This object class information is utilized as a key factor in several sub-components of the tracking system such as model selection, data association, and movement classification. A more appropriate tracking model (either a box model or a point model) is selected depending on the object class information and quality of the sensor data. In addition, when a LIDAR's edge feature is associated to an object being tracked with a box model, the class information is used for pruning unnecessary enumerations of interpretation of the edge feature for vehicles. In addition, a customized parameter set can be used for movement classification based on an object type.

5.2 Multi-Sensor Setup

The underlying ideas of our sensor configuration are to 1) minimize any potential alterations of a vehicle's original appearance, 2) completely cover the area around the vehicle within a certain range, and 3) utilize existing, off-the-shelf and stock-car grade sensors. Based on these guidelines and prior experiences, a new sensing system was built as shown [109] in Figure 5.1 (a). All sensors are seamlessly integrated into the vehicle chassis and their appearance is indistinguishable from outside. In particular, the vehicle is equipped with six radars, six LIDARs, and two cameras. A radar is paired with a LIDAR. This was done to maximize the reliability and range of measurements. With the current sensor layout, any objects within 200 meters will be projected onto the sensing coverage and any objects within 60 meters or so will be seen by at least two different types of sensors (i.e., radar and LIDAR, or radar and camera). Figure 5.1 (b) illustrates measurements

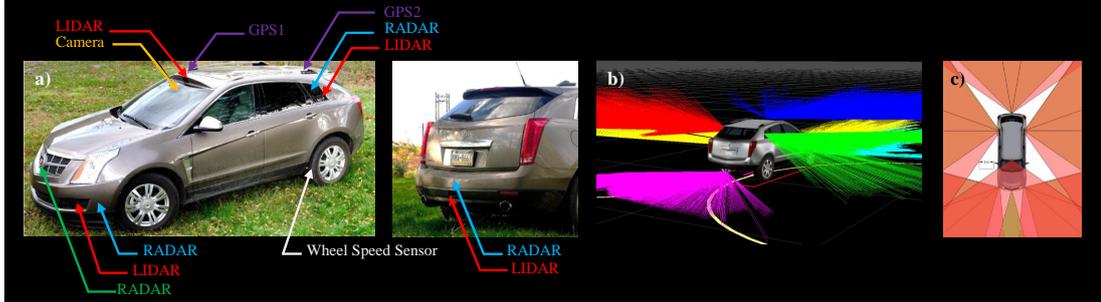


Figure 5.1: CMU’s new autonomous vehicle and its sensor configuration. (a) CMU’s new autonomous vehicle is designed to minimize alterations of a stock-car appearance while installing multiple sensors to maximize the sensing coverage. (b) Visualization of LIDAR measurement. LIDAR scans acquired from individual sensors are depicted in different color. (c) A horizontal field of view (HFOV) of sensing coverage, emphasizing the coverage around the vehicle.

from all six LIDAR sensors. For the vision sensors’ setup, a camera is installed, in a forward-looking manner, inside the front window, next to the rear-view mirror and another is installed at the rear bumper to provide the front and back side of perspective images. The third camera is a thermal camera that captures scenes in infrared spectrum to perceive objects in challenging driving conditions, such as at night and in fog. Table 5.1 details the types and specifications of the sensors. All these sensors are stock-car grade and readily available on the market. Figure 5.1 (c) depicts the blind spots. Due to the integration of multiple wide-FOV sensors the blind spots are small enough that no vehicle will be overlooked.

5.3 Sensor Calibration

All 14 sensors listed in Table 5.1 are carefully calibrated with respect to the host vehicle’s coordinate system. Each sensor coordinate system as well as the vehicle coordinate system are defined as shown in Figure 5.2, i.e.:

- Camera: $x = \text{right}$, $y = \text{down}$, $z = \text{forward}$
- LIDAR: $x = \text{forward}$, $y = \text{left}$, $z = \text{up}$
- Radar: $x = \text{forward}$, $y = \text{left}$, $z = \text{up}$

Table 5.1: Installed sensors specifications and their primary usages.

Sensor Name (Type)	HFOV (°)	MaxRange (m) /Resolution	Update Rate (Hz)	Tracking Features
IBEO LUX [7] (LIDAR)	85~110	200	12.5/25/50	edge target
Bosch LRR3 [1] (Radar)	30	250	12.5	point target
Delphi ESR [3] (Radar)	90 (near) 20 (far)	60 (near) 174 (far)	20	point target
Flea3 [4] Camera	45	2448×2048	8	vision target
FLIR [5] Camera	36	640×480	24	vision target

- Vehicle: x = forward, y = right, z = down

Note that the origin of the vehicle coordinate system lies at the center of the rear axle of the host vehicle, which is the location where the IMU (Inertial Measurement Unit) of the high performance localization system [109] exists.

5.3.1 Synchronization

All six IBEO LUX sensors are synchronized using the external triggering signal from the synchronization unit provided from the sensor manufacturer [7] so that synchronized universal scan data from the six LIDARs can be obtained. Unfortunately, other sensors (i.e., six radars and two cameras) do not have such a hardware synchronization device so that essentially there is some timing difference between (possibly) all sensors. However, the localization information from the GPS/IMU system updates at 100Hz, and this issue can be addressed by registering all sensor data according to their timestamps.

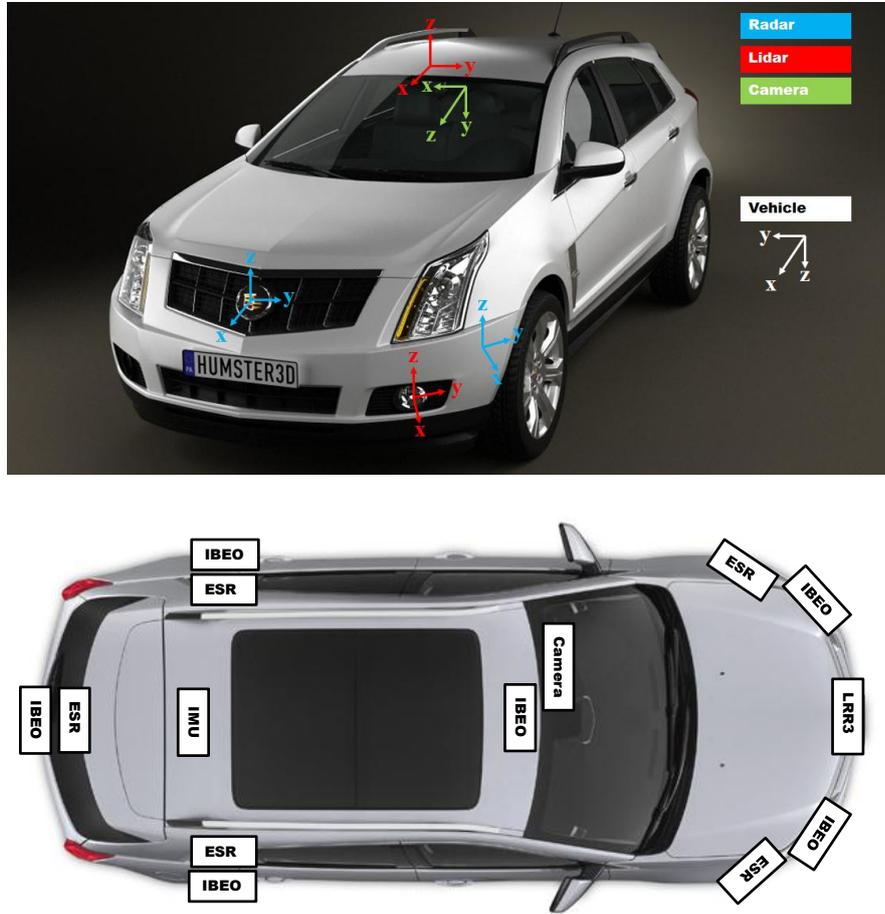


Figure 5.2: (a) Definition of sensor coordinate systems as well as the vehicle coordinate system. (b) The mounting location of sensors on the vehicles.

5.3.2 Camera and IBEO LUX Calibration

For the intrinsic parameters (\mathbf{K}) of the forward-looking camera, we used the Caltech calibration toolbox [114]. With this information, the projection of a 3D point in the camera coordinates $\mathbf{x} = (x, y, z, 1)^T$ into a point $\mathbf{y} = (u, v, 1)$ in the image is given by

$$\mathbf{y} = \mathbf{K}\mathbf{x} \tag{5.1}$$

For the extrinsic parameters of the camera, we found the relative spatial relationship between the camera and the IBEO LUX (\mathbf{T}_{ibeo}^{cam}) mounted on the roof using the method proposed in [58]. With this information, now we can project 3D points in the IBEO LUX sensor coordinates $\mathbf{x}^{ibeo} = (x, y, z, 1)^T$ into the image by

$$\mathbf{y} = \mathbf{K} \mathbf{T}_{ibeo}^{cam} \mathbf{x}^{ibeo} \quad (5.2)$$

5.4 Sensor Characterization

Once measurements from any sensors are delivered to the tracking system, they are treated similarly as units of measurements, but represented differently based on their sensing modalities. This section discusses the characteristics of each sensing modality.

5.4.1 Radar

A radar provides 2-dimensional position and velocity of an object. It usually reaches objects at relatively farther distances (e.g., more than 200 meters) from a host vehicle and offers a direct velocity measurement (using Doppler shift effect). We represent radar measurements at time step k as

$$\begin{aligned} \mathbf{o}_k^R &= \{\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_p\} \\ \mathbf{r}_i &= [x \ y \ \dot{x} \ \dot{y}]^T \quad i = 1, \dots, p \end{aligned} \quad (5.3)$$

where \mathbf{r}_i is a point position and velocity measurement with respect to the radar sensor coordinate and p is the number of radar measurements at time step k . As shown in Figure 5.3, the uniform information regardless of distances is a unique feature of radar sensors.

5.4.2 LIDAR

By contrast, measurements from LIDAR sensors provide a varying density of 3-dimensional point cloud depending on distances. Mostly these point measure-

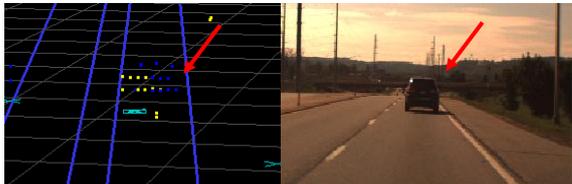
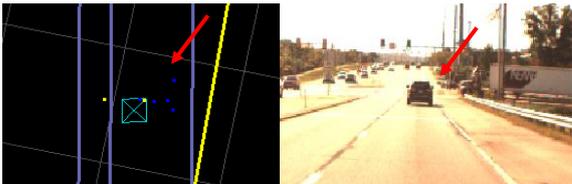
Distance	Sensor Measurements	No. of Pts in LIDAR	Pixel size of car in image
20m		50	58 x 58
40m		20	32 x 32
60m		8	20 x 20
80m		4	14 x 14

Figure 5.3: Example images show raw sensor data as a function of distances.

ments are dense enough to partially or completely delineate the shape of objects. In our case (i.e., IBEO LUX), as shown in Figure 5.3, this distance corresponds to around 40 m. Note that the actual formation of point clouds and their coverages of objects' shapes are dependent upon various factors, e.g., field of view (FOV), angular resolutions, line of sight between that sensor and an object. A high-density measurement comes with additional processing cost because it is necessary to pre-process (e.g., segmentation or feature extraction, like line segments or corners) point clouds to make them attached to objects to track. For example,

to keep tracking the vehicle right front of a car, one needs to know which of point clouds are parts of the vehicle (i.e., segmentation) and to represent that clustered point cloud as a computational form (i.e., feature extraction – representing the set of clustered points as a line).

For representing LIDAR measurements, we treat six, four-plane LIDARs as one homogeneous sensor, analyze their measurements using built-in segmentation and extract features, like line segments or junctions of lines (“L”) shape [67]. LIDAR measurements at time step k are expressed by:

$$\begin{aligned}\mathbf{o}_k^L &= \{\mathbf{l}_1, \mathbf{l}_2, \dots, \mathbf{l}_q\} \\ \mathbf{l}_i &= [x \ y \ \psi \ \dot{x} \ \dot{y} \ w \ l]^T \quad i = 1, \dots, q\end{aligned}\tag{5.4}$$

where \mathbf{l}_i consists of the position of the center of the box (fitted by the feature), orientation (ψ), velocity, width (w), and length (l) of the box. In fact, w is computed as $\max(OW, e1)$, where OW is the canonical width of that object class and $e1$ is the actual measured length of a short edge of the feature. The same idea applies to l . q is the number of LIDAR measurements at time step k .

5.4.3 Camera

Lastly, cameras provide high-definition snapshots of scenes. While rich information in the image frames makes vision data interesting, determining what features to extract and how to interpret them for detection and tracking of moving (or even static) objects is still an active research topic. In this thesis, we exploit the object category detection to effectively utilize visual information. In Chapter 3, we developed a real-time onboard vision system that aims to identify pedestrians, bicyclists, and vehicles. For sensor fusion purpose, we represent the detected objects using bounding boxes and treat them as measurements from vision sensors. Then camera measurements at time step k is expressed by:

$$\begin{aligned}\mathbf{o}_k^C &= [\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_r] \\ \mathbf{c}_i &= [x1 \ y1 \ x2 \ y2 \ c]^T \quad i = 1, \dots, r\end{aligned}\tag{5.5}$$

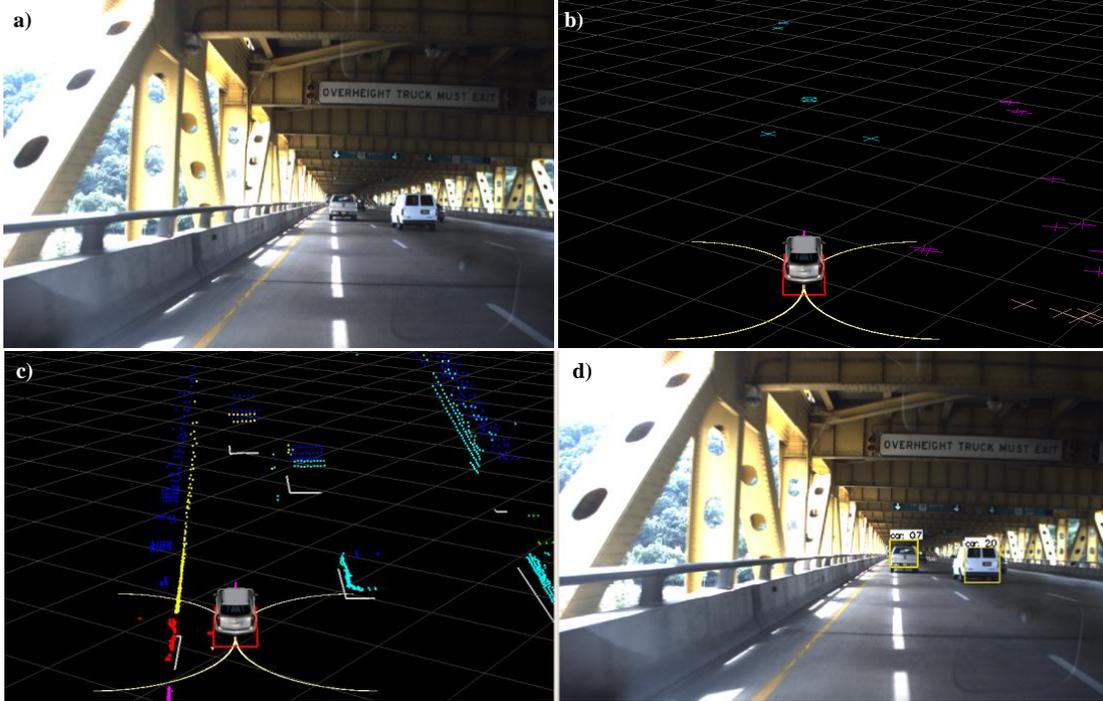


Figure 5.4: Example images show raw sensor data and extracted features as measurements. (a) An input scene. (b) Raw data from six radars. The data are used as features (called ‘point target’) for tracking directly. (c) Raw scan data from six LIDARs. “L” shaped features (called ‘edge target’) are extracted for tracking. (d) Bounding boxes from the vehicle detection system are used as features (called ‘vision target’) for tracking.

where $(x1, y1)$ and $(x2, y2)$ are the coordinates of the left-top and right-bottom point of a bounding box in the image space, respectively and c indicates an object class (i.e., pedestrian, bicyclist, or vehicle). r is the number of bounding box measurements at time step k .

In summary, we represent measurements from different sensors differently based on individual sensors’ acquisition principles and operating characteristics, but treat them in a common way to facilitate the information fusion process. Our tracking system takes measurements from three different types sensors at time step k as:

$$\mathbf{o}_k = \{\mathbf{o}_k^R, \mathbf{o}_k^L, \mathbf{o}_k^C\} \quad (5.6)$$

In practice, these measurements are asynchronously acquired by each sensor and are timestamped to be published on the data communication channel. Figure 5.4 shows an example of those sensor data and corresponding extracted features.

5.5 Multi-Sensor Fusion

It is a challenge to seamlessly fuse measurements from 14 sensors and to generate tracking results consistent over time. To effectively address such a challenge, we extended, based on lessons learned from participation of several autonomous vehicle competitions, our earlier tracking system [29] and introduced new methods to effectively tackle real-world perception problems occurring in urban autonomous driving. Figure 5.5 shows a diagram that describes our new tracking system. Our system consists of two parts: sensor and fusion layer. By taking care of hardware specific operations, the sensor layer offers a separation between actual sensing hardware and specific tasks regarding detection and tracking of objects. By this way, the tasks at the fusion layer can be developed without knowing the details about the lower-level’s sensing mechanisms. Each sensor reader acquires raw sensor data and extracts features (e.g., lines or corners), if any, and publishes them in a shared communication channel. A task at a higher-level, fusion layer, can pick up these features from the channel for its purpose. For example, based on lower-level’s features, e.g., point or polygonal shapes, we execute point or box models to track the feature over time. For the underlying rationale of such a tracking architecture, we refer readers to [29].

5.5.1 Sensor Fusion with Kalman filters

To fuse sensor’s measurements to accurately detect and consistently track neighboring objects, we applied an Extended Kalman Filter (EKF). The (extended) Kalman filter has a number of features which make it ideal for dealing with complex multi-sensor fusion problems. In particular, the explicit description of motion and observation models allows a wide variety of different sensor modalities to be incorporated within the filter. In addition, the consistent use of statistical measures of uncertainty makes it possible to quantitatively evaluate the role each

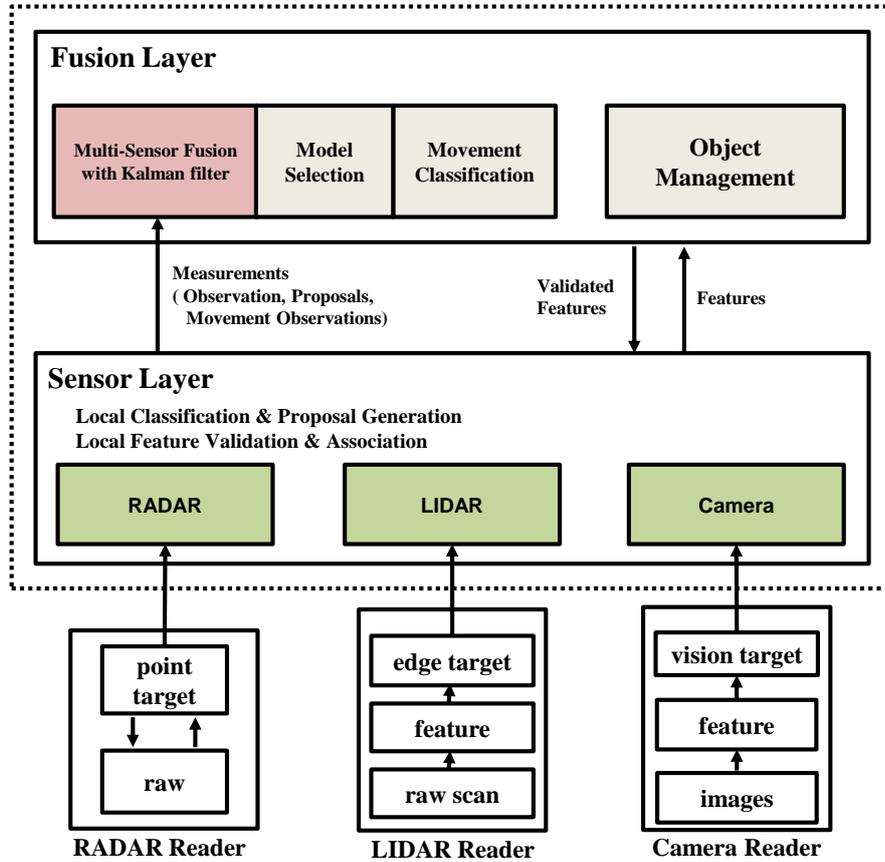


Figure 5.5: A diagram of our tracking system. Our system is mainly comprised of two layers: a sensor and a fusion layer. We enhance and improve the architecture of our earlier tracking system [29] by adding a vision sensor.

sensor plays in the context of overall system performance.

To effectively apply this filter to our setup, we employ the sequential-sensor method [35] that treats observations from individual sensors independently and sequentially feeds them to the EKF’s estimation process. We choose such a method to sequentially process multiple, heterogeneous measurements arriving in an asynchronous order.

5.5.2 Tracking Models

This work aims at developing a tracker that, using multiple sensors in different modalities, reliably tracking pedestrians, bicyclists, and vehicles. To effectively handle the constraints and characteristics of target objects' motions, we use two motion models: a point model (\mathcal{M}_P) and a 3D box model (\mathcal{M}_B). In particular, we expanded our earlier, 2-dimensional box model [29] to 3-dimensional one, to realistically represent the detected objects. For the three different sensing modalities, we devise three observation models: radar (\mathcal{O}_R), LIDAR (\mathcal{O}_L), and camera (\mathcal{O}_C) observation model.

$$\begin{aligned}\mathcal{M} &: \{\mathcal{M}_P, \mathcal{M}_B\} \\ \mathcal{O} &: \{\mathcal{O}_R, \mathcal{O}_L, \mathcal{O}_C\}\end{aligned}\tag{5.7}$$

Motion Models: Each of three moving objects of interest (i.e., pedestrians, bicyclists, and vehicles) has its own motion kinematics and constraints. For example, a pedestrian can move in any directions whereas the motions of a vehicle or a bicyclist is confined by non-holonomic constraints. To estimate these motions, we use two motion models: a point model and a 3D box model. For the point model, we assume an object moves with a constant acceleration [15]. We represent the state of the moving point at time step k by its 2-dimensional coordinates, velocities, and accelerations:

$$\mathbf{x}_k = [x_k \quad y_k \quad \dot{x}_k \quad \dot{y}_k \quad \ddot{x}_k \quad \ddot{y}_k]^T\tag{5.8}$$

and the discrete-time state equation for this point model is given by [15]:

$$\mathbf{x}_{k+1} = \begin{bmatrix} 1 & 0 & T & 0 & T^2/2 & 0 \\ 0 & 1 & 0 & T & 0 & T^2/2 \\ 0 & 0 & 1 & 0 & T & 0 \\ 0 & 0 & 0 & 1 & 0 & T \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x}_k + \begin{bmatrix} 1/2T^2 \\ 1/2T^2 \\ T \\ T \\ 1 \\ 1 \end{bmatrix} q_k\tag{5.9}$$

where T is a sampling period and q_k is a scalar-valued discrete-time process noise.

Secondly, our 3D box model is a simplified bicycle model (SB) [54] with its estimated 3D cuboid. The state of a 3D box model is represented by

$$\mathbf{x}_k = [x_k \quad y_k \quad \psi_k \quad v_k \quad \omega_k \quad a_k \quad w_k \quad l_k \quad h_k]^T \quad (5.10)$$

where (x, y) , ψ , v , ω , and a are the position of the center of the box, yaw angle, velocity, yaw rate, and acceleration. The yaw angle defines the orientation of the velocity and acceleration vectors. The volume of a 3D box is defined by its components, width, w , length, l , and height, h . For the state propagation, an approximated version of the SB model (Equation 4.6) is used.

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \begin{bmatrix} (v_k + \frac{1}{2}a_kT) \cos(\psi_k)T \\ (v_k + \frac{1}{2}a_kT) \sin(\psi_k)T \\ \omega_k T \\ a_k T \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ T \\ T \end{bmatrix} q_k \quad (5.11)$$

where T is a sampling period and q_k is a scalar-valued discrete-time process noise. Since the propagation of the state is a nonlinear function, corresponding Jacobian, i.e., $(\frac{\partial \mathbf{f}}{\partial \mathbf{x}})$ should be computed to get the final form of linearized propagations, which is given by:

$$\begin{bmatrix} 1 & 0 & -v_k \sin(\psi_k)T - \frac{1}{2}a_k \sin(\psi_k)T^2 & \cos(\psi_k)T & 0 & \frac{1}{2} \cos(\psi_k)T^2 \\ 0 & 1 & v_k \cos(\psi_k)T + \frac{1}{2}a_k \cos(\psi_k)T^2 & \sin(\psi_k)T & 0 & \frac{1}{2} \sin(\psi_k)T^2 \\ 0 & 0 & 1 & 0 & T & 0 \\ 0 & 0 & 0 & 1 & 0 & T \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.12)$$

Observation Models: We devise three different observation models for each of three different sensing modalities: \mathcal{O}_R , \mathcal{O}_L , and \mathcal{O}_C .

The radar observation model (\mathcal{O}_R) aims at modeling observations about a

point target. It is designed to process direct position and velocity measurements.

The LIDAR observation model (\mathcal{O}_L) is primarily used to model a box target. This is a nonlinear mapping of the state space into the LIDAR's measurement space.

$$\begin{bmatrix} x \\ y \\ \psi \\ \dot{x} \\ \dot{y} \\ w \\ l \end{bmatrix} = \begin{bmatrix} x(k) \\ y(k) \\ \psi(k) \\ v(k) \cos(\psi(k)) \\ v(k) \sin(\psi(k)) \\ w(k) \\ l(k) \end{bmatrix} + \mathbf{v}(k) \quad (5.13)$$

where $\mathbf{v}(k)$ is the measurement noise at time step k . To make this noise realistic, one needs to analyze a collected, labeled LIDAR data set to derive its statistics such as covariance matrix. Our LIDAR observation model \mathcal{O}_L is derived to support both the point motion model \mathcal{M}_P and the 3D box model \mathcal{M}_B . For example, when \mathcal{O}_L is used for \mathcal{M}_P , only the position measurement is used to update the state, where the position corresponds to the center of the edge that is closer to the host vehicle.

The last observation in our system is the camera observation model (\mathcal{O}_C). The camera observation model is primarily used to deal with bounding box measurements in the image plane. However, due to depth ambiguity, we do not use such bounding box detections to update motion estimation, but use the detection results to estimate the width and the height of an object and determines objects' classes. Accordingly, \mathcal{O}_C cannot be used for a new object initialization or termination. If the data association between image frames is correctly done, it is straightforward to compute the relationship between a pixel height $y_2 - y_1$ (width $x_2 - x_1$) and a physical height $h(k)$ (width $w(k)$), based on the camera geometry: $y_2 - y_1 \approx h(k)f_p/d$, where f_p is the focal length expressed in pixels and d is a distance which we can estimate in a precise manner via radars and LIDARs. Based on this, we define the camera observation model (\mathcal{O}_C) for a box

model (\mathcal{M}_B) as

$$\begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \end{bmatrix} = \begin{bmatrix} w(k) f_p / d \\ h(k) f_p / d \end{bmatrix} + \mathbf{v}(k) \quad (5.14)$$

Note that the \mathcal{O}_C can support only the \mathcal{M}_B since a \mathcal{M}_P model does not have the concept of shape.

5.5.3 Data Association

It is critical to associate the current measurements with the earlier state variables, to optimally estimate the state of tracking objects. This section details the improvement we made on our previous data association algorithm [29] for radar and LIDAR sensors by utilizing our new camera observation model.

5.5.3.1 Camera

Firstly, to associate camera observations (we call vision targets) over frames, we project the center of the predicted moving object hypotheses, represented by either a point or a box model, onto the next image frame under the pinhole camera model. After the projection, we search for the nearest neighbor that minimizes the distance between the projected point and the center of the bottom line of the detected bounding boxes. Figure 5.6 (a) illustrates this search. Once such an association is successfully made, the camera observation and its object classification is instantiated using equation 5.14. For the box model, its volume is also associated as well as its object class membership. For a point model, the observation are instantiated only with its class membership.

5.5.3.2 Radar

For data association of radar observations (we call point targets), a set of possible point targets is generated from the predicted moving object hypotheses. Since radars are usually poor in determining a lateral position of an object, when a tracked object is modeled as a 3D box model, we generate multiple points along the contour of the box model. If an object is tracked through a point model, we

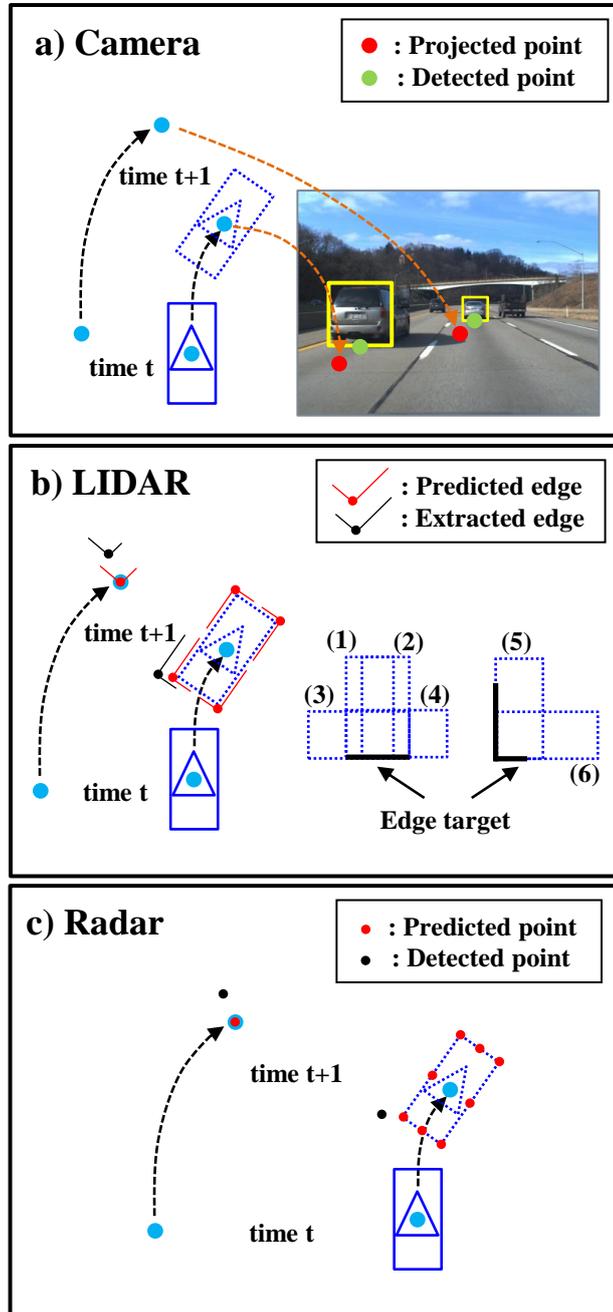


Figure 5.6: Illustration of data association methods for each sensor. (a) Camera, (b) LIDAR, and (c) Radar.

generate a single point. The association between the predicted and the actual measurement is made by the nearest-neighbor approach. Figure 5.6 (c) illustrates such a radar measurement association.

5.5.3.3 LIDAR

For the association of LIDAR observations (we call edge targets), we generate, based on the predicted moving object hypotheses, a set of possible alignments of edge targets. There are four alignments for a box model and one for each point model. The left side of Figure 5.6 (b) illustrates such an alignment generation. Similar to the camera measurement association, the extracted edge targets are associated to the closest predicted one that minimizes the distance of the corner points. If an extracted edge target is associated to a predicted box model, all possible interpretations of the edge target as the box model are generated as illustrated in the right side of Figure 5.6 (b). Among the interpretations, one that has the maximum overlap with the predicted box is chosen to generate the observation. In practice, however, we found that considering all possible interpretations of an edge target occasionally fails to correctly match edge targets.

To improve our earlier association method, we utilize the vision target. For example, when our vision object detector returns a highest response of a vehicle's rear view, we ignore irrelevant alignment (e.g., side-view alignments of edge targets). For example, in Figure 5.6(b), the alignments, (3), (4), and (6) are hypotheses about a vehicle's side-view and hence are ignored when the vision target casts a vote for a vehicle's rear-view.

5.5.4 Movement Classification

Knowing whether a detected object has non-zero motion is important to optimally estimate the state of a tracking object. This is particularly true for urban driving scenarios where there is frequent stop-and-go traffic, queuing at traffic signals, abnormal vehicle interactions, and so forth. In principle, a tracking system should be able to track trajectories of any moving objects around the host-vehicle. However, it is challenging to reliably track an object that was moving and now temporarily stops, but is going to move in the near future. To track such irreg-

ular temporal patterns, it is necessary to keep a record about series of motions as well as determining whether an object is in motion. To implement this idea, our previous system [29] introduced two movement flags about 1) the movement history, i.e., *observed moving* and *not observed moving* and 2) the movement state, i.e., *moving* and *not moving*. The flag *moving* is set when the tracking system decides the object is currently in motion. The flag *observed moving* is set when the tracking system determines that the position of a tracked object has significantly changed. For the classification of the current movement state, the direct movement observations from radars was used. Since LIDARs do not provide a direct movement confirmation, the statistical test which compares an objects estimated velocity with a threshold v_{min} was used. For the classification of the movement history state, the distance traveled is computed from the last time stamp that the object has been classified as *not observed moving*. Then this distance is compared with a threshold, $d_{traveled}$. In practice, it is very hard to set up a single set of parameters that works well for different object class. For example, parameters optimized for vehicles do not work well for pedestrians. Thus, during the development phase, we empirically found multiple sets of parameters that work optimally for each object class.

5.6 Experiments

To evaluate the performance of our new multi-sensor, object tracking system, we drove our robotic vehicle and collected data (i.e., images, radar points, and LIDAR scans) in about a 25-minute driving. The route is comprised of a mix of streets and inter-city highways between Carnegie Mellon University’s campus and Pittsburgh international airport. The distance is about 20miles. We first describe the system setup for the detection and tracking system and then discuss the evaluation results.

5.6.1 System Setup

Our tracking system runs on a computing cluster that consists of four mini-ITX, form-factor computers (i.e., Core 2 Extreme QX9300@2.53GHz, 8GB RAM).



Figure 5.7: Our experimental vehicle and its sub-components. (a) Autonomous Cadillac SRX [109], (b) Display and control console, (c) Computing cluster.

Each of the sensors generates its measurements at its own operating cycle (See Table 5.1). Software modules read measurements from individual sensors and publish them through an inter-process communication channel. While doing so, measurements acquired at a local coordinate system are converted into the host-vehicle’s global coordinate system. The sensors’ poses are calibrated with respect to the host vehicle’s coordinates system. Those reader tasks also perform a pre-processing of raw measurements to produce features (e.g., “L” shape from a point cloud) for object detector or tracker. Our tracking system is designed to run at 100Hz on a single machine on the computing cluster. In practice, however, the operating cycle varies based on the number of features. A typical latency, for example, is around 100 *ms* in highways and around 200 *ms* in urban environments. The maximum latency is fixed to 300 *ms*. The vehicle and described sub-components are shown in Figure 5.7.

For the LIDAR observation model, we used widths and lengths of three objects: $OW_{ped}=1\text{m}$, $OL_{ped}=1\text{m}$, $OW_{bike}=1\text{m}$, $OL_{bike}=1.7\text{m}$, and $OW_{veh}=2\text{m}$, $OL_{veh}=5\text{m}$. For the object management system, we begin to track an object when three consecutive measurements of that object are verified and stop to track the object when no observations are available for 400 *ms*. For the movement classification, we used $v_{ped_min}=0.5\text{m/s}$, $v_{bike_min}=1.0\text{m/s}$, and $v_{veh_min}=2.0\text{m/s}$ for *moving* classification and $d_{ped_traveled}=1\text{m}$, $d_{bike_traveled}=2\text{m}$, and $d_{veh_traveled}=4\text{m}$ for *observed moving* classification.

A vision sensor is installed in a forward-looking manner and acquires image frames of 640×480 at 8Hz. Those acquired images are fed to the system over a Gigabit Ethernet interface. To detect three objects (i.e., pedestrians, bicy-

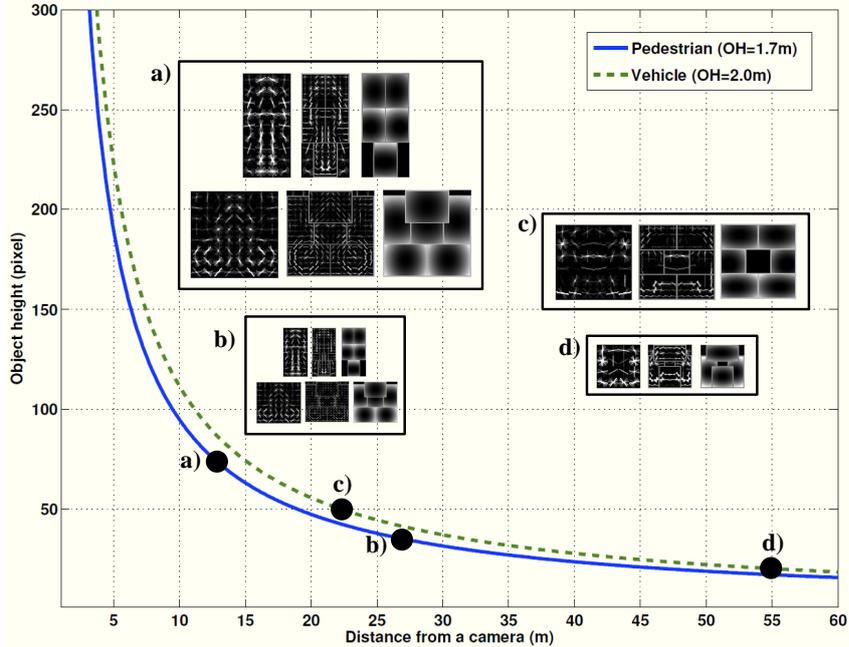


Figure 5.8: Pixel height in image space as a function of distance d from the camera. Based on this analysis, all models are designed and visualized here. (a) Normal-sized pedestrian/bicyclist model (72×32 with 8×8 HOG cell). (b) Small-sized pedestrian/bicyclist model (36×16 with 4×4 HOG cell). (c) Normal-sized vehicle model (48×48 with 8×8 HOG cell). (d) Small-sized vehicle model (16×16 with 4×4 HOG cell).

clists, and vehicles) from images, we used the real-time implementation [20] of the deformable part-based models [41] and produce corresponding vision targets.

To accurately determine the dimensions of objects' models, we computed those objects' pixel height with respect to the distance to our vehicle. From this analysis, we found that the dimension, 72×32 , is appropriate to detect pedestrians/bicyclists, reliably up to 13m. For the range between 13m and 26m, we trained a 36×16 pixel-sized model with a HOG cell size of 4×4 . Similarly, we trained two rear-view vehicle models, one a 48×48 sized model for the range up to 22m and the other a 16×16 sized model with a HOG cell size of 4×4 for the range between 22m and 55m. Figure 5.8 shows actual objects' models based on the distance to the vehicle.

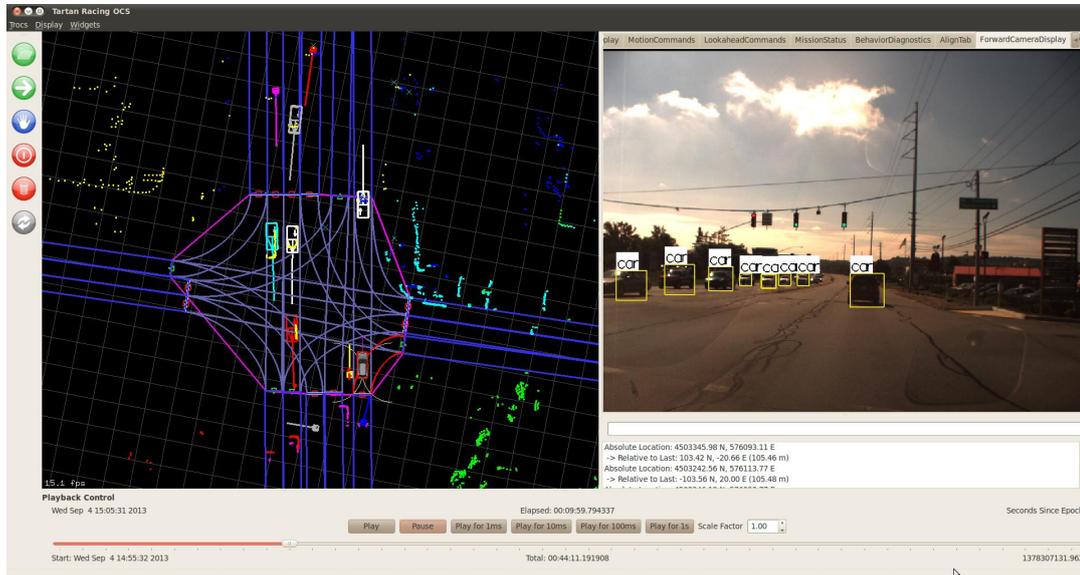


Figure 5.9: Screenshot of our evaluation tool [66]

5.6.2 Data Collection and Evaluation Methodology

For the quantitative performance evaluation, it is required to manually label each of the frames in the entire data set. Because this is labor-intensive and error-prone, we evaluate the performance differently. In particular, we had human annotators, using our evaluation tool [66] as shown in Figure 5.9, go over the tracking results second-by-second. While doing so, they counted the number of correctly (and incorrectly) tracked objects. Objects being considered for the evaluation include vehicles in a 150m radius of the host vehicle and pedestrians/bicyclists up to 20m on our vehicle’s path.

5.6.3 Tracking Results

Overall, our tracking system showed a good performance on the entire data. For example, when a vehicle is more than 150m away from our vehicle, the tracker begins to track the vehicle with a point model, and is able to switch the point model to a 3D box model when the tracked vehicle is less than 40m from the host vehicle. This is a desirable feature for other modules (e.g., a motion plan-

Table 5.2: Quantitative evaluation of our multi-sensor tracking system.
 Total Seconds - Session1: 900sec, Session2: 600sec.
 Total Objects - Session1: 1,762, Session2: 1,371

Dataset Section	Vsion Fusion	Correctly Tracked	Falsely Tracked	TPR (%)	False Positive Per Minute
Session 1	w/	1,585	183	89.9	12.2
(w/o RNDF)	w/o	1,466	208	83.2	13.9
Session 2	w/	1,285	57	93.7	5.7
(w/ RNDF)	w/o	1,238	79	90.3	7.9

ner) of self-driving vehicles because a host vehicle should know the exact (or approximately close) dimension of a moving object as the objects gets closer to the host vehicle. Figure 5.10 shows some examples of object tracking. a) and b) show pedestrian and bicyclist tracking results. From these examples, we found that our movement classification worked well to effectively track slow-moving and stop-and-go objects. For the case of c), LIDAR targets were reflected by walls of a tunnel. Because of this, our tracker tracked “ghost” targets with a point model. Despite this, because a vision target was available and associated with the target, our tracker was able to track the target with a 3D box, instead of tracking them with the point model. The cases between d) and h) in Figure 5.11 show some examples of vehicle tracking result on city roads and highways.

We investigated if the tracking performance is improved when a topological map is given. We also studied how much the performance was improved when the vision target is incorporated. Table 5.2 summarizes the experimental results. In short, the detection rate was 93.7% with 5.7 false positives per minute. All result videos for the entire route are available on our project website¹.

¹<http://users.ece.cmu.edu/~hyunggic/multiSensorFusion.html>

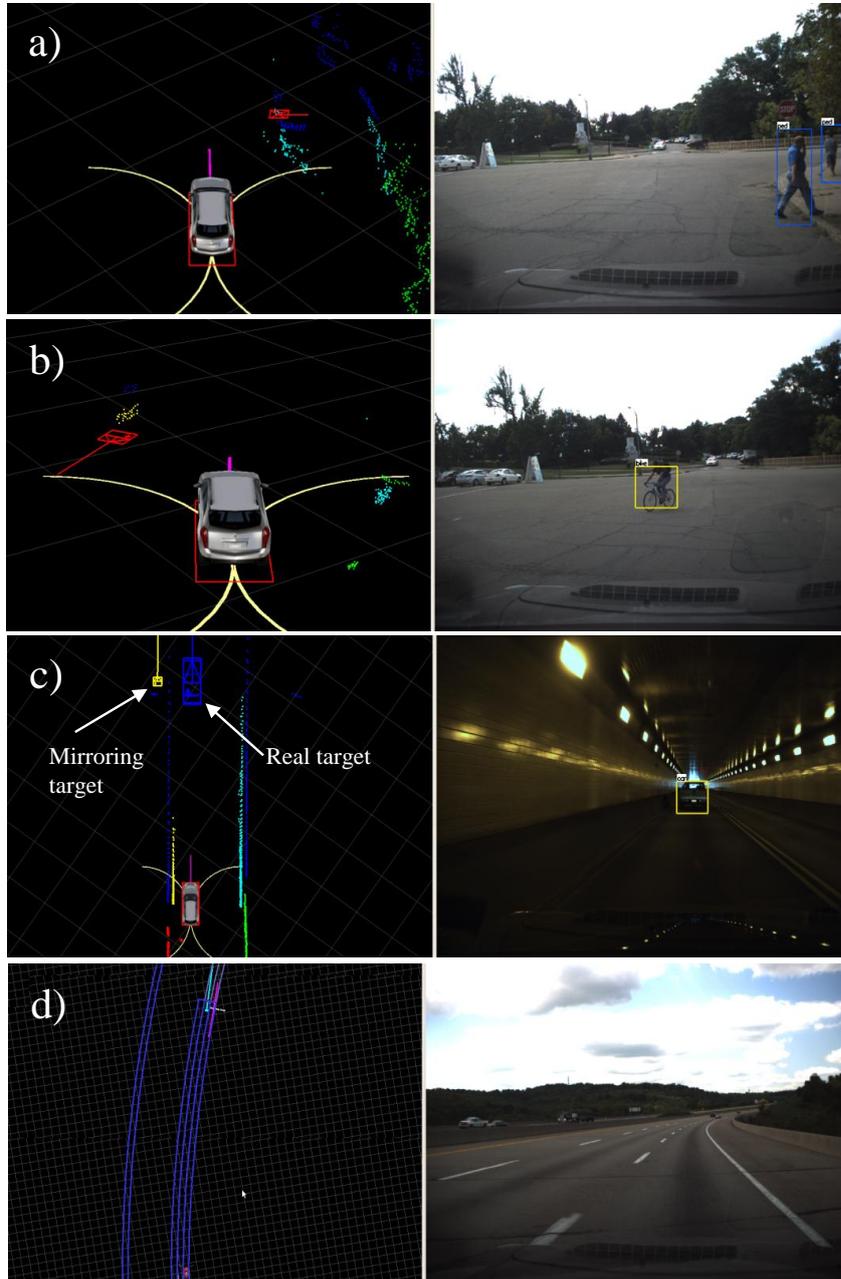


Figure 5.10: Typical tracking results for the qualitative evaluation. Tracking of a pedestrian (a) and a bicyclist (b), which was enabled by the vision recognition system. (c) Mirroring target issue (see text for the detail). (d) Tracking of a vehicle at far distance.

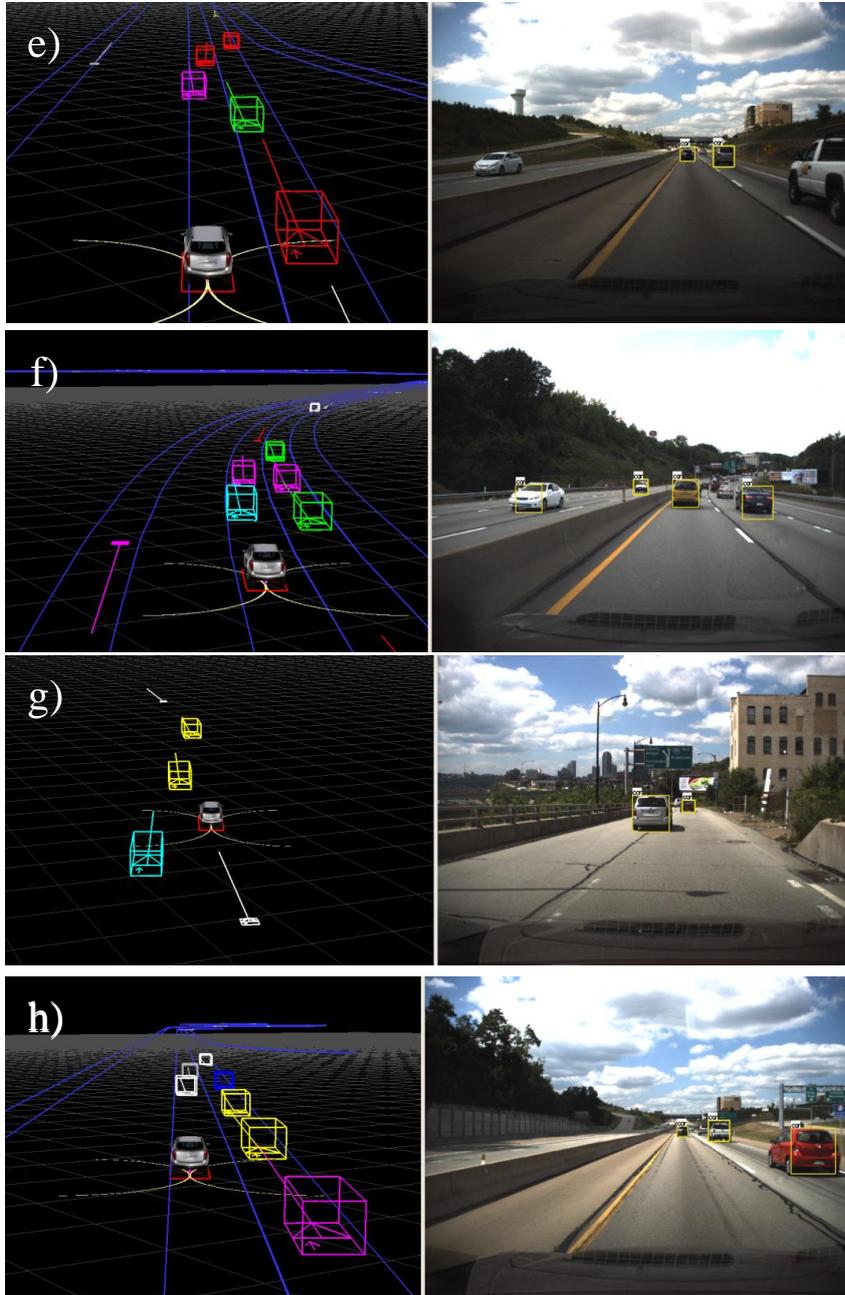


Figure 5.11: Typical tracking results for the qualitative evaluation (More examples). (e)~(h) Vehicle tracking results in various situations.

5.7 Summary

This chapter presented our new moving object detection and tracking system. To improve our earlier system, we re-designed sensor-configuration and installed multiple of radar and LIDAR pairs and two vision sensors. To seamlessly incorporate measurements in different modalities, we revised the previous motion and measurement models and introduced new models for vision measurements. In particular, by using vision's object class and shape information, our tracking system effectively switched between two motion models (i.e., a point and a 3D box models) based on objects' distances to our vehicle. The newly introduced vision targets were also useful to improve the performance of data association and movement classification for measurements from active sensors. Through the test using the data log of actual driving, we demonstrated the improvement and performance gain of our new tracking system.

Chapter 6

Toward a Holistic Approach to Moving Object Tracking

In this chapter, we describe a holistic approach which leverages contextual cues to further improve the performance of the multi-sensor tracking system presented in the previous chapter. As we learned that the semantic information from a vision recognition system can lead to better tracking performance, we posit that other traffic context cues such as lane markers and sidewalks detected can be exploited to improve the quality of object tracking. We verified this idea by integrating a lane marker detection system with our multi-sensor tracking system and demonstrating better orientation estimation of moving vehicles and better prediction of objects' future trajectories (corresponding to a few seconds) guided by the heading direction of the lane marker. Section 6.1 discusses the motivation for this approach and reviews related prior works. Section 6.2 describes how we design a model which captures contextual interplay between lane markers and moving objects. Section 6.3 details how we integrate lane marker detection results with the multi-sensor tracking system for better orientation and future trajectory estimation. Section 6.4 discusses the experimental results and Section 6.5 concludes this chapter.

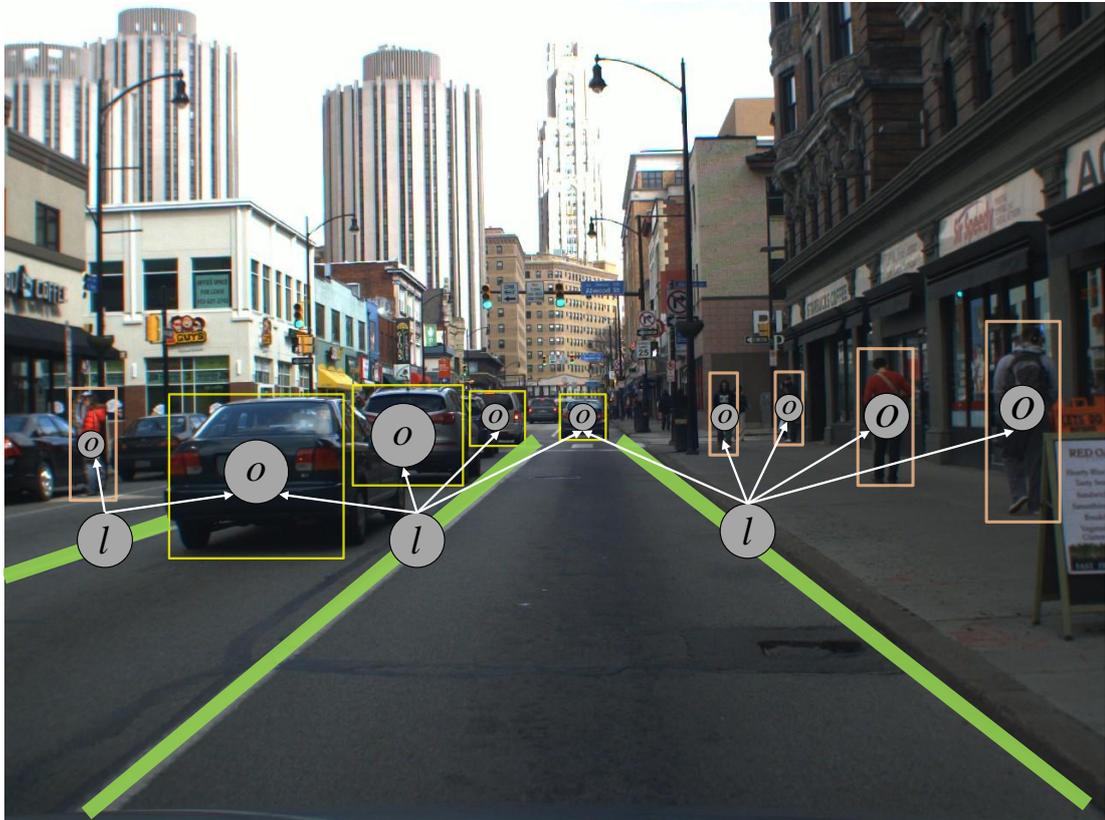


Figure 6.1: Contextual interactions between moving objects and traffic environments. Observe the relationship between a lane markers (l) and moving objects (o), especially vehicles. This relationship leads us to a simple and efficient probabilistic model for the traffic scene in an intuitive way.

6.1 Introduction

Traffic scene understanding requires estimating many different aspects of the scene. This includes, but not is limited to, estimating road geometry (e.g., lane width and curvature), detecting and tracking of moving objects, and recognizing traffic signs and lights. For the last few decades, each of these problems has been studied in isolation. Although impressive progress has been achieved in each domain, the idea of combining these tasks together in a way that improves the performance of each of them has not yet been fully exploited. For example, as can be inferred from Figure 6.1, knowledge about the road geometry from a lane

marker detection system can be used as a priori information for vehicle’s heading direction, and knowledge about the location of a sidewalk can be exploited for better pedestrian tracking, and information about traffic light state (i.e., red light or green light) can be also utilized for better vehicle tracking, and many more such scenarios.

This approach is indeed motivated by the use of vision sensors because cameras can provide a high-resolution view of the scene, which can be used for extracting different types of context cues, while a planar LIDAR or low-resolution scanning radar sensor is primarily used for just object detection and tracking as they cannot provide sufficient resolution to estimate the scene context. Therefore, this line of research for solving several vision sub-tasks jointly (also known as the problem of “holistic scene understanding”) became a hot research topic in computer vision [52, 51, 38] only recently. In [52], Hoiem et al. proposed a simple framework for synergistic integration of multiple vision algorithms by using feature maps (called ‘intrinsic images’) as an interface. They validated their framework by combining the sub-tasks of surface orientations, occlusion boundaries, object detection, and depth. Heitz et al. proposed CCMs (Cascaded Classification Models) [51] for a mechanism of combining models for holistic scene understanding. Main idea is having repeated instantiations of the classifiers, where their input/output variables are connected in a cascade structure. They demonstrated the effectiveness of their framework by combining the sub-tasks of scene categorization, object detection, image segmentation, and 3D reconstruction. In [38], Ess et al. proposed an approach which jointly estimates camera position, stereo depth, object detection, and tracking for multi-person tracking applications. They used a graphical model to capture the interplay among those components. However, unfortunately, those frameworks are not directly applicable to our multi-sensor fusion context mainly because their main goal is a holistic fusion for only vision tasks and more importantly, none of them was operating in real-time (i.e., takes several seconds to perform all necessary processing for a single frame on a modern PC). More recently, due to potential benefits of those approaches, many researchers in the intelligent vehicle community have tried similar approaches to real-time vehicle perception applications. For example, [82, 90] integrated a vision-based lane detection system and a vehicle detection system. Whereas Ponsa et al. [82] utilized

lane marker detection results to increase the accuracy of vehicle detection and also to reduce the processing time, Sivaraman et al. [90] exploited interplays between lane detection and vehicle detection so that each task can benefit from the results of other tasks. The work of Weigel et al. [110], which is quite similar to our approach, fused vehicle tracking using a LIDAR (i.e., IBEO LUX) with lane marking detection using a monocular camera.

In this chapter, we pursue a very efficient approach which utilizes contextual cues to further improve the performance of our multi-sensor tracking system. With all the contributions we made for the multi-sensor object tracking, what we want to exploit additionally are contextual interactions between moving objects and traffic environments. While the majority of classical approaches for object tracking try to estimate only the kinematic properties (e.g., position and velocity) of objects from sensor’s measurements, our new approach tries to improve the tracking performance by exploiting their relationship with the traffic scene context. In other word, we try to understand a moving object tracking problem in a whole context of traffic scene. The main context cue we want to investigate for moving object tracking is a road structure, specifically road lane markers. As shown in Figure 6.1, detection of lane markers can be utilized for constraining the orientations of moving vehicles to physically consistent ones with a heading direction of the lane marker detected near the tracked vehicles. Note that although our current work focuses on using contextual information provided by lane marker detection, similar approaches are possible for integrating sidewalk detection, traffic light detection, etc. In the next section, we discuss how to formulate the relationship in a formal way.

6.2 Problem Formulation

We use a simple probabilistic model to capture the interaction between moving vehicles and lane markers. Obviously, there are bi-directional interactions between the two, i.e., vehicle tracking results can be also exploited for better lane marker detection, but we want to use lane markers for constraining vehicle’s orientation estimation in this work. Further making the simplifying assumption that there is no interaction between moving vehicles leads to multiple instances

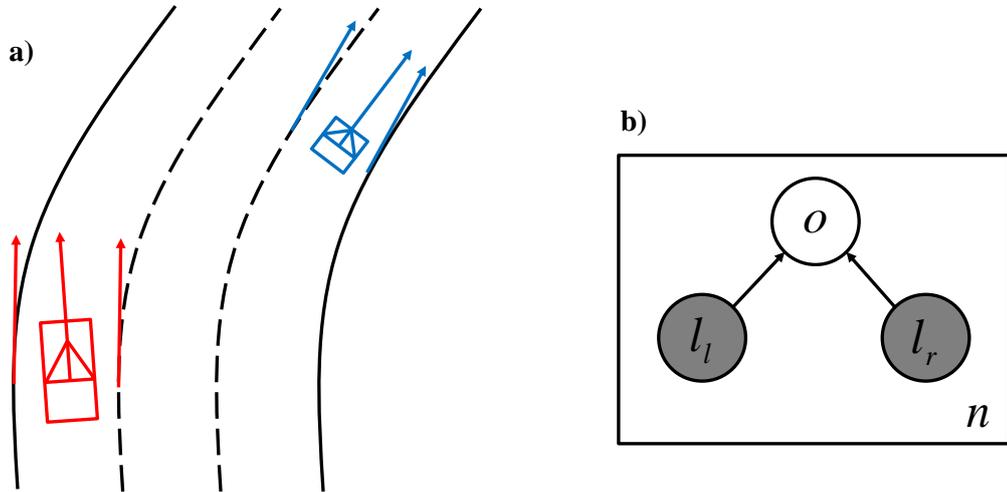


Figure 6.2: Graphical model for the problem formulation. This simple probabilistic model captures the relationship between moving vehicles (o_i) and lane marker detections (l_i). Shaded nodes indicate observable variables whereas the unshaded nodes indicate hidden variables.

(i.e., number of tracked vehicles) of a simple graphical model [56] as shown in Figure 6.2. We use the standard notation in a graphical model for repetition of the contained parts for the number of times specified at bottom right corner. For each graphical model, o indicates an orientation of a tracked vehicle and l_l and l_r indicates tangential direction of left and right lane markers, respectively.

To improve the estimate of an orientation of a tracked vehicle, we formulate it as a MAP (Maximum A Posteriori probability) estimation problem. Let us suppose that l_l and l_r are two i.i.d. (independent and identically distributed) random variables with $N(o, \sigma_l^2)$. We treat the orientation estimate of the tracked vehicle as a prior distribution of o , which is given by $N(o_0, \sigma_o^2)$. Then, our goal is to find the MAP estimate of o , which can be formulated as:

$$\begin{aligned}
 \hat{o}_{MAP}(l) &= \arg \max_o p(l|o)p(o) \\
 &= \arg \max_o \frac{1}{\sqrt{2\pi}\sigma_o} \exp\left(-\frac{1}{2}\left(\frac{o-o_0}{\sigma_o}\right)^2\right) \prod_{i=1}^2 \frac{1}{\sqrt{2\pi}\sigma_l} \exp\left(-\frac{1}{2}\left(\frac{l_i-o}{\sigma_l}\right)^2\right)
 \end{aligned} \tag{6.1}$$

After taking the log to the expression, above optimization is equivalent to minimizing the following function of o :

$$\sum_{i=1}^2 \left(\frac{l_i - o}{\sigma_l} \right)^2 + \left(\frac{o - o_0}{\sigma_o} \right)^2 \quad (6.2)$$

The MAP estimator for o can be analytically derived and is given by

$$\hat{o}_{MAP} = \frac{2\sigma_o^2}{2\sigma_o^2 + \sigma_l^2} \left(\frac{1}{2} \sum_{i=1}^2 l_i \right) + \frac{\sigma_l^2}{2\sigma_o^2 + \sigma_l^2} o_0 \quad (6.3)$$

which turns out to be a linear interpolation between the estimate of an orientation of the vehicle and the sample mean of the lane markers direction estimates, inversely weighted by their respective variances. Thanks to a simple structure of the problem formulation, this MAP estimation for all moving vehicles can be carried out in real-time inside our multi-sensor tracking system.

6.3 Integration with Lane Detection

To run the MAP estimation in the previous section, lane markers on the road should be first detected reliably. Since our goal is to improve the performance of moving object tracking rather than developing a new lane marker detection system, we make use of a commercial lane detection system [8]. In this section, we discuss the functional overview of a lane detection system and how we integrate the lane detection system with our multi-sensor tracking system.

6.3.1 Lane Detection

Since the problem of lane detection is a crucial component for Advanced Driver Assistance Systems (ADAS) [17], it has been an active field of research for the last two decades and considerable progress has been made in the last few years. For detailed surveys of the topic, we refer the readers to [65, 12]. Although specific algorithms for each module are different, most lane detection systems follow the “generic system” architecture presented in [12]. According to the authors, a

typical functional decomposition for lane marker detection systems is: image pre-processing, feature extraction, lane model fitting, temporal integration, and finding image to world correspondence. The first step in the pipeline is image pre-processing. There are several operations which can be applied to the image before feature extraction to reduce clutter and enhance the quality of features. Obstacle (i.e., vehicles and pedestrians) regions can be identified and removed. Shadows can be significantly suppressed using a preprocessing transformation applied to the entire image. Over and under exposure situations can be accounted for by image normalization or by actively controlling the camera exposure. Next, various features are extracted from the image to support lane detection. For lane detection, evidence for lane markers is collected. In the third step in the pipeline, a lane hypothesis is formed by fitting a lane model to the evidence gathered. And next, the lane hypothesis from the current frame is reconciled with lane hypotheses from the previous frame and with global positioning system (GPS) information, if available. Finally, image to world correspondence module provides mapping between image and ground coordinates, using assumptions about the ground structure and camera parameters.

6.3.2 Fusion with Lane Detection

The MobilEye lane detection system [8] we used for this work provides a list of parameters for geometric models of detected lanes as a final output. The lane model which is used in the system is a commonly used approximate model of the clothoid, given as follows:

$$\begin{aligned} L_l(x) &= c_3x^3 + c_2x^2 + c_1x + c_{0l} \\ L_r(x) &= c_3x^3 + c_2x^2 + c_1x + c_{0r} \end{aligned} \tag{6.4}$$

The basic description of the used variables is given in Table 6.1 and Figure 6.3 illustrates this lane model defined on the vehicle coordinate system. The MobilEye lane detection system usually detects as many as possible lanes including the “ego lane” (defined as the lane the ego-vehicle currently drives on). It usually detects three lanes, left, right adjacent lanes and the ego lane. To fuse these

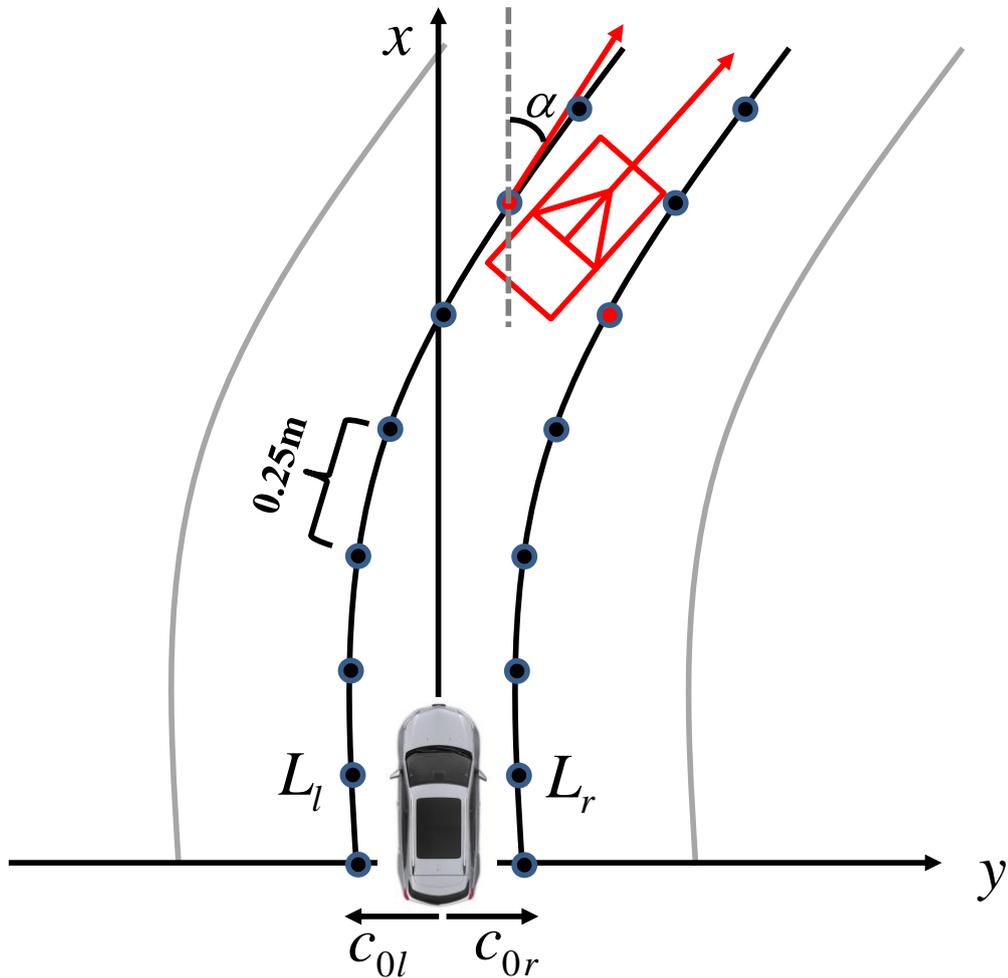


Figure 6.3: Illustration of lane association method for each vehicle target.

lane marker detection results with the current tracks of moving vehicles using the MAP estimator we described in the previous section, each tracked vehicle should be associated with a valid detected lane and then tangential heading angles should be extracted from the left and right lane markers. The tangential heading angles (i.e., l_l and l_r) together with associated variances are used as measurements for the MAP estimator. Next, we explain how we find a lane association to an existing vehicle target.

Let us suppose that an autonomous vehicle is driving on a road with three lanes as shown in Figure 6.3. First, given the parametric models of four lane

Variable	Meaning
L_l, L_r	Left and right lane markers
x	Longitudinal distance
c_3	Derivative of curvature
c_2	Curvature
c_1	Lane heading angle
c_{0l}, c_{0r}	Left and right offset

Table 6.1: Variables used for the lane marker model and their meaning

markers (so, we have three lanes), a set of waypoints along the each lane marker are generated at distance intervals of span of 0.25m. Since our tracking performs in the global coordinate system, the coordinates of the waypoints should also be converted into the global coordinate system using the pose of the ego-vehicle. Two lane (marker) associations (i.e., left and right lane marker) can be found by finding a waypoint (on each lane marker) that has the minimum distance to the center point of the tracked vehicle. We treat the closest two waypoints to the tracked vehicle as waypoints associated to that vehicle. Finally, the tangential heading angles at the associated waypoints are easily computed by taking gradient of the curve at that point, i.e., $L'(x) = \tan \alpha = 3c_3x^2 + 2c_2x + c_1$. Furthermore, the lane markers associated to the vehicle target can be utilized to predict a future trajectory of the vehicle reliably. By predicting the vehicle’s trajectory guided by the associated lane markers rather than using its current velocity (i.e., point model) or orientation (i.e., box model) state information, reliable prediction to a few seconds ahead (depending on the view range of the lane marker) can be achieved.

6.4 Experiments

To evaluate the performance of our new fusion module with the lane marker detection system, we collected new multi-sensor data (i.e., images, radar points, and LIDAR scans) similar to the data we used in the previous chapter. This new data set was collected during a 44-minute drive from Cranberry Township

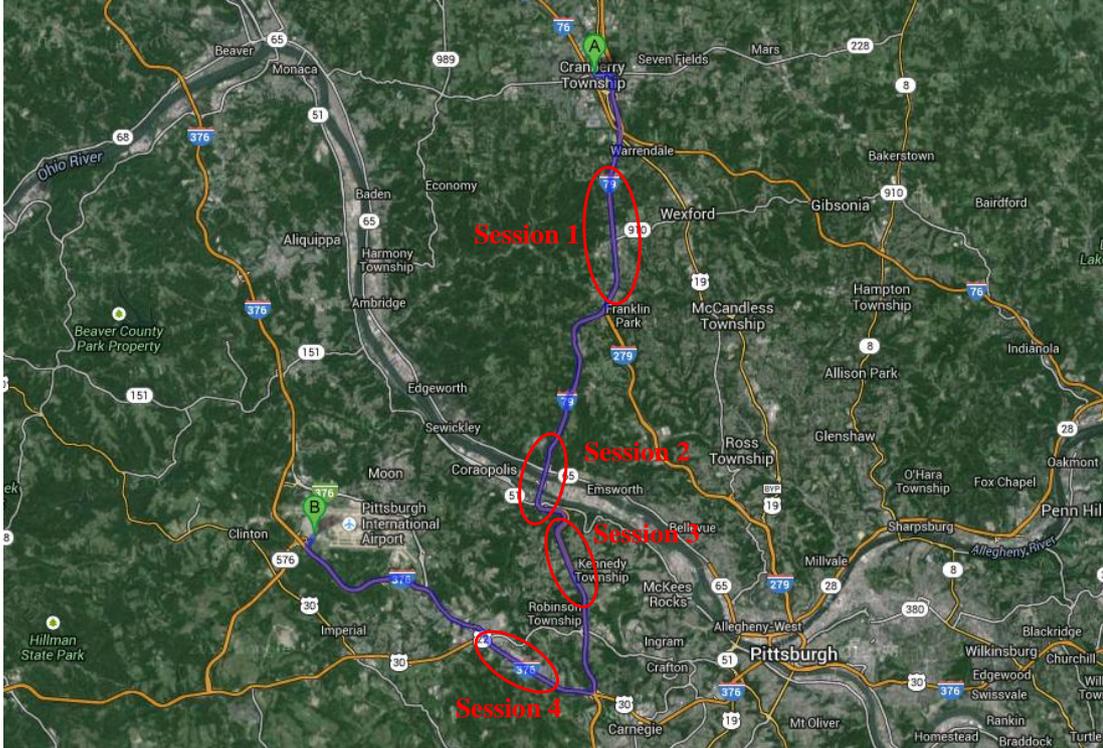


Figure 6.4: Route for the evaluation. Four different straight road segments (Session1~Session4) were specified for a quantitative evaluation of orientation estimation.

to Pittsburgh international airport as shown in Figure 6.4. It also contains lane marker detection results from the MobilEye lane detection system [8]. We discuss the evaluation results of performance gain from the fusion with the lane marker detection system.

6.4.1 Qualitative Evaluation of Lane Detection System

Since the lane marker detection system we used in this work is a commercial system widely used in the automotive area, we did not perform a quantitative analysis on its performance. Instead, after we integrated the lane detection system into our vehicle platform (runs at 10Hz), we evaluated its performance qualitatively. In the absence of dense traffic, the system usually can detect all existing lane markers reliably as shown in Figure 6.5 (a)~(c) and 6.6 (d). Typical max-

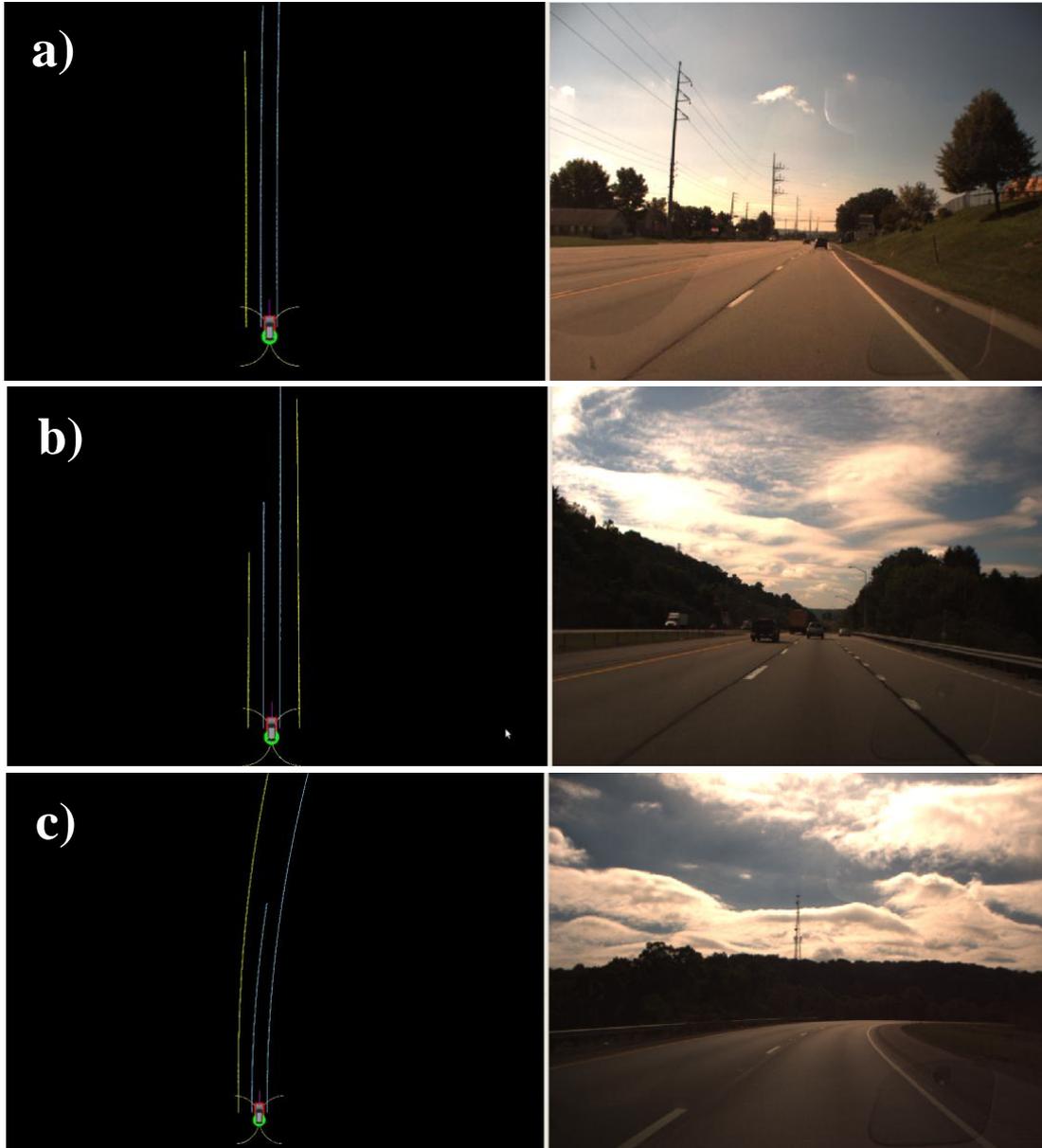


Figure 6.5: Qualitative evaluation of the lane marker detection system.

imum view range is around 90m in that case. However, if there is dense traffic (i.e., moving vehicles), the number of detected lane markers are reduced accordingly and view range distance is significantly reduced depending on the distance of a leading vehicle. This case is shown in Figure 6.6 (e). The system also pro-

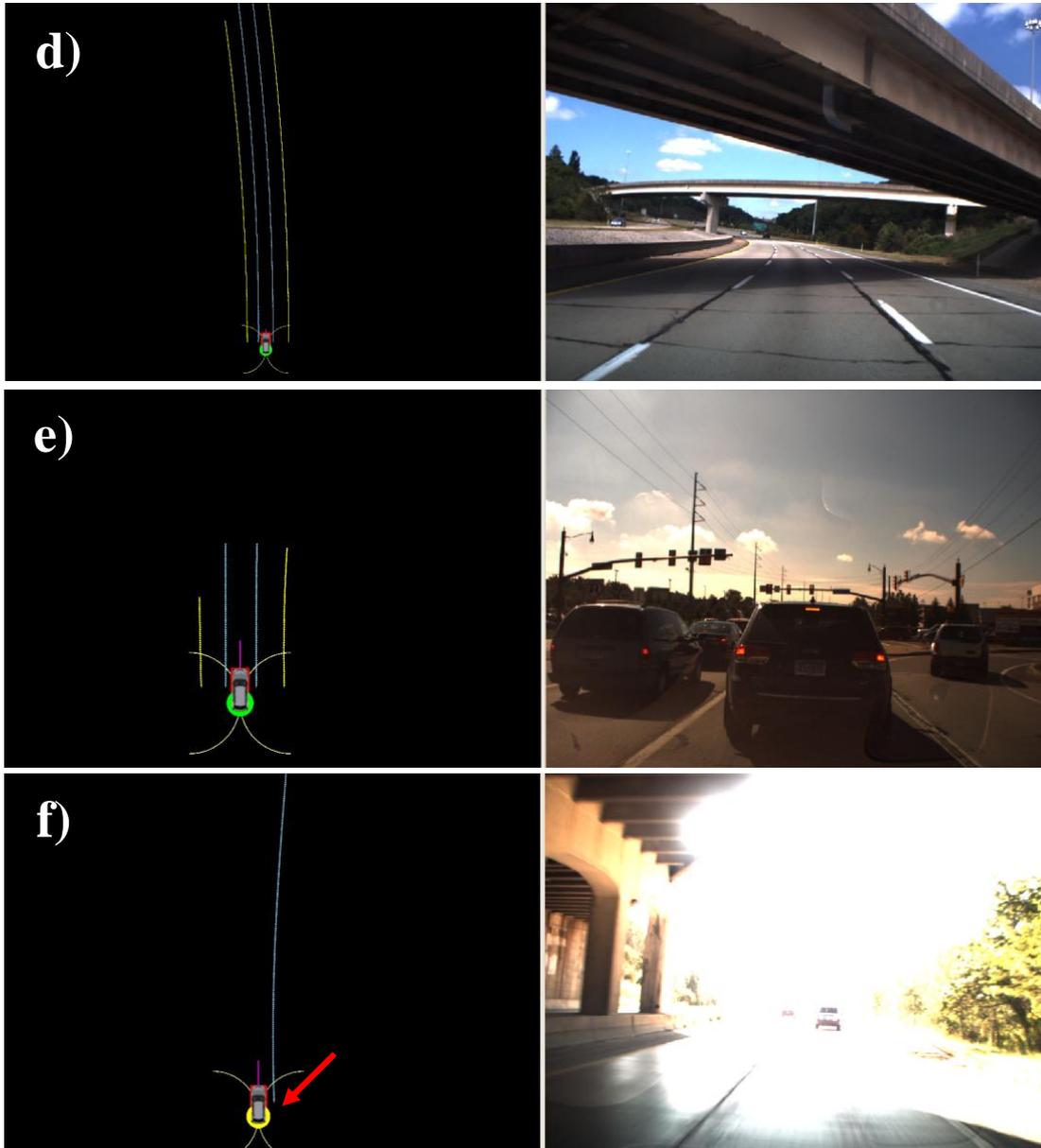


Figure 6.6: Qualitative evaluation of the lane marker detection system (More examples).

vides a measure for the quality of lane estimation. When the system experiences some exceptional conditions such as over exposure due to the strong sun glare as shown in Figure 6.6 (f), a confidence probability of the lane estimation goes

Table 6.2: Quantitative evaluation on orientation estimation (Unit: degree).

Dataset Section	Mean w/o Lane	Std. Dev. w/o Lane	Mean w/ Lane	Std. Dev. w/ Lane
Session 1 (140 sec)	2.02	0.63	0.12	0.09
Session 2 (60 sec)	1.87	0.57	0.10	0.075
Session 3 (75 sec)	1.91	0.87	0.14	0.11
Session 4 (130 sec)	2.45	0.92	0.17	0.13

down below to 0.5 (shown as a yellow circle under the ego-vehicle, otherwise it is a green circle), indicating that the output of the lane detection system is no longer reliable.

6.4.2 Evaluation of Orientation Estimation

As discussed in Section 6.3, the main goal of integrating the lane detection system is to improve the orientation estimation of moving vehicles. Since we do not have a ground truth mechanism for each tracked vehicle, it is quite challenging to evaluate the performance gain of the fusion module. To achieve the goal, we devised a somewhat special evaluation methodology. We chose to utilize the vehicle state information of the ego-vehicle for the ground truth since our localization system relies on highly accurate GPS/IMU system [109]. However, the ego-vehicle’s state information is not the same as that of the tracked vehicles. Thus, we carefully selected four different “straight” road segments (specified by the red ellipses in Figure 6.4) so that we can use the orientation of the ego-vehicle as a ground truth for leading vehicles. Although selected road segments are not perfectly straight lines, we verified from the experiments that the errors from that fact are negligible compared to the estimation errors. For a quantitative evaluation, we considered only the tracked vehicle on the ego-lane and recorded orientation estimates of the tracked vehicle. Table 6.2 shows the mean absolute

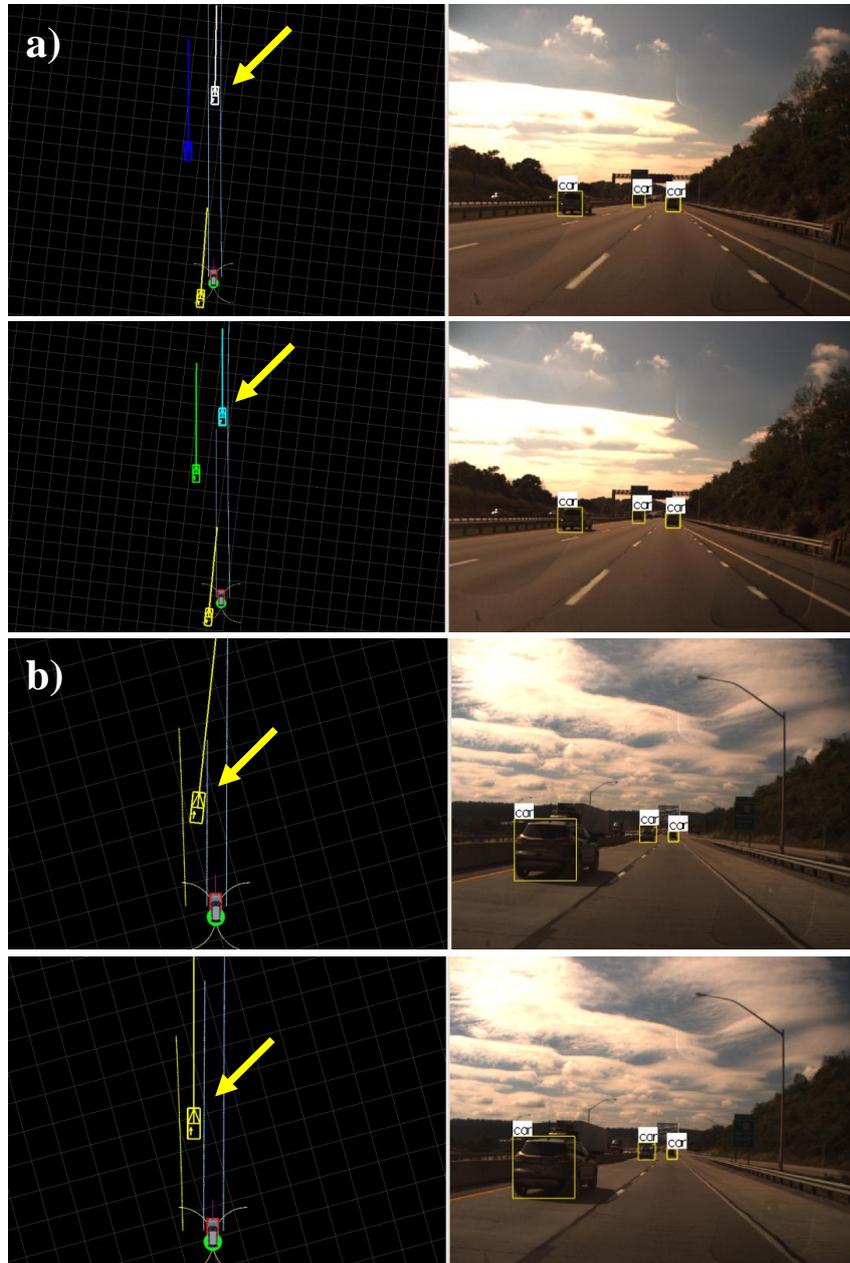


Figure 6.7: Qualitative evaluation of vehicle orientation estimation. (a) Case of correct orientation estimation. (b) Case of incorrect orientation estimation. In each case, the upper (lower) figure shows a result without (with) lane marker fusion.

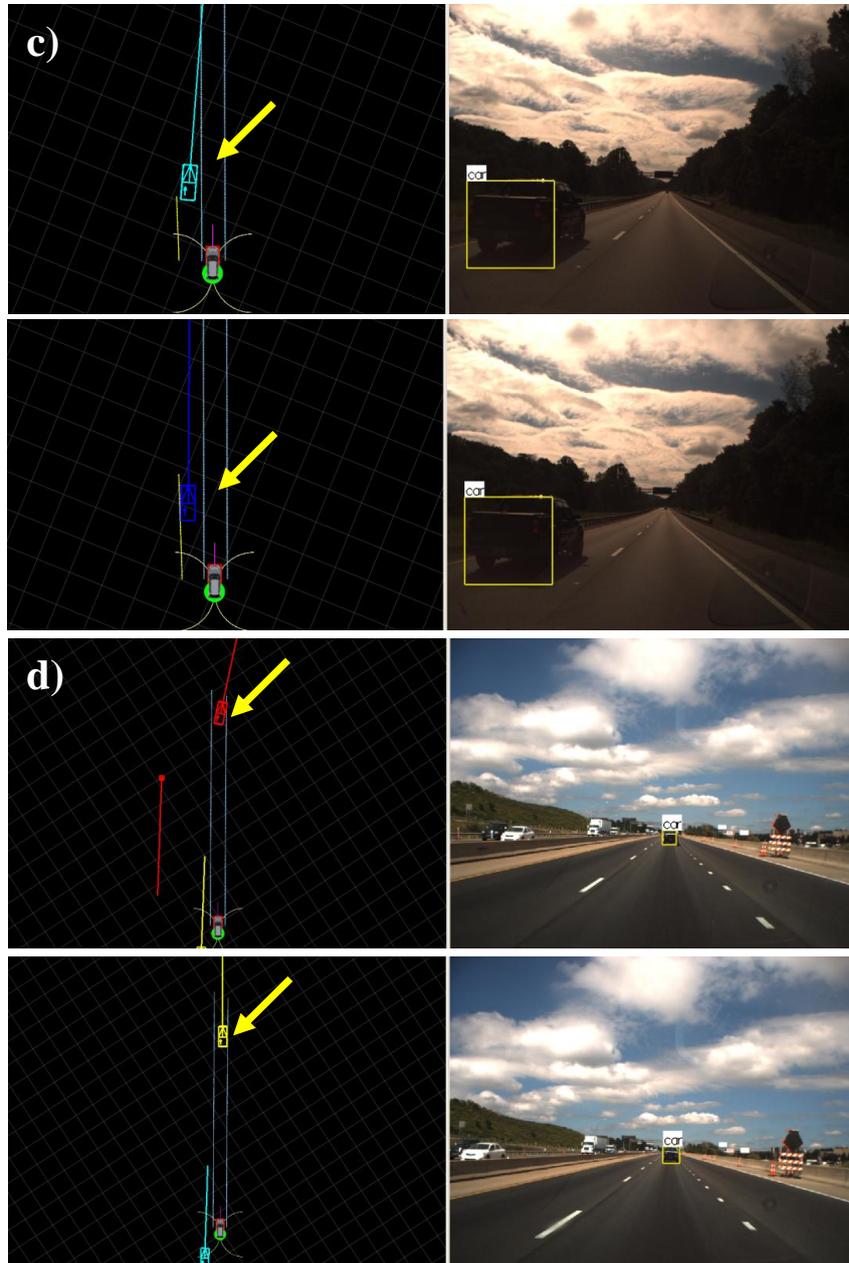


Figure 6.8: Qualitative evaluation of vehicle orientation estimation. (c) and (d) show more cases of incorrect orientation estimation.

error and standard deviation of the orientation error over each session with or without the fusion of lane markers. As shown, our fusion module with lane mark-

ers significantly reduces the estimation error. In addition, we looked into the worst orientation estimation errors for both cases, which might be important for our applications. The worst case orientation errors are 12.45 and 2.86 degree for without and with fusion, respectively. Typical results with or without the fusion module are shown in Figure 6.7. In many cases, since the multi-sensor tracking system tracks moving vehicles very well, its orientation estimates for the vehicles are also quite accurate, so we do not see a large discrepancy frequently as shown in Figure 6.7 (a). But, in some cases, the tracking system produces quite large errors on the orientation estimation where our holistic fusion module kicks in. Such examples are shown in Figure 6.7 (b) and 6.8 (c) and (d). In the case of (b) and (c), an SUV on the left lane overtook the ego-vehicle quickly and the tracker produced wrong orientation estimates for a short period of time, but with the fusion, the estimation was properly corrected. In (d), since the leading vehicle was too far away from the ego-vehicle, the tracker mis-estimated its orientation, but with the fusion, it could be fixed easily.

6.4.3 Evaluation on Trajectory Prediction

The additional benefit of our holistic approach exploiting lane marker detection for moving vehicle tracking is that we can actually predict future trajectories of tracked vehicles based on the detected lane markers as discussed in the previous section. First, along the lane marker, a set of waypoints is generated and then tangential heading direction on each waypoint is used for better prediction of the moving vehicle. Since we are interested in the scenarios where a couple of vehicles drive in front of the ego-vehicle, typical view range of the lane detection system was roughly around 60m. Thus, a usual prediction window is one or two seconds. Some typical results are shown in Figure 6.9 and 6.10. In (a), without lane marker fusion, trajectory prediction was incorrect since it is based on the current velocity of the target vehicle. With fusion, however, the prediction was smooth and correct. In (b), interestingly, the orientation estimation was correct, but since the road was curvy, the prediction based on lane markers was more accurate. More similar examples are shown in (c) and (d).

6.5 Summary

This chapter presented a holistic approach for the moving object tracking system we developed in Chapter 5. For lane detection, we used MobilEye’s proprietary lane detection system [8]. Following the very basic fact of ‘cars follow roads’, the idea of fusing the vehicle’s orientation estimate with tangential lane directions was formulated as a MAP estimation problem. MobilEye’s lane marker detection system was qualitatively evaluated on different scenarios. Then, the fusion with lane markers was quantitatively evaluated using four log sessions. Finally, trajectory prediction module was also properly evaluated. Our experiments show that the holistic approach significantly improves the estimation of vehicles orientation and prediction of vehicles future trajectories.

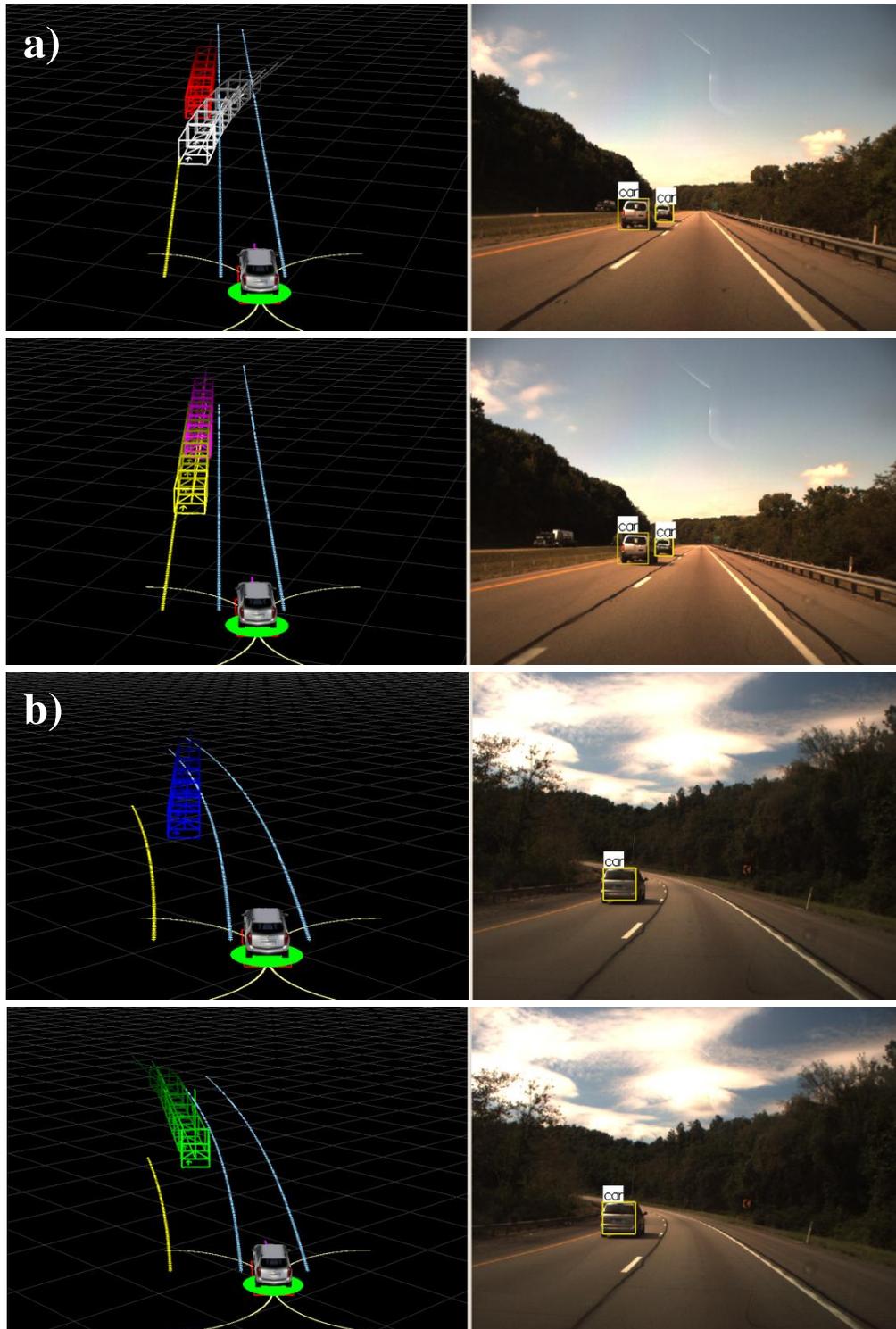


Figure 6.9: Qualitative evaluation on vehicle trajectory prediction. In each case, the upper (lower) figure shows a result without (with) lane marker fusion.

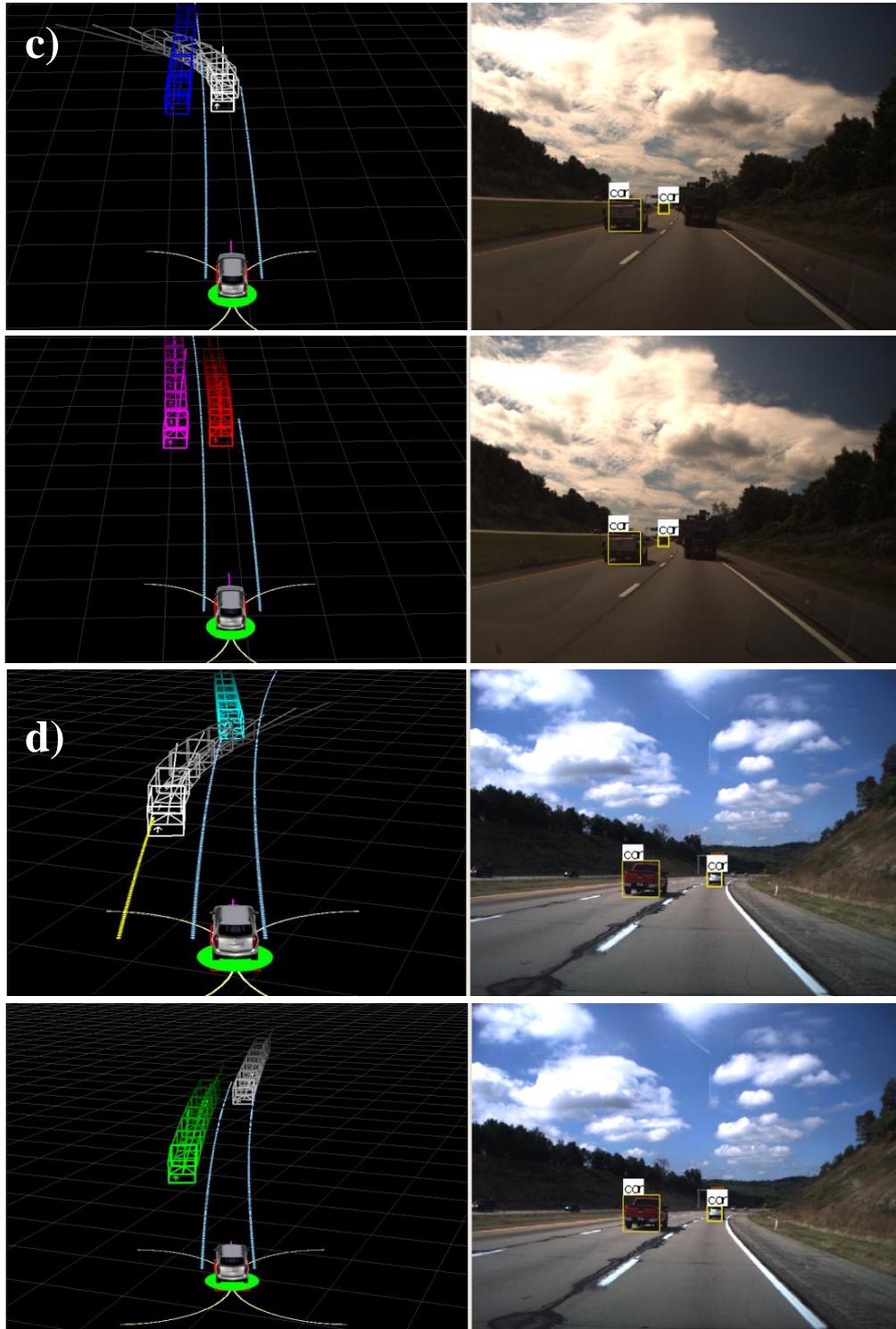


Figure 6.10: Qualitative evaluation on vehicle trajectory prediction (More examples).

Chapter 7

Conclusions and Perspectives

This thesis has described a multi-sensor fusion system for the problem of detecting and tracking of moving objects in highway and urban traffic environments. The proposed tracking system integrates the vision-based object detection system [21, 22, 20], active sensor-based (e.g., radars and LIDARs) tracking system [29, 23], and fusion with a lane marker detection system [8] to generate a composite model of moving objects. In contrast to the majority of tracking systems used in current self-driving vehicles [102, 71, 60, 101, 16], we quantitatively investigated vision-based object category detection to increase the recognition capability for various urban objects such as pedestrians, bicyclists, and vehicles. This semantic information from the vision system brings new synergistic benefits to the active sensor-based tracking, where the information is exploited to improve performance of the key components of the tracking system such as model selection, data association, and movement classification. Furthermore, the vision-based lane marker detection system [8] is also integrated to improve orientation estimation of moving vehicles and to enable more reliable prediction of vehicles' future trajectories. Although it is still at a primitive level of "holistic scene understanding" approach, our fusion method with lane detection enables a self-driving vehicle to track nearby moving objects consistently without compromising its real-time performance. Through the test using the challenging data log collected from our autonomous vehicle [109], we have demonstrated the improvement and performance gain of the proposed tracking system.

7.1 Key Contributions

- **Improvement of the DPM.** In this thesis, we significantly improved the speed performance of ‘deformable part-based models.’ [41, 40] by exploiting parallelism based on multi-cores and known scene geometry. In addition, we significantly improved pedestrian and vehicle detection accuracy by designing more optimized object models for automotive applications. This effort allowed us to run one-component pedestrian model at the speed of 14 Hz (two-component bicyclist model at the speed of 8 Hz) on our autonomous vehicle. We also studied the part-centric motion of pedestrians which we defined as ‘*inner motion*’. The *inner motion feature* together with the appearance feature (i.e., HOG features in our case) lead to the state-of-the-art pedestrian detection performance (39% log-average miss rate in the ‘*reasonable*’ scenario, see Section 3.5 for details).
- **Data association for multi-object tracking.** We studied a new data association algorithm for multi-object tracking. It incorporates a Rao-Blackwellized Particle Filter which runs a particle filter for data association and an EKF for each object tracking. We applied the algorithm to the problem of vision-based object tracking by improving the measurement likelihood computation which takes advantage of rich appearance information from images. Our experiments on three different urban objects show its superior tracking performance.
- **Multi-sensor fusion system.** The culmination of this thesis is a novel sensor fusion system which exploits high-level semantic information from vision-based object detection with intermediate features from radar and LIDAR sensors. This object class information is utilized as a key factor in several sub-components of the tracking system such as model selection, data association, and movement classification. A more appropriate tracking model (either a box model or a point model) is selected depending on the object class information and quality of the sensor data. In addition, when a LIDAR’s edge feature is associated with an object being tracked with a box model, the class information is used for pruning unnecessary interpretation

of the edge feature for vehicles. Finally, a different parameter set can be used for movement classification based on an object type.

- **Fusion with lane marker detection.** Lane marker information from a proprietary lane marker detection system is exploited to improve tracking performance of our multi-sensor tracking system. The fusion brings two benefits to the tracking system. It not only improves orientation estimation of moving vehicles by fusing the tangential heading angle of the associated lane marker but also enables the prediction of future trajectories of the moving vehicles.

7.2 Limitations

The proposed fusion system has some limitations. A few of these are intrinsic limitations and others are practical limitations due to limited computation power, sparse measurements and limited coverage of sensors. This section presents some of these limitations and the next section provides a discussion of the future work.

- Lack of capability in the DPM to deal with multiple models is the first intrinsic limitation. Even though the DPM is equipped with deformable parts and view-based representation (so called “components”) for the intra-class variation, its expression power for general objects is still limited (e.g., distinguishing between an adult pedestrian and a sitting child). The main practical issue of the DPM is its high computational cost. Although we demonstrated real-time performance (i.e., 14 Hz with a pedestrian model) of the DPM detector on a modern PC, it still requires significant speed improvement because such a detection system needs to run multiple models of different object categories (e.g., pedestrians, bicyclists, motorcyclists, dogs, strollers, and possibly many more). In addition, the detection system should be able to operate in real-time on an automotive-grade embedded system eventually.
- Regarding to the multi-sensor fusion system, different sensors produce different kinds of misinterpretations of measurements and erroneous detections, so called ‘*sensor artifacts*’. These artifacts pose lots of challenges to

reliable moving object tracking. For example, a manhole cover in the middle of a road appears as a strong obstacle to radar sensors, and curbs, vegetation, and guardrails at road boundaries are frequently picked up as strong measurements to LIDAR sensors. It is well-known that vision-based object detection suffers from intermittent false positives. Some of these artifacts are suppressed by the proposed multi-sensor fusion approach (e.g., manhole cover detection suppressed by LIDAR measurements or mirroring LIDAR target suppressed by vision detections as discussed in Chapter 5.6), but some artifacts still remain, causing either false tracking or tracking failures.

- Our holistic approach between the multi-sensor tracking system and lane marker detection system is not mutually beneficial. Since the lane detection system we used is a proprietary system, we could not access the internal algorithms for the lane detection system. Thus, interaction was uni-directional, utilizing the lane markers in the tracking system to improve the orientation estimation of tracked vehicles. Indeed, results from the tracking system, i.e., the moving object list, can be utilized to improve performance of the lane detection algorithm conceptually.

7.3 Future Work

As future work, it would be interesting to further explore the idea of holistic scene understanding. In this thesis, we exploited the interplay between moving objects and lane markers on the roads, but we believe that there is a broader range of background cues to support this approach, resulting in mutual benefits. For example, as discussed briefly in the introduction of Chapter 6, a sidewalk can be utilized for better pedestrian tracking and traffic lights also can be exploited for better velocity estimation of slow moving vehicles. In addition, a vision-based vehicle detection task can exploit lane marker detection results to reduce the search space and traffic sign recognition results can be utilized for better vehicle tracking. These contextual interactions for holistic approach are listed in Table 7.1. This line of research has potential for autonomous vehicles to have a higher-level reasoning capability to support contextual inferences.

Contextual Cues	Challenges
Vehicle tracking and lane markers	Investigated in this study
Pedestrian tracking and sidewalk	Unexpected pedestrian movement, e.g., jaywalking
Traffic light and slow moving vehicles	Efficient model switching
Vehicle detection and lane markers	Model derivation
Vehicle tracking and traffic signs	Relationship identification

Table 7.1: More examples of contextual cues and the possible challenges for integration.

It is also worth investigating a hardware implementation of the DPM object detector to deploy the system in some real automotive settings. As we analyzed in Chapter 3, two significant computational bottlenecks for the detection pipeline are a) HOG feature computation and b) a number of cross-correlations. Since these two main functions take up the majority of the needed computation, we can achieve the goal relatively easily by implementing only these two functions using contemporary FPGAs (Field-Programmable Gate Array).

Appendix A. Data Sets

One of the important goal of this thesis is to implement a robust visual recognition system for urban traffic objects such as pedestrians, bicyclists, motorcyclists, and vehicles. To build such a robust vision-based object recognition system, all object models should be trained on appropriate data sets which reflect real characteristics of the real environments. For this purpose, we collected a large amount of object data including object categories of bicyclist, motorcyclists, and vehicles. For pedestrians, we used the Caltech Pedestrian data set since it is one of the largest pedestrian data sets as of this publication. For a statistics analysis for the Caltech Pedestrian data set, we refer the reader to [33]. This appendix describes the data set we collected for our evaluation and implementation.

.1 Bicycle Data Set

For bicycle detection, we collected a new bicycle dataset in various places and weather conditions. The collection scenarios are based on the analysis of bicycle collision statistics [53]. The dataset was captured at 3fps following the lessons in [20] and consists of two sessions (S0~S1) for training and two sessions (S2~S3) for testing, each with 1 or 2 video files and associated annotation files. Due to our purpose, most of samples corresponds to rear view and side view of samples of bicyclist. Figure 1 and 2 shows some sample images with and without ground truth bounding boxes, respectively. In Figure 3, the left pie chart shows the distribution of the portion of bicycle samples for each viewpoint and right-hand side figure shows the distribution of bottom lines of all (i.e., 4,284) bounding boxes. The number of bounding boxes for each viewpoint is shown in Table 2.

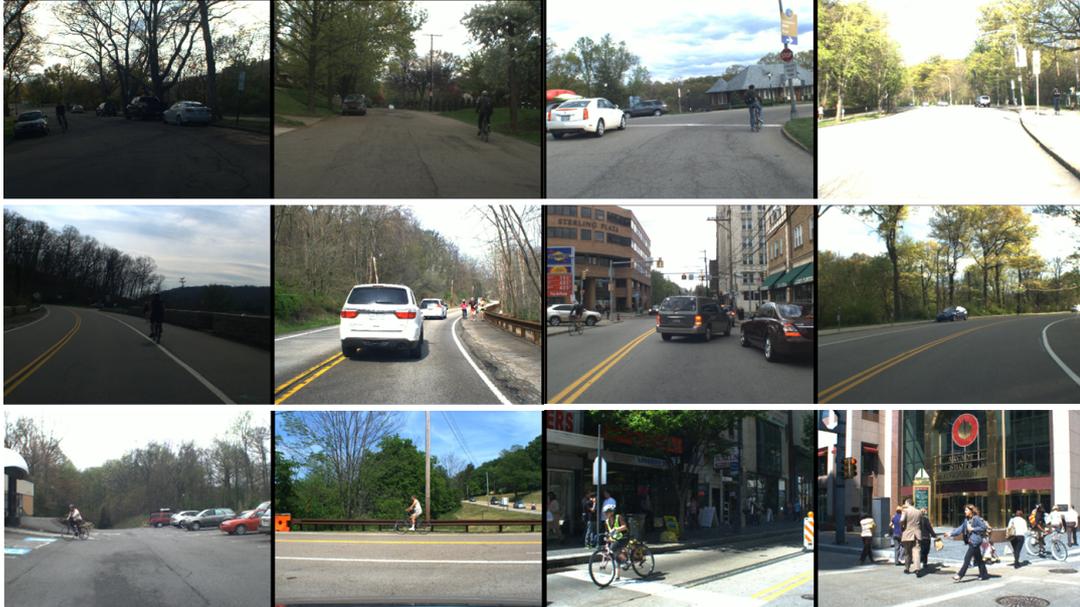


Figure 1: Some sample images from the bicycle data set.



Figure 2: Some sample images with ground truth bounding boxes.

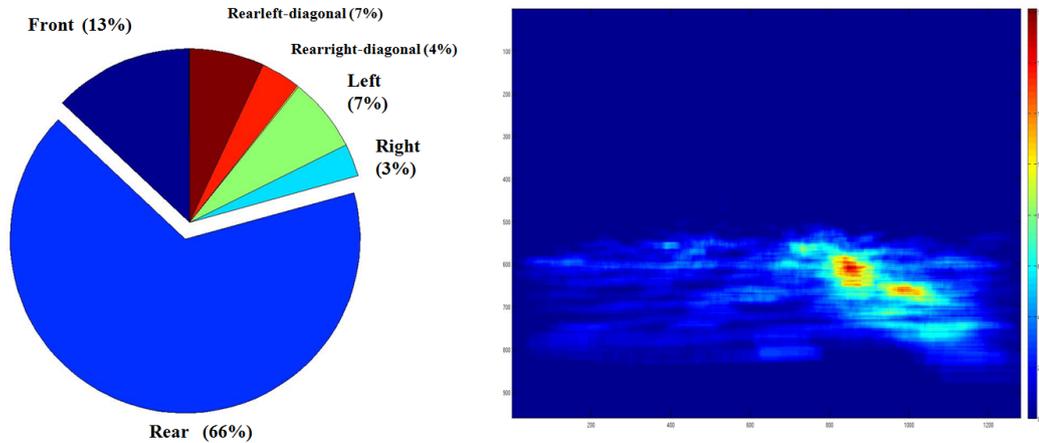


Figure 3: Distribution of bicycle samples (left) and distribution of bottom lines of all bounding boxes (right).

Profile	No. of Bounding Boxes
Rear	2,845
Front	553
Right	129
Left	298

Table 2: Bounding box distribution.

.2 Motorcycle Data Set

The motorcyclist DPM was designed and trained especially for automotive active safety applications such as FCW (Forward Collision Warning). For this goal, we collected motorcycle data set in various places. The dataset was captured at 3fps and consists of one training session (S0) and one testing session (S1). Same as the bicycle data set, most of samples corresponds to rear view samples of motorcycles. Figure 4 and 5 shows some sample images with and without ground truth bounding boxes, respectively. In Figure 6, the left pie chart shows the distribution of the portion of motorcycle samples for each viewpoint and right-hand side figure shows the distribution of bottom lines of all (i.e., 5,276) bounding boxes. The number of bounding boxes for each viewpoint is shown in



Figure 4: Some sample images from the motorcycle data set.



Figure 5: Some sample images with ground truth bounding boxes.

Table 3.

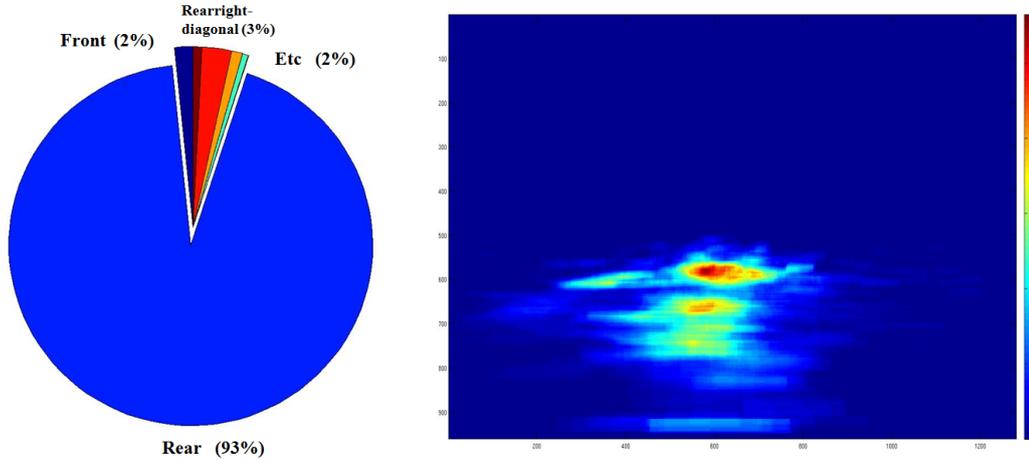


Figure 6: Distribution of motorcycle samples (left) and distribution of bottom lines of all bounding boxes (right).

Profile	No. of Bounding Boxes
Rear	4,927
Front	85
Right	3
Left	27

Table 3: Bounding box distribution.

.3 Vehicle Data Set

For vehicle detection, we collected a new vehicle dataset based on the analysis of vehicle collision statistics [76]. It consists of three training sessions (S0~S2) and two testing sessions (S3~S4), each with 1 or 2 video clips and associated annotation files. Due to our purpose, most of samples corresponds to rear view samples of vehicles such as sedan, SUV (Sports Utility Vehicle), truck and bus. Figure 7 and 8 shows some sample images with and without ground truth bounding boxes, respectively. In Figure 9, the left pie chart shows the distribution of the portion of vehicle samples for each sub-type and right-hand side figure shows



Figure 7: Some sample images from the vehicle data set.

the distribution of bottom lines of all (i.e., 14,453) vehicle bounding boxes. The number of bounding boxes for each vehicle sub-type is shown in Table 4.

Vehicle sub-type	No. of Bounding Boxes
Sedan Rear	5,976
SUV Rear	4,344
Truck Rear	390
Bus Rear	299

Table 4: Bounding box distribution for each vehicle sub-type.

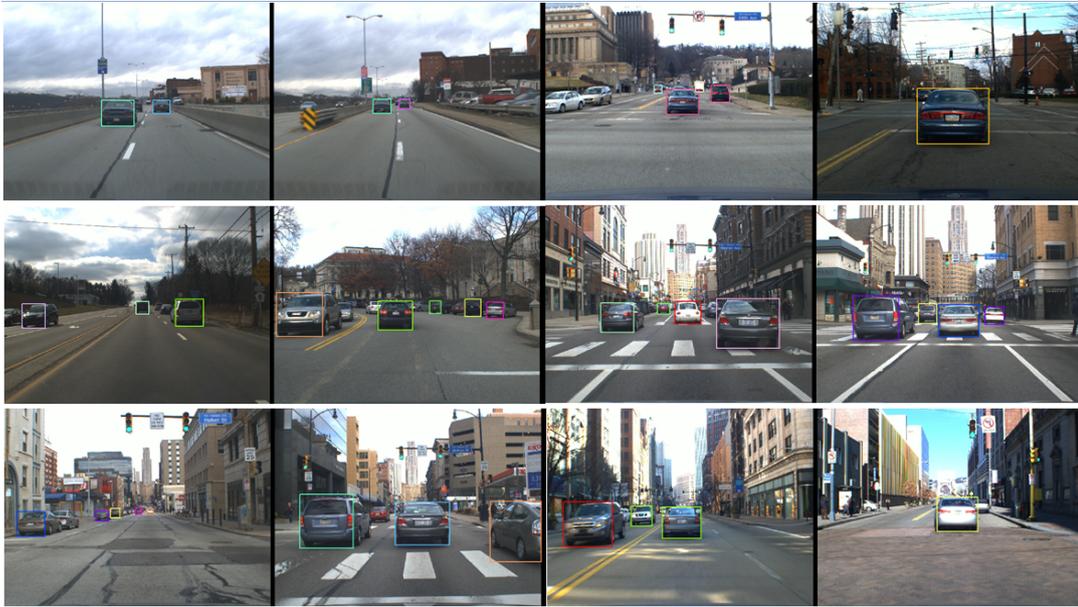


Figure 8: Some sample images with ground truth bounding boxes.

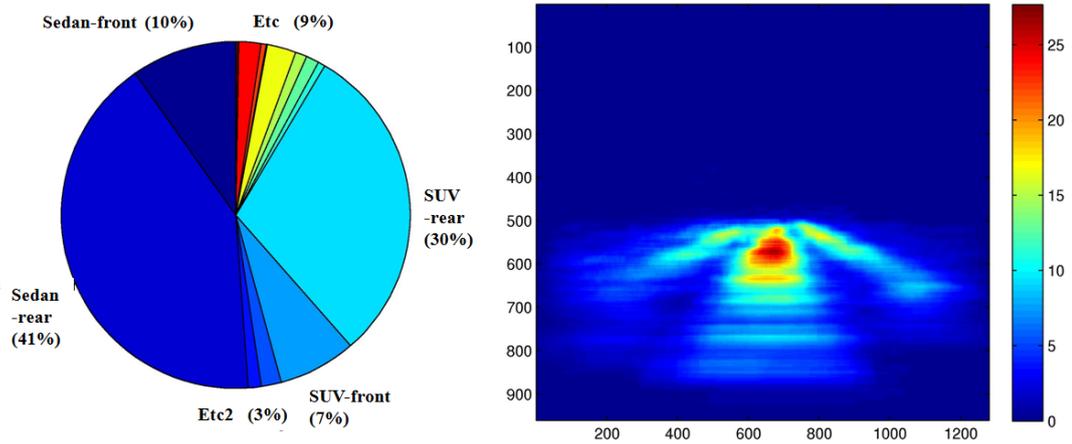


Figure 9: Distribution of vehicle sub-type samples (left) and distribution of bottom lines of all bounding boxes (right).

References

- [1] Bosch long-range radar. <http://www.bosch-automotivetechnology.com>. 5, 82
- [2] Continental ars 300 long range radar. <http://www.conti-online.com/>. 5
- [3] Delphi esr radar. <http://www.delphi.com/>. 5, 82
- [4] Flea3 fl3-ge-50s5c-c camera. <http://www.ptgrey.com/products/>. 82
- [5] Flir pathfindir camera. <http://www.flir.com>. 82
- [6] History of autonomous car. <http://en.wikipedia.org/wiki/Autonomous-car>. 14
- [7] Ibeo lux. <http://www.ibeo-as.com/>. 6, 82
- [8] Mobileye lane detection. <http://www.mobileye.com/technology/applications/lane-detection/>. 12, 109, 110, 113, 120, 123
- [9] Sick lms. <http://www.sick.com/>. 6
- [10] Velodyne: High-definition lidar. <http://velodynelidar.com/>. 6, 8, 79
- [11] R. Aufrere, J. Gowdy, C. Mertz, C. Thorpe, C.-C. Wang, and T. Yata. Perception for collision avoidance and autonomous driving. *Mechatronics*, 13(10):1149–1161, December 2003. 19
- [12] A. Bar-Hillel, R. Lerner, D. Levi, and G. Raz. Recent progress in road and lane detection: a survey. *Machine Vision and Applications*, 2012. 109

REFERENCES

- [13] A. Bar-Hillel, D. Levi, E. Krupka, and C. Goldberg. Part-based feature synthesis for human detection. In *Proc. European Conference on Computer Vision*, 2010. 17, 24, 43
- [14] Y. Bar-Shalom. *Tracking and data association*. Academic Press Professional, Inc., San Diego, CA, 1987. 18
- [15] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation*. Wiley Interscience, 2001. 19, 60, 63, 65, 90
- [16] BBC. Toyota sneak previews self-drive car ahead of tech show. <http://www.bbc.com/news/technology-20910769>. 3, 4, 123
- [17] R. Bishop. *Intelligent Vehicle Technologies and Trends*. 2005. 59, 109
- [18] A. Broggi. The vislab intercontinental autonomous challenge. <http://viac.vislab.it/>. 2, 3, 15, 79
- [19] A. Broggi, M. Bertozzi, A. Fascioli, C. G. L. Bianco, and A. Piazzzi. The argo autonomous vehicles's vision and control systems. *International Journal of Intelligent Control and Systems*, 3(4):409–441, 1999. 15
- [20] H. Cho, P. Rybski, A. B. Hillel, and W. Zhang. Real-time pedestrian detection with deformable part models. In *Proc. IEEE Intelligent Vehicles Symp.*, 2012. 23, 27, 36, 37, 43, 59, 64, 98, 123, 128
- [21] H. Cho, P. Rybski, and W. Zhang. Vision-based bicycle detection and tracking using a deformable part model and an ekf algorithm. In *Proc. IEEE Conf. Intelligent Transportation Systems*, 2010. 62, 123
- [22] H. Cho, P. Rybski, and W. Zhang. Vision-based 3d bicycle tracking using deformable part model and interacting multiple model filter. In *Proc. IEEE Int. Conf. Robotics and Automation*, 2011. 18, 60, 63, 123
- [23] H. Cho, Y.-W. Seo, B. V. Kumar, and R. R. Rajkumar. A multi-sensor fusion system for moving object detection and tracking in urban driving environments. In *Proc. IEEE Int. Conf. Robotics and Automation*, 2014. 123

-
- [24] H. Cho, W. Zhang, S. W. Bang, and B. V. Kumar. Real-time pedestrian and vehicle detection and tracking for automotive active safety systems (under review). 2014. [59](#)
- [25] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–577, 2003. [18](#)
- [26] Daimler. Mercedes-benz s500 intelligent drive. Youtube video. [3](#), [4](#)
- [27] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2005. [16](#), [17](#), [18](#), [22](#), [28](#), [43](#), [44](#), [47](#)
- [28] N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. In *Proc. European Conf. of Computer Vision*, 2006. [29](#)
- [29] M. S. Darms, P. Rybski, C. Baker, and C. Urmson. Obstacle detection and tracking for the urban challenge. *IEEE Transaction on Intelligent Transportation Systems*, 10(3), 2009. [xiv](#), [88](#), [89](#), [90](#), [93](#), [96](#), [123](#)
- [30] P. Dollár, R. Appel, and W. Kienzle. Crosstalk cascades for frame-rate pedestrian detection. In *Proc. European Conference on Computer Vision*, 2012. [16](#), [44](#), [47](#)
- [31] P. Dollár, S. Belongie, and P. Perona. The fastest pedestrian detector in the west. In *Proc. British Machine Vision Conf.*, 2010. [9](#), [16](#), [24](#), [32](#), [40](#), [44](#)
- [32] P. Dollár, Z. Tu, P. Perona, and S. Belongie. Integral channel features. In *Proc. British Machine Vision Conf.*, 2009. [16](#), [18](#), [22](#)
- [33] P. Dollár, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99(Preliminary), 2011. [xi](#), [16](#), [23](#), [24](#), [36](#), [37](#), [38](#), [40](#), [41](#), [44](#), [72](#), [128](#)

REFERENCES

- [34] A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice*. 2001. [18](#), [67](#)
- [35] H. Durrant-Whyte. *Multi Sensor Data Fusion*. Lecture Notes, 2006. [89](#)
- [36] A. Efros, A. Berg, G. Mori, and J. Malik. Recognizing action at a distance. In *Proc. International Conference of Computer Vision*, 2003. [29](#)
- [37] M. Enzweiler and D. Gavrilu. Monocular pedestrian detection: Survey and experiments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31:2179–2195, 2008. [16](#)
- [38] A. Ess, B. Leibe, K. Schindler, , and L. van Gool. Robust multi-person tracking from a mobile platform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(10), 2009. [18](#), [19](#), [106](#)
- [39] A. Fathi and G. Mori. Action recognition by learning mid-level motion features. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2008. [29](#)
- [40] P. Felzenszwalb, R. Girshick, and D. McAllester. Cascade object detection with deformable part models. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2010. [10](#), [21](#), [22](#), [24](#), [26](#), [32](#), [60](#), [77](#), [124](#)
- [41] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99, 2010. [9](#), [17](#), [18](#), [21](#), [22](#), [24](#), [26](#), [28](#), [32](#), [43](#), [60](#), [77](#), [98](#), [124](#)
- [42] P. Felzenszwalb and D. Huttenlocher. Distance transforms of sampled functions. *Cornell Computing and Information Science Technical Report TR2004-1963*, 2004. [26](#)
- [43] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester. Discriminatively trained deformable part models, release 3. <http://www.cs.brown.edu/~pff/latent-release3/>. [26](#), [27](#), [39](#), [47](#), [50](#)

-
- [44] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester. Discriminatively trained deformable part models, release 4. <http://people.cs.uchicago.edu/~pff/latent-release4/>. 26
- [45] T. Gandhi and M. M. Trivedi. Pedestrian protection systems: Issues, survey, and challenges. 8(3):413–430, 2007. 16, 59
- [46] D. M. Gavrila. Sensor-based pedestrian protection. *IEEE Intelligent System*, 16(6):77–81, 2001. 15
- [47] D. M. Gavrila and S. Munder. Multi-cue pedestrian detection and tracking from a moving vehicle. *International Journal of Computer Vision*, 73:41–59, 2007. 17, 18, 19
- [48] A. Geiger. *Probabilistic Models for 3D Urban Scene Understanding from Movable Platforms*. PhD thesis, Karlsruhe Institute of Technology, 2013. 67
- [49] D. Geroñimo, A. López, and T. G. A. Sappa. Survey of pedestrian detection for advanced driver assistance systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:1239–1258, 2010. 16, 56
- [50] F. Han, Y. Shan, R. Cekander, H. S. Sawhney, and R. Kumar. A two-stage approach to people and vehicle detection with hog-based svm. In *PMIS*, 2006. 17
- [51] G. Heitz, S. Gould, A. Saxena, and D. Koller. Cascaded classification models: Combining models for holistic scene understanding. NIPS, 2008. 106
- [52] D. Hoiem, A. Efros, and M. Hebert. Closing the loop on scene interpretation. CVPR, 2008. 106
- [53] W. Hunter, W. Pein, and J. Stutts. Bicycle crash types: A 1990’s informational guide. (FHWA-RD-96-104), 1997. 72, 128

-
- [54] N. Kaempchen, K. Weiss, M. Schaefer, and K. Dietmayer. Imm object tracking for high dynamic driving maneuvers. In *Proc. IEEE Intelligent Vehicles Symp.*, pages 825–830, Parma, Italy, 2004. [60](#), [63](#), [64](#), [91](#)
- [55] S. Kammel, J. Ziegler, B. Pitzer, M. Werling, T. Gindele, D. Jagszent, J. Schröder, M. Thuy, M. Goebel, F. von Hundelshausen, O. Pink, C. Frese, and C. Stiller. Team annieway’s autonomous system for the darpa urban challenge 2007. *Journal of Field Robotics, Special Issue on the 2007 DARPA Urban Challenge, Part I*, pages 359–391, 2009. [3](#)
- [56] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009. [108](#)
- [57] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83-97, 1955. [19](#)
- [58] K. H. Kwak, D. Huber, H. Badino, and T. Kanade. Extrinsic calibration of a single line scanning lidar and a camera. In *International Conference on Intelligent Robots and Systems (IROS 2011)*, December 2011. [84](#)
- [59] B. Leibe, N. Cornelis, K. Cornelis, and L. V. Gool. Dynamic 3d scene analysis from a moving vehicle. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2007. [17](#), [18](#), [19](#)
- [60] J. Leonard, J. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. S. Huang, S. Karaman, and O. Koch. A perception-driven autonomous urban vehicle. *Journal of Field Robotics, Special Issue on the 2007 DARPA Urban Challenge, Part I*, 2008. [2](#), [3](#), [15](#), [20](#), [79](#), [123](#)
- [61] Z. Li, L. L. K. Wang, and F. Wang. A review on vision-based pedestrian detection for intelligent vehicles. In *Conference on Vehicular Electronics and Safety*, 2006. [15](#)
- [62] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence*, 1981. [18](#), [30](#)

-
- [63] J. Maddox. Improving driving safety through automation. 2012. [2](#), [59](#)
- [64] M. Mählich, R. Schweiger, W. Ritter, and K. Dietmayer. Sensorfusion using spatio-temporal aligned video and lidar for improved vehicle detection. In *IV*, 2006. [20](#)
- [65] J. C. McCall and M. M. Trivedi. Video-based lane estimation and tracking for driver assistance: Survey, systems, and evaluation. *IEEE Transactions on Intelligent Transportation Systems*, 7:20–37, 2006. [109](#)
- [66] M. McNaughton, C. R. Baker, T. Galatali, B. Salesky, C. Urmson, and J. Zigar. Software infrastructure for an autonomous ground vehicle. *Journal of Aerospace Computing, Information, and Communication*, 5(12):491–505, December 2008. [xiv](#), [99](#)
- [67] C. Mertz. Moving object detection with laser scanners. *Journal of Field Robotics*, 30(1):17–43, 2013. [19](#), [86](#)
- [68] K. Mikolajczyk, C. Schmid, and A. Zisserman. Human detection based on a probabilistic assembly of robust part detectors. In *ECCV*, 2004. [17](#)
- [69] I. Miller, M. E. Campbell, D. Huttenlocher, A. Nathan, F.-R. Kline, P. Moran, N. Zych, B. Schimpf, S. Lupashin, E. Garcia, J. Catlin, M. Kurdzial, and H. Fujishima. Team cornell’s skynet: Robust perception and planning in an urban environment. *Journal of Field Robotics, Special Issue on the 2007 DARPA Urban Challenge, Part I*, 25(8):493–527, 2008. [2](#), [3](#), [15](#), [20](#), [79](#)
- [70] G. Monteiro, C. Premebida, P. Peixoto, and U. Nunes. Tracking and classification of dynamic obstacles using laser range finder and vision. In *IROS Workshop on safe navigation in open environments*, 2006. [20](#)
- [71] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, A. Petrovskaya, M. Pflueger, G. Stanek, D. Stavens, A. Vogt, and S. Thrun. Junior: The stanford entry in the urban challenge. *Journal*

REFERENCES

- of Field Robotics, Special Issue on the 2007 DARPA Urban Challenge, Part I*, 25(9):569–597, 2008. [2](#), [3](#), [15](#), [20](#), [79](#), [123](#)
- [72] F. Moutarde, B. Stanciulescu, and A. Breheret. Real-time visual detection of vehicles and pedestrians with new efficient adaboost features. In *PPNIV*, 2008. [17](#)
- [73] NewYorkTimes. Google cars drive themselves in traffic. <http://www.nytimes.com/2010/10/10/science/10google.html?r=2>. [2](#), [3](#)
- [74] NHTSA. National motor vehicle crash causation survey. (DOT HS 811 059), 2008. [59](#), [61](#), [71](#)
- [75] NHTSA. National motor vehicle crash causation survey. (DOT HS 811 059), 2008. [61](#), [71](#), [73](#)
- [76] NHTSA. 2010 motor vehicle crashes: Overview. (DOT HS 811 552), 2012. [37](#), [132](#)
- [77] K. Okuma, A. Taleghani, N. de Freitas, J. Little, and D. Lowe. A boosted particle filter: Multitarget detection and tracking. In *Proc. European Conference on Computer Vision*, 2004. [19](#)
- [78] C. Papageorgiou and T. Poggio. A trainable system for object detection. *International Journal of Computer Vision*, 38(1):15–33, 2000. [16](#), [17](#), [18](#), [22](#)
- [79] D. Park, C. L. Zitnick, D. Ramanan, and P. Dollár. Exploring weak stabilization for motion feature extraction. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2013. [29](#), [30](#), [31](#), [47](#), [48](#), [56](#)
- [80] M. Piccardi. Background subtraction techniques: a review. *IEEE Transactions on Systems, Man, and Cybernetics*, 4:3099–3104, 2004. [29](#)
- [81] D. Pomerleau. Ralph: Rapidly adapting lateral position handler. In *IEEE Symposium on Intelligent Vehicles*, pages 506 – 511, September 1995. [15](#)

-
- [82] D. Ponsa, J. Serrat, and A. M. López. On-board image-based vehicle detection and tracking. *Transactions of the Institute of Measurement and Control*, 33:738–805, 2011. [106](#)
- [83] C. Premebida, O. Ludwig, and U. Nunes. Lidar and vision-based pedestrian detection system. *Journal of Field Robotics*, 26(9):696–711, 2009. [19](#), [20](#)
- [84] D. B. Reid. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 24(6), 1979. [18](#)
- [85] C. F. Reinholtz, D. Hong, A. Wicks, A. Bacha, C. Bauman, R. Faruque, M. Fleming, C. Terwelp, T. Alberi, D. Anderson, S. Cacciola, P. Currier, A. Dalton, J. Farmer, J. Hurdus, S. Kimmel, P. King, A. Taylor, D. V. Covern, and M. Webster. Odin: Team victortangos entry in the darpa urban challenge. *Journal of Field Robotics, Special Issue on the 2007 DARPA Urban Challenge, Part I*, 25(8):467–492, 2008. [2](#), [3](#), [15](#), [79](#)
- [86] B. Ristic, S. Arulampalam, and N. Gordon. *Beyond the Kalman Filter*. 2004. [67](#), [70](#)
- [87] Y.-W. Seo and R. Rajkumar. Use of a monocular camera to analyze a ground vehicle’s lateral movements for reliable autonomous city driving. In *PPNIV*, 2013. [33](#)
- [88] A. Shashua, Y. Gdalyahu, and G. Hayun. Pedestrian detection for driving assistance systems: Single-frame classification and system level performance. In *Proc. IEEE Intelligent Vehicles Symp.*, pages 13–18, 2004. [17](#), [49](#), [56](#)
- [89] J. Shi and C. Tomasi. Good features to track. pages 593–600, 1994. [29](#)
- [90] S. Sivaraman and M. M. Trivedi. Integrated lane and vehicle detection, localization, and tracking: A synergistic approach. *IEEE Transactions on Intelligent Transportation Systems*, 14:906–917, 2013. [106](#), [107](#)
- [91] E.-J. Sol. The grand cooperative driving challenge. <http://www.gcdc.net/>. [2](#)

-
- [92] J. Sousanis. World vehicle population tops 1 billion units. 2011. [1](#)
- [93] L. Spinello and R. Siegwart. Human detection using multimodal and multidimensional features. In *ICRA*, 2008. [20](#)
- [94] P. Stenquist. Nissan announces plans to release driverless cars by 2020. *The New York Times*. [3](#), [4](#)
- [95] C. Stiller, J. Hipp, and A. E. C. Róssig. Lidar and vision-based pedestrian detection system. *Journal of Image and Vision Computing*, pages 389–396, 2000. [19](#)
- [96] P. Sudowe and B. Leibe. Efficient use of geometric constraints for sliding-window object detection in video. In *ICVS*, 2011. [33](#)
- [97] Z. Sun, G. Bebis, and R. Miller. On-road vehicle detection: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(5), 2006. [17](#)
- [98] S. Srkk, A. Vehtari, and J. Lampinen. Rao-blackwellized particle filter for multiple target tracking. *Information Fusion*, 8(1):2–15, 2007. [10](#), [60](#), [67](#), [69](#)
- [99] A. Takeuchi, S. Mita, and D. McAllester. On-road vehicle tracking using deformable object model and particle filter with integrated likelihoods. In *Proc. IEEE Intelligent Vehicles Symp.*, 2010. [17](#), [18](#), [19](#)
- [100] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney. Winning the darpa grand challenge. *Journal of Field Robotics*, 2006. [15](#)
- [101] C. Urmson. The self-driving car logs more miles on new wheels. <http://googleblog.blogspot.com/2012/08/the-self-driving-car-logs-more-miles-on.html>. [2](#), [3](#), [15](#), [79](#), [123](#)

-
- [102] C. Urmson, J. Anhalt, H. Bae, J. A. D. Bagnell, C. R. Baker, R. E. Bittner, T. Brown, M. N. Clark, M. Darms, D. Demitrish, J. M. Dolan, D. Duggins, D. Ferguson, T. Galatali, C. M. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. Howard, S. Kolski, M. Likhachev, B. Litkouhi, A. Kelly, M. McNaughton, N. Miller, J. Nickolaou, K. Peterson, B. Pilnick, R. Rajkumar, P. Rybski, V. Sadekar, B. Salesky, Y.-W. Seo, S. Singh, J. M. Snider, J. C. Struble, A. T. Stentz, M. Taylor, W. R. L. Whittaker, Z. Wolkowicki, W. Zhang, and J. Zigar. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics, Special Issue on the 2007 DARPA Urban Challenge, Part I*, 25(8):425–466, 2008. [2](#), [3](#), [4](#), [15](#), [79](#), [123](#)
- [103] C. Urmson, J. Anhalt, D. Bartz, M. Clark, T. Galatali, A. Gutierrez, S. Harbaugh, J. Johnston, H. Kato, P. L. Koon, W. Messner, N. Miller, A. Mosher, K. Peterson, C. Ragusa, D. Ray, B. K. Smith, J. M. Snider, S. Spiker, J. C. Struble, J. Zigar, and W. R. L. Whittaker. A robust approach to high-speed navigation for unrehearsed desert terrain. *Journal of Field Robotics*, 23(8):467–508, August 2006. [15](#)
- [104] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2001. [16](#), [32](#)
- [105] P. Viola, M. J. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. *International Journal of Computer Vision*, 63(2):153–161, 2005. [16](#), [17](#), [18](#), [22](#), [29](#)
- [106] S. Walk, N. Majer, K. Schindler, and B. Schiele. New features and insights for pedestrian detection. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2010. [16](#), [29](#), [44](#)
- [107] R. Wallace and G. Silberg. Self-driving cars: The next revolution. <http://www.cargroup.org>. [2](#), [59](#)

- [108] H. Wang, M. Ullah, A. Klaser, I. Laptev, and C. Schmid. Evaluation of local spatio-temporal features for action recognition. In *Proc. British Machine Vision Conf.*, 2009. [16](#), [17](#), [24](#), [29](#), [43](#)
- [109] J. Wei, J. M. Snider, J. Kim, J. M. Dolan, R. Rajkumar, and B. Litkouhi. Towards a viable autonomous driving research platform. In *Proc. IEEE Intelligent Vehicles Symp.*, 2013. [xii](#), [xiv](#), [3](#), [4](#), [15](#), [49](#), [50](#), [79](#), [80](#), [82](#), [97](#), [116](#), [123](#)
- [110] H. Weigel, P. Lindner, and G. Wanielik. Vehicle tracking with lane assignment by camera and lidar sensor fusion. In *Proc. IEEE Intelligent Vehicles Symp.*, 2009. [107](#)
- [111] C. Wojek and B. Schiele. A performance evaluation of single and multi-feature people detection. In *DAGM Symposium Pattern Recognition*, 2008. [16](#), [43](#)
- [112] C. Wojek, S. Walk, and B. Schiele. Multi-cue onboard pedestrian detection. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2009. [22](#), [29](#)
- [113] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Comput. Surv.*, 38(4), Dec. 2006. [29](#)
- [114] Z. Zhang and Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page 2000. [83](#)