

PERCEPTION FOR AUTONOMOUS DRIVING IN URBAN ROAD ENVIRONMENT

QIN BAOXING
B. Eng (SJTU)

A THESIS SUBMITTED
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
DEPARTMENT OF MECHANICAL ENGINEERING
NATIONAL UNIVERSITY OF SINGAPORE
2014

Declaration

I hereby declare that the thesis is my original work and it has been written by me in its entirety.

I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

QBX 秦宝星.

QIN BAOXING

31/July/2014

Acknowledgment

I would like to express my sincere gratitude to my advisor, Prof. Marcelo H. Ang Jr., who brings me to this fantastic project of autonomous vehicle, who gives me invaluable suggestions during my research all the way, who reviews my papers carefully and patiently, who generously supports me to attend many good conferences and exposes my eyes to the most brilliant ideas in the world. He is not only a respectable advisor, but an adorable friend.

I would like to express my thanks to Prof. Emilio Frazzoli and Prof. Daniela Rus from MIT, who advise our project, keep us motivated all the time, give priceless guidances, and bring many excellent collaborators into this project.

I would like to thank all of our current and previous team members: Chong Zhuang Jie Demian, Liu Wei, Shen Xiaotong, Scott Pendleton, Dr. James Fu, Dr. Harold Soh, Tawit Uthaicharoenpong, Dr. Tirthankar Bandyopadhyay, Dr. Brice Rebsamen, Dr. Tichakorn Wongpiromsarn Nok, Dr. Kim Seong-Woo Sean and Dai Peilong. They give me a lot of help and great suggestions in my research. Many thanks to these good friends.

Last but definitely not the least, I have to thank my family. I owe the deepest gratitude to my parents, who gave birth to me, brought me up, and who are always supporting me and loving me. I owe my gratitude to my wife, Lize, who accompanies me, shares my pain and brings me happiness everyday.

Table of Contents

Declaration	i
Acknowledgments	ii
Table of Contents	iii
Summary	ix
List of Tables	xii
List of Figures	xiii
1 Introduction	1
1.1 Background	1
1.2 Perception for Autonomous Driving	3
1.2.1 Characteristics of Urban Road Environment	4
1.2.2 Minimal Sensing	5
1.3 Scope of the Thesis	6
1.3.1 Localization	7
1.3.2 Road Detection	8
1.3.3 Moving Object Recognition	9
1.3.4 Environment Understanding	9
1.4 Thesis Outline	11
2 Literature Review	12

2.1	Autonomous Vehicles and Their Perception Modules	12
2.1.1	A Brief History	12
2.1.2	Current Trends	14
2.1.3	Discussion	17
2.2	Localization	17
2.2.1	GPS/INS Fusion Approach	18
2.2.2	Road-Matching Approach	19
2.2.3	Map-aided Approach	20
2.2.4	SLAM Approach	22
2.2.5	Discussion	24
2.3	Object Recognition	26
2.3.1	A General Review	26
2.3.2	Road Detection	27
2.3.3	Moving Object Recognition	28
2.3.4	Discussion	30
2.4	Environment Understanding	30
2.4.1	Semantic Mapping	31
2.4.2	Activity Learning	32
2.4.3	Discussion	33
2.5	Summary	34
3	Curb-Intersection Feature based Monte Carlo Localization	35
3.1	Introduction	35
3.2	System Overview	36
3.3	Curb-Intersection Feature Extraction	36
3.3.1	Segmentation of Laser Scan	37
3.3.2	Classification of Scan Segments	39
3.3.3	Intersection Feature	40
3.4	Synthetic LIDAR Construction	40
3.4.1	Synthetic LIDARs in the Horizontal Plane	40

3.4.2	Scan-Assembled Synthetic LIDARs	41
3.5	Monte Carlo Localization Algorithm	42
3.5.1	MCL Overview	42
3.5.2	Pseudo-3D Odometry Motion Model	43
3.5.3	Curb-Intersection Measurement Model	44
3.5.4	Practical Considerations	45
3.6	Experiments	46
3.6.1	Experimental Setup	46
3.6.2	Experimental Results	46
3.6.3	Autonomous System Demonstration	51
3.7	Summary	51
4	Synthetic 2D LIDAR for Precise Localization in 3D Urban Environment	54
4.1	Localization on a Virtual Plane	55
4.1.1	Projecting the 3D world to the 2D plane	55
4.1.2	System Overview	56
4.2	3D perception	58
4.2.1	3D rolling window	58
4.2.2	Point Classification	59
4.2.3	Synthetic LIDAR construction	60
4.3	Online Localization	62
4.3.1	MCL Localization	62
4.3.2	Synthetic LIDAR Measurement Model	62
4.4	Experiments	63
4.4.1	Experiment Setup	63
4.4.2	Experiment Results	64
4.5	Summary	67
5	A General Framework for Road Marking Detection and Analysis	69
5.1	Introduction	69

5.2	Framework Overview	70
5.3	Image Processing	71
5.3.1	Inverse Perspective Mapping (IPM)	71
5.3.2	Image Binarization	72
5.3.3	Contour Extraction	74
5.4	Contour Classification and Analysis	74
5.4.1	Lane Module	75
5.4.2	Arrow Module	76
5.4.3	Zebra-Crossing Module	77
5.4.4	Word Module	79
5.5	Experiments	80
5.6	Summary	83
6	Road Detection and Mapping using 3D Rolling Window	86
6.1	Introduction	86
6.2	3D Rolling Window	87
6.2.1	Construction and Maintenance	88
6.2.2	Probabilistic Characteristics	88
6.2.3	Extended Point Cloud	92
6.3	Cascaded Process of Road Detection	92
6.3.1	Noise Prefiltering	93
6.3.2	Region-growing Method	94
6.3.3	Road Boundary Adjustment	95
6.3.4	Post Classification	96
6.3.5	Road Detection for Point Cloud Segmentation	97
6.4	Probabilistic Road Mapping	97
6.5	Experiments	98
6.5.1	Experiment Setup	98
6.5.2	Experiment Results	99
6.6	Summary	101

7	A Spatial-Temporal Approach for Moving Object Recognition with 2D LI-DAR	104
7.1	Introduction	104
7.2	Technical Approach	106
7.2.1	Data Accumulation in T-domain	106
7.2.2	Graph-based ST Segmentation	107
7.2.3	Spatial-Temporal (ST) features	108
7.3	Experiments	110
7.4	Results	111
7.5	Summary	118
8	From Human Activity Learning to Semantic Mapping	119
8.1	Introduction	120
8.2	System Overview	121
8.2.1	Multi-dimensional Grid Map	121
8.2.2	System Framework	121
8.3	Pedestrian Activity Learning	122
8.3.1	Pedestrian Detection and Tracking	123
8.3.2	Track Classification and Clustering	123
8.3.3	Activity Learning with Gaussian Process	125
8.4	Activity-based Semantic Reasoning	128
8.4.1	Pedestrian Path Learning	129
8.4.2	Refined Semantics Learning	130
8.5	Experiments	134
8.5.1	Experiment Setup	134
8.5.2	Experiment Results	134
8.6	Summary	139
9	Conclusions and Future Work	140
9.1	Conclusions	140

9.1.1	A Brief Review	140
9.1.2	Insights of Minimal Sensing	142
9.1.3	Contributions	143
9.2	System Integration and Practical Issues	144
9.3	Future Work	146
	Bibliography	148
	Appendices	165
	A Author's Publications	166
	B Video Links	168

Summary

Vehicle autonomous driving has become one of the most popular research areas in robotics. Autonomous vehicles will not only enhance operational safety and efficiency of the transportation system, but also provide convenience to the vehicle users and improve their productivity. This thesis focuses on developing the perception functions for vehicle autonomous driving in the urban road environment. Fundamental perception requirements are identified through literature review, and the contributions of this thesis are the minimal-sensing solutions for these perception requirements. We demonstrate that with the minimal sensing ability, our algorithms are able to achieve equivalent or better performance compared to existing techniques.

We first review the history and current status of autonomous vehicle technology, and summarize the important perception requirements for autonomous navigation in the urban road environment. Three fundamental perception tasks are identified from the review, including localization, object recognition, and environment understanding. Our researches on these three perception tasks are the main body of this thesis.

To address the problem of vehicle localization, we manage to utilize the typical features in the urban road environment for pose estimation, with only a tilted-down 2D LIDAR and the odometry system. In the first stage of our research, curb-intersection features are extracted to localize the vehicle. While curb features help estimate the position in the lateral direction, intersection features are beneficial to reduce the longitudinal uncertainty. Our algorithm makes use of both curb and intersection features for localization, and shows accurate results. However, the curb-intersection-based algorithm only applies to roads where curbs exist, and may not be general enough for all the urban road

scenarios. For this reason, in the second stage of the research, we consider incorporating other urban features for localization. Since the urban environment is composed of artificial objects or structures which usually have vertical surfaces, we try to utilize these vertical surfaces as the localization features. Compared to the curb-intersection algorithm, the “vertical surface” algorithm is applicable to general urban road scenarios (with/without curbs), and has better localization.

Problems of object recognition are also studied in this thesis. While object recognition is a broad research topic, our attention is focused on two specific tasks that are more relevant to vehicle autonomous driving, i.e., road detection and moving object recognition. For the task of road detection, we investigate two categories of research, i.e., road marking detection using vision, and road surface-boundary detection using LIDAR. For vision-based marking detection, we propose a general framework for the detections and analyses of various types of markings. For LIDAR-based surface-boundary detection, we introduce the idea of a “3D rolling window” and solve the problem in a 3D manner. As for the task of moving object recognition, we propose a spatial-temporal approach to solve it, with only a 2D planar LIDAR. Avoiding using more elaborate and costly sensors like the 3D Velodyne, we show that it is possible to obtain highly accurate object classification via temporal accumulation. Our algorithm is tested in both campus and highway scenarios, and shows good accuracy.

Besides the object recognition functions developed for the short-term object-oriented detection purpose, to endow the robot with higher-level intelligence, we are also interested in acquiring some long-term environment-oriented understanding. While the understanding of an environment can be about any dimension of its properties, in our research, we concentrate on the semantic and activity dimensions. Unlike existing researches approaching different dimensions of understanding independently, we argue that these dimensions are highly correlated and can be learned from each other. We implement this idea to infer the semantic property from learned activity knowledge, and

achieve promising results.

Keywords: Autonomous Vehicle; Urban Road Environment; Perception; Localization; Road Detection; Moving Object Recognition; Environment Understanding; Semantic Mapping; Activity Learning.

List of Tables

1.1	Sensor configuration for golfcart testbed	6
2.1	A summary of perception systems for the six finishers in DARPA Urban Challenge	15
2.2	A summary of perception systems for recent notable projects	16
3.1	Pseudo-3D Odometry Sample Motion Model (u_t, x_{t-1})	44
3.2	Localization error at several marked points	49
5.1	Geometric features of marking contours	75
5.2	Classification performance for road markings	81
5.3	Confusion matrix of arrow classification	81
5.4	Confusion matrix of word classification	82
5.5	Computation time of different processes	83
6.1	classification accuracy for boundary candidates	101
7.1	Feature vector of the compressed segment	109
7.2	Classification using different feature sets	117
8.1	Mapping results for semantic properties of the four types	138
9.1	Sensor usage of different perception functions	144

List of Figures

1.1	SMART autonomous golfcart	3
1.2	SMART autonomous iMiev	3
1.3	Software structure of SMART autonomous vehicle	4
1.4	Thesis scope and outline	6
3.1	Curb-intersection feature based localization flowchart	37
3.2	Model of LIDAR sensing on-road	38
3.3	Raw LIDAR reading and filter response	39
3.4	Scan segment classification	40
3.5	Curb-intersection feature at a T-junction	41
3.6	Synthetic LIDAR construction	42
3.7	Assembled synthetic LIDARs	43
3.8	Pseudo-3D localization	44
3.9	Yamaha G22E golf cart mounted with various sensors	47
3.10	Localization results	47
3.11	Road boundary map and detected curb features	48
3.12	Position estimation standard deviation	49
3.13	Typical particle behaviours	50
3.14	Autonomous system demonstration	52
4.1	Synthetic LIDAR based localization flowchart	57
4.2	3D rolling window	59
4.3	Construction of synthetic LIDAR	61

4.4	Vehicle testbed	63
4.5	Mapping of the NUS Engineering Campus	64
4.6	Localization results during a manual drive	65
4.7	Localization standard deviations	66
4.8	Autonomous navigation with synthetic LIDAR	68
5.1	A general framework for road marking detection and analysis	71
5.2	Inverse Perspective Mapping	72
5.3	An example of image processing	73
5.4	Four marking modules	75
5.5	Lane processing	76
5.6	Common arrow types	77
5.7	Arrow processing	77
5.8	Zebra-crossing processing	79
5.9	Word recognition	80
5.10	Shadow-highlight situation	83
5.11	Marking detection and analysis results	84
6.1	Coordinate system for 3D data accumulation	89
6.2	Road detection flowchart	93
6.3	Curvature hint for road detection	94
6.4	Boundary point adjustment	96
6.5	Point cloud segmentation example	97
6.6	Three typical cases of rolling window	100
6.7	Noisy scan rolling window filtering	100
6.8	Increased classification accuracy through SVM	102
6.9	Results of the SVM classification of road boundary and surfaces	102
6.10	Road mapping results	103
7.1	One example to illustrate spatial-temporal method	105
7.2	Flowchart of the spatial-temporal algorithm	107

7.3	Compressed segment.	109
7.4	Pose-variant and Pose-invariant feature sets	110
7.5	Testbed for moving object recognition	111
7.6	Non-vehicle to vehicle ratios in different environments.	112
7.7	Classification at different environments	113
7.8	Moving vehicle detection examples	114
7.9	Overall vehicle detection performance	116
8.1	Correlated multiple dimensions of information	122
8.2	Flowchart of semantic mapping from activity learning	122
8.3	Experiment environment and road network information	135
8.4	Track clustering results	136
8.5	Moving direction of activity model	137
9.1	Software system of our autonomous vehicle	145

Chapter 1

Introduction

Vehicle autonomous driving has become one of the most popular research areas in robotics. Autonomous vehicles will not only enhance operational safety and efficiency of the transportation system, but also provide convenience to the vehicle users and improve their productivity. This thesis focuses on developing the perception ability for vehicle autonomous driving in the urban road environment.

1.1 Background

DARPA Challenges. To stimulate research on autonomous vehicles, the Defense Advanced Research Projects Agency (DARPA) has organized a series of competitions for autonomous vehicles. Two of the largest and most recent field demonstrations are the DARPA Grand Challenge (DGC) [1] and the DARPA Urban Challenge (DUC) [2]. DGC was held twice in 2004 and 2005. In the first competition none of the vehicles finished the route. In the second one, five teams were able to complete a 212 km off-road course in the desert.

In DUC, held in 2007, autonomous robots were required to complete autonomous navigation in the urban environment, which was a more difficult task compared to the previous one. In this competition, the autonomous vehicles had to navigate, in a fully autonomous manner, through a partially known urban-like environment populated with (static and dynamic) obstacles and perform different tasks such as road and off-road driving, parking and visiting certain areas while obeying traffic rules. As the emphasis of the competition was geared more towards military applications, the vehicles had to

be fully self-contained in every aspect including perception, motion planning, behavior reasoning, etc. Although this leads to an elegant setup, the situations encountered in DUC do not closely represent those faced in real-world crowded environments such as in cities like Singapore, London, etc. In addition, the cost of the hardware components on these autonomous vehicles is extremely high, making them impractical to be employed in social or commercial applications. The majority of the cost comes from the expensive, high-performance sensors (e.g. Velodyne LIDAR) and localization units (e.g. Applanix Inertial Navigation System), which are needed so that the vehicles can effectively handle all the possible (even adversarial) environments they may encounter.

Google Car. The company Google has taken the lead in autonomous vehicle research after the success of the two challenges [3]. The Google driverless cars have demonstrated their ability of autonomous navigation in the crowded urban environment. However, similar to the DARPA stories, their full autonomy comes with the high cost of expensive sensors, like Velodyne LIDAR, high-precision GPS/INS, and multiple cameras and radars.

Future Urban Mobility Project. In 2010, the Singapore-MIT Alliance for Research and Technology (SMART) initiated an interdisciplinary research program project on Future Urban Mobility (FM). Autonomy in Mobility-on-Demand Systems is one research project in FM. As part of the research, we are aiming to realize vehicle autonomy under minimal sensing [4–12]. Instead of relying on powerful but expensive sensors like 3D LIDAR, we use cheap sensors of limited sensing ability, such as 2D laser range finders and cameras, to realize localization, obstacle detection, and other various perception functions. The requirement of minimal sensing pushes us to develop more intelligent algorithms under available perception resources. On the other hand, we want to utilize infrastructure sensors for additional information. Nowadays modern cities are evolving into sense-able cities with the increasing deployment of various sensors. For example, cameras are mounted to monitor traffic, loop detectors to count vehicle numbers, and so on. In the electronic road pricing (ERP) project in Singapore, even 24/7 tracking of vehicles is in trials. All the information from the above infrastructure sensors can be very useful for a vehicle to perceive its environment.

Till now, we have two autonomous prototypes converted, one is a Yamaha G22E golf cart shown by Figure 1.1, and another is an iMiev shown by Figure 1.2. These

two vehicles have been thoroughly tested inside and outside the NUS campus, and have demonstrated good performances [13] [14]. More details of our project can be found from Video (1)(2)(3) in Appendix B.



Figure 1.1: SMART autonomous golfcart



Figure 1.2: SMART autonomous iMiev

1.2 Perception for Autonomous Driving

The software system of our autonomous vehicle is composed of three modules: the perception module for environment sensing, the planning module for decision making

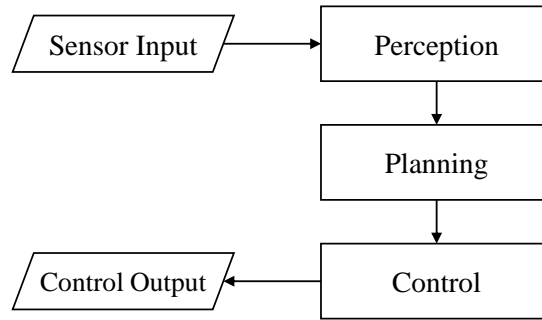


Figure 1.3: Software structure of SMART autonomous vehicle

and path planning, and the control module for the low-level speed and steering control. Figure 1.3 illustrates the software structure of the vehicle. This thesis is focused on developing the perception module of an autonomous vehicle, which parses measurement readings from sensors, and generates perception results about its **own states** and the **surrounding environment**. The perception results will be used as direct inputs to a vehicle’s planning module, and are hence of vital importance.

1.2.1 Characteristics of Urban Road Environment

Vehicle autonomous driving in different environments requires different perception functions, as suggested in the two DARPA Challenges. In DGC, the basic perception functions are only localization and traversable road detection. However in DUC, autonomous vehicles have to be able to not only localize themselves and detect roads, but also detect other agents and negotiate the traffic. Indeed, the urban road environment is a typical environment with distinctive characteristics. On one hand, the urban road environment is more challenging than the rural or desert environment, considering that it is full of other dynamic agents like pedestrians and cars, so that an autonomous vehicle has to be able to detect and deal with these agents well. On the other hand, however, since the urban road environment is a semi-structured environment, it provides multiple conveniences for autonomous navigation: since urban roads are usually well paved, road surface and boundary can be easily detected; markings on the road surface can be relied on to provide useful navigation guidance; vertical surfaces of urban buildings can be utilized for localization; etc. In the development of the perception functions, these characteristics of the urban road environment are fully considered and utilized, making our perception

functions accurate and robust.

1.2.2 Minimal Sensing

The contributions of this thesis are the minimal-sensing solutions for the fundamental perception functions of autonomous driving. While some of the perception problems may have been widely studied, we manage to solve them with minimal sensing ability. For example, expensive GPS/INS systems or 3D LIDARs (like a Velodyne) have been utilized for decimeter-level localization [15], our method manages to achieve the same accuracy with only the odometry and a 2D LIDAR [8]. The idea of “perception under minimal sensing” helps bring down the cost of our autonomous vehicle, as well as distinguishes our researches from other projects. We demonstrate that with the minimal sensing ability, our algorithms are able to achieve equivalent or better performance compared to the existing work.

While we have two different autonomous testbeds, they share the same sensor configuration, i.e., an odometry system, two 2D LIDARs, and one webcam. Table 1.1 lists the sensors used for the golfcart testbed. According to the objects that sensors measure, they can be classified into two types, i.e., the proprioceptive type (such as the odometry system) to measure a robot’s internal states, and the exteroceptive type (such as the LIDAR and vision sensors) to observe the surrounding environment. In our application, the odometry system is used to measure the vehicle’s orientation and displacement, which provides the necessary ego-motion information for vehicle localization as well as other perception functions. The exteroceptive sensors of our system include two 2D LIDARs and one webcam, where the tilted-down LIDAR is used to perceive the road surface and the nearby off-road structures, the planar LIDAR to handle the obstacles on a horizontal plane from a distance, and the webcam to provide visual clues of the environment. This sensor configuration provides the most essential sensing ability for vehicle autonomous driving, and the focus of this thesis is to realize the perception functions under this minimal sensing configuration.

Table 1.1: Sensor configuration for golfcart testbed

Sensor Type	Sensor Name		Modules
Proprioceptive	Odometry	IMU	MicroStrain 3DM-GX3-25
		Wheel Encoder	Scancon-2RS
Exteroceptive	LIDARs	Tilted-down LIDAR	SICK LMS-151
		Planar LIDAR	SICK LMS-151
	Vision	Webcam	Logitech C910

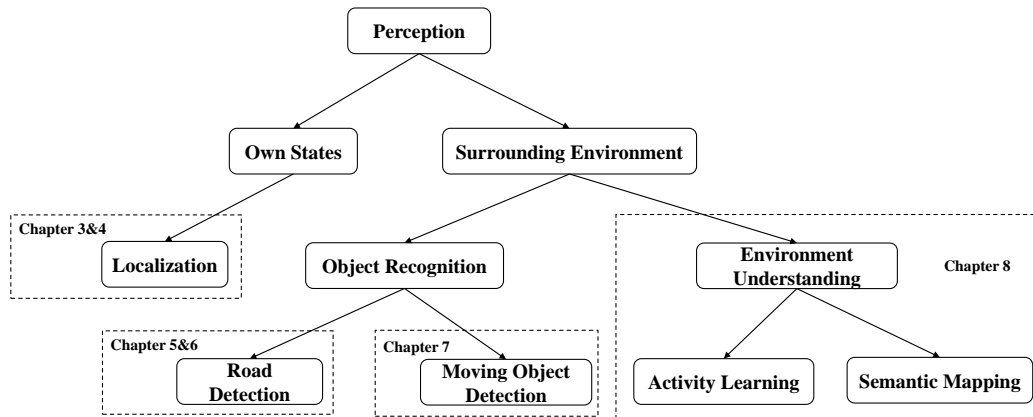


Figure 1.4: Thesis scope and outline

1.3 Scope of the Thesis

While perception is a broad research area covering a large variety of topics, this thesis only studies a small portion of them which are critical to vehicle autonomous driving. Figure 1.4 summarizes the perception functions studied in this thesis. From a broad view, these perception functions can be classified into two categories, i.e., functions to estimate the vehicle’s own states, and functions to perceive the vehicle’s surrounding environment. For own state estimation, the localization problem is studied, which is to estimate the vehicle pose in given global coordinates. For surrounding environment sensing, we are not only interested in short-term object-oriented recognition, for example, road detection and moving object recognition, but also interested in acquiring long-term environment-oriented understanding, such as a place’s semantic meaning, learning human activity patterns at this place, etc.

1.3.1 Localization

Mobile robot localization is the problem of determining the pose of a robot relative to a given map of the environment [16]. Localization is one fundamental requirement for vehicle autonomy. While other researchers rely on an expensive GPS/INS system or 3D sensor for high-accuracy localization, we acquire the ability using only the odometry system and a planar 2D LIDAR.

In the first stage of our research, we propose a Monte Carlo Localization (MCL) method using curb-intersection features on urban roads [6]. The curb, which defines the boundary of the road surface, is one of the most prominent features on an urban road. An intersection is a junction where two or more roads merge, appearing where no curb exists. The combination of curb and intersection features carries significant information about the urban road network, and can be exploited to localize a vehicle. We propose a novel idea of “synthetic LIDAR” to encode the two types of features into the format of laser scans, and integrate these synthetic laser scans into a MCL framework for precise localization. The proposed algorithm is implemented with only a single tilted-down 2D LIDAR and the odometry system, and achieves satisfactory results.

However, the curb-intersection-based algorithm only applies to the road segments where curbs exist, and may not be general enough for all the urban road scenarios. For this reason, in the second stage of the research, we consider incorporating other urban features for localization. In fact, there are actually many other salient features in the urban environment that can facilitate localization, such as lamp posts and building outlines. The common traits of these artificial objects (or structures) are that they all have vertical surfaces. To counter the limitations of the curb-intersection-based method and utilize the general artificial objects, we propose to use the “vertical surface” features for localization [8]. In the proposed method, a 3D point cloud is first accumulated with a tilted-down LIDAR in a rolling-window manner, from which the points cast on vertical surfaces are extracted. We introduce the idea of “Synthetic LIDAR” to compress the extracted points into a scan-like format, and use the synthesized scans for localization. Compared to the curb-intersection algorithm, the “vertical surface” algorithm is applicable to general urban road scenarios (with/without curbs), and has better localization accuracy.

1.3.2 Road Detection

Road surfaces are traversable areas where vehicles can safely navigate through, the detection of which is hence of much interest for vehicle autonomous driving. The results of road detection can not only serve as the guidance for vehicle path planning and control, but also provide the contextual information to solve other perception problems. Current research of road detection can be mainly classified into two categories, i.e., road marking detection and road surface-boundary detection, both of which will be covered in this thesis [9] [17].

Road Marking Detection. Road markings are the paintings on the road surface to provide traffic guidance information for vehicles and pedestrians. Common road markings include lane markings, arrows, characters, zebra crossing, etc. Road marking detection has been a popular research topic in the context of Autonomous Driver Assistance Systems (ADAS). Researchers aim to detect and locate the road markings, and utilize the results to provide driver assistance. While researchers have proposed various methods to detect the different types of markings, there is a lack of a general framework which supports all the detection purposes. In our work, we develop a general framework for road marking detection and analysis using vision [17]. Our basic idea is to extract individual markings and classify them based on their contours. Each type of marking will have its own dedicated classifier, which extracts the markings of interest, and filters out the rest. The recognized markings will be fed into an analysis process to analyze its guidance information. Unlike existing researches which only deal with certain specific types of markings, our proposed method is general enough to support a variety of marking types.

Road Surface-Boundary Detection. While marking detection is only applicable to the painted roads where markings exist, road surface-boundary detection is not limited to that. In our research, we try to detect the surfaces and boundaries of urban roads using a tilted-down 2D LIDAR.

In our initial research, road surfaces and boundaries are detected based on individual laser scans [6]. Sensing on an urban road, scans from a tilted-down LIDAR will show a piecewise property in their angle-range functions. Laser scans can be segmented utilizing this property, and the segmented scan pieces can then be classified as either road surface, boundary or background noise according to certain heuristic criteria. While this method achieves satisfactory results, it has a strong assumption about the sensing sce-

nario: road boundaries should always intersect the projected laser line on the ground, and there are at most two boundary points in each scan. This assumption does not apply to all the sensing scenarios, e.g. where road boundaries are actually parallel to the projected laser line. In addition, since the detection is based on individual measurement, it ignores the temporal relationship between adjacent scans, and is hence vulnerable to noise.

Considering these limitations, in the following research, we develop a new detection method using accumulated 3D data [9]. We introduce the idea of a 3D rolling window to maintain the data accumulated from 2D scans, and develop a cascaded process for the road detection purpose. Since this method relies on no assumption about the sensing scenario, it is able to deal with all the different situations. In addition, since the temporal relationship between consecutive scans can be well maintained in the 3D accumulation, this detection method is able to achieve better accuracy and more robust performance.

1.3.3 Moving Object Recognition

Since the urban road environment is shared by human beings, an artificially intelligent vehicle has to be able to recognize and live with these dynamic agents, pedestrians and vehicles for example. In our research into “dynamic” human agents recognition, a reduced problem of “moving” object recognition is studied: while every human agent has the potential to move - noted as “dynamic”, our attention is focused on recognizing those entities actually moving. We propose a spatial-temporal (ST) approach for moving object recognition using only modest sensory data [12]. Avoiding using more elaborate and costly solutions (e.g., outdoor depth cameras and 3D range finders), our method works with a simple planar 2D LIDAR on a mobile platform. Although the sparsity of sensor information from a 2D LIDAR complicates the detection task, we show that it is possible to obtain highly accurate object classification via temporal accumulation and a coupled classification process.

1.3.4 Environment Understanding

While researchers have spent significant efforts on the detection problems of various objects, getting a good understanding of the environment that hosts these objects and the robot is also of great importance. Unlike the object-oriented detection problems

which are focused on the short-term momentary recognition of objects in the robot’s local neighborhood, the environment-oriented understanding aims at acquiring a long-term consistent model at the global scale. The understanding of an environment can happen at different dimensions, such as the metric dimension, semantic dimension, etc.

In traditional studies, attention has been mostly focused on the metric dimension, where various algorithms have been proposed to build a consistent metric representation of the environment. The most representative work is SLAM (Simultaneously Localization and Mapping). However, the understanding of an environment is never limited to metric mapping, but extends to other more broad dimensions, such as the dimensions of activity and semantics. We argue that the activity and semantic information are two additional important dimensions of information for vehicle autonomous driving, and can be inferred from each other [11].

Activity Learning. Activities of human agents in the urban road environment are usually not erratic but follow certain patterns, which are implicitly determined by social norms and traffic rules. Knowing these motion patterns not only helps an autonomous vehicle predict human behaviors and intentions, but also enables it to perform human-like path planning. In our work, we propose an activity learning method using collected trajectories from a mobile platform. Firstly, pedestrians are detected and tracked using on-board sensors. Secondly, track classification and clustering are performed. Thirdly, the information of the tracks is registered into a grid map, and finally the pedestrian activity model is learned using Gaussian Processes (GP).

Semantic Mapping. Semantic mapping has become a popular research topic in recent years. By augmenting the traditional metric/topological maps with higher-level semantic knowledge, researchers aim to help robots to really “understand” their environments. A semantic map can not only facilitate human robot interaction, but also help a robot perform advanced reasoning and planning. Unlike existing research acquiring semantic knowledge through interpreting appearance features, we propose a novel method of semantic mapping by analyzing the learned activity patterns. While an environment serves as the space for different agents to conduct different activities, it can be divided into different functional areas, with each area corresponding to certain types of activities. For this reason, we can infer the semantic meaning of an area from its associated activity information.

In our implementation, we want to recognize the different functional areas for pedestrians in the urban road environment, i.e., “pedestrian path”, “entrance/exit”, “crossing” and “sidewalk”. By observing pedestrian activities over time, the semantic property of a place can be inferred from their motion patterns. A pedestrian activity model of the environment is first learned, and then 2D grid semantic mapping is performed by classifying the semantic properties of each grid cell with the learned activity model. Our proposed method is validated through experiments, and has shown promising results.

1.4 Thesis Outline

In summary, this thesis is focused on developing the perception ability for autonomous driving in the urban road environment. The contributions of this thesis are the minimal-sensing solutions for the fundamental perception functions described above. The thesis is organized as follows. Chapter 2 provides a general literature review about the perception techniques for vehicle autonomous driving. Chapter 3 and Chapter 4 study the problem of vehicle localization. Chapter 5-7 are focused on object-oriented recognition, where Chapter 5 and Chapter 6 introduce our research on road detection, and Chapter 7 presents our method of moving object recognition. In Chapter 8, environment-oriented understanding problems are addressed. Chapter 9 concludes the thesis and discusses the future work.

More supplementary materials can be found in the appendices, including the author’s publication list, and the video links for the overall project and for the perception functions studied in this thesis.

Chapter 2

Literature Review

Serving as the sensorium of an autonomous vehicle, the perception module plays a vital role for this artificially intelligent agent, enabling it to localize itself, approach the destination via drivable ground, and at the same time avoid collisions with other agents like pedestrians or vehicles. In this chapter, we study the development of autonomous vehicle technology worldwide, and discuss the perception modules of different autonomous vehicle projects. Three fundamental perception functions for vehicle autonomous driving are identified from the study, i.e. localization, object recognition, and environment understanding, which will be reviewed in detail in the subsequent sections.

2.1 Autonomous Vehicles and Their Perception Modules

2.1.1 A Brief History

The technology of autonomous vehicle dates back to the 1920s, when a radio-controlled driverless car was demonstrated on New York streets. While this vehicle was not really autonomous but remotely controlled, it is considered as the beginning of the era of autonomous vehicles [18]. Ever since then, the technology of vehicle autonomy has started its long-term evolution.

Early-stage autonomous vehicles worked in a lane-following way. They were able to perform lane-keeping, cruise control, collision avoidance and other basic operations. These systems were usually semi-autonomous in the sense that they had dedicated lanes, and needed human intervention from time to time. The perception modules in these vehicles were generally simple, where vision/magnetic sensors played an important role

for the lane detection and tracking purpose. Representative projects of this stage are the Prometheus Project in Europe and the California Path Project in USA [19].

To spur the innovation of autonomous vehicle technology, DARPA launched DGC and DUC. Autonomous vehicles in DGC were required to navigate through a 142-mile long desert course, with GPS way points provided two hours before the challenge [1]. Two major perception functions were required to complete the challenge, i.e. localization and drivable terrain detection. In the perception system of the challenge winner – Stanley, a high-accuracy GPS/INS device was used to solve the localization problem, and the terrain analysis was performed with both LIDAR and vision data [20].

Although DGC achieved great success, vehicles in this challenge were designed to drive in desert off-road terrain, and were not applicable to urban environments. To foster research into vehicle autonomy on urban roads, DARPA launched DUC in 2006 [2]. The challenge was held in a city-like environment, and participants were required to complete a 96 km course within 6 hours. In the challenge, the vehicles had to avoid other human-driven/autonomous vehicles, handle intersections and maneuver in car parking zones, while obeying all traffic regulations. DUC provided an invaluable chance for researchers to really understand the problem of autonomous driving in urban environments, and to recognize not only the challenges but also the opportunities. Research carried out in DUC stood as the state of the art at that time, and shed light on the development of autonomous vehicles nowadays.

We review the perception modules of the six finishers in DUC, and summarize their sensor configurations and perception functions in Table 2.1. It is observed that although these autonomous vehicles are designed independently and differently, they identify several common perception functions: vehicle localization, object/obstacle detection, object tracking, and road/lane detection. These perception functions actually provide the most basic perception ability for a vehicle navigating in the urban environment: to determine where to go, the vehicle has to be able to localize itself first; to safely navigate through the urban traffic the vehicle has to be able to detect other objects or obstacles; to predict the motions of dynamic agents the ability of object tracking is desired; last but not the least, road/lane detection is desired for the vehicle to drive on-road. Sensors of different modalities and types are placed around the DUC vehicles to realize the above perception functions: high-accuracy GPS/INS devices are employed for localization; Radars and

2D/3D LIDARs are mounted for object detection and tracking; cameras are used for the lane detection purposes; etc.

2.1.2 Current Trends

Seven years have passed since DUC. Nowadays autonomous vehicle technology is embracing its golden age – almost every automobile company is engaged in developing its own autonomous prototype. We review several current state-of-the-art projects [21] [22] [23] [24] [25], and summarize their perception modules in Table 2.2. Unlike the DUC vehicles equipped with redundant and expensive sensors regardless of the cost, current autonomous vehicles projects try to achieve equivalent perception ability with fewer and cheaper sensors. For example, it is noticed that some of the current projects are trying to get rid of the expensive GPS/INS devices, and use feature matching methods to solve the localization problem. It is also noticed that while 3D LIDARs are heavily relied on for object detection and tracking in the past DUC [2] and in the current Google project [21], to reduce the cost, some groups are researching to achieve equivalent detection ability using alternative cheaper sensory modalities [22] [23] [24].

Besides the interest in low-cost sensors, there are two other notable research trends in the current autonomous vehicle community. The first trend is to maximize the utilization of the vision modality. Back at the DUC time, vision was usually just used to recognize road markings or as a complement to other sensors like LIDAR. Nowadays, researchers are applying vision for various perception tasks, such as localization [26] [27] [28], real-time object recognition [29], terrain mapping [30], etc. The driving force behind this is that the vision modality is relatively cheaper, but able to provide rich information about the environment.

Another trend is to endow the vehicles with high-level intelligence of semantic understanding. While in the past years researchers have spent huge efforts to help robots build metric maps of their environments, nowadays much attention is turned to the semantic interpretation. By augmenting the traditional metric environment model with higher-level semantic knowledge, researchers want to help robots really “understand” the surrounding environments [31] [32].

2.1. Autonomous Vehicles and Their Perception Modules

Table 2.1: A summary of perception systems for the six finishers in DARPA Urban Challenge [2]. Several common perception functions are identified: vehicle localization, object/obstacle detection, object tracking, and road/lane detection. To realize these perception functions, various sensors of different modalities were used.

Ranking	Vehicle-Team Name	Sensors	Perception Functions	Brief Descriptions
1st	Boss-Tartan	<ul style="list-style-type: none"> • Applanix POS-LV 220/420 GPS/IMU; • SICK LMS LIDAR; • Velodyne HDL-64 LIDAR; • Continental ISF 172 LIDAR; • IBEO Alasca XT LIDAR; • Continental ARS 300 Radar; • Point Grey Firefly; 	Moving Object Detection and Tracking	Detect and track moving objects;
			Static Obstacle Detection and Mapping	Detect static obstacles, such as curbs;
			Localization	Localize the vehicle relative to road using road boundary information;
			Road Shape Estimation	Estimate the geometry of unknown roads;
2nd	Junior-Stanford	<ul style="list-style-type: none"> • Applanix POS-LV 220/420 GPS/IMU; • SICK LMS LIDAR; • RIEGL LMS-Q120 LIDAR; • Velodyne HDL-64E; • IBEO LIDARS; • BOSCH Long Range Radars LRR2; • Distance Measurement Unit; 	Laser Obstacle Detection	Detect general obstacles using LIDAR data;
			Static Mapping	Build a local map by accumulate static data overtime;
			Dynamic Object Detection and Tracking	Detect and track dynamic objects;
			Precise Localization	Localize the vehicle using road reflectivity and boundary information;
3rd	Odin-Victor Tango	<ul style="list-style-type: none"> • NovAtel GPS/INS; • SICK LMS LIDAR; • IBEO Alasca XT LIDARS; • IBEO Alasca A0 LIDARS; • Cameras; 	Object Classification	Identify both static and dynamic obstacles using IBEO software and vision;
			Localization	Compute vehicle global and local position using GPS/INS information;
			Road Detection	Detect road boundary and identify the drivable area;
			Dynamic Obstacle Precision	Predict likely paths of dynamic obstacles;
4th	Talus-MIT	<ul style="list-style-type: none"> • Applanix POS-LV 220 GPS/INS; • Velodyne HDL-64 LIDAR; • SICK LMS LIDARS; • Point Grey Firefly Cameras; • Delphi Radars 	Localization	Estimate vehicle global position using GPS/INS information;
			Obstacle Detection	Detect and track obstacles using 2D LIDAR and Velodyne data;
			Hazard Detection	Determine hazards the vehicle shouldn't drive over, such as curbs;
			Lane Finding	Identify the road using both vision and LIDARS;
5th or 6th	Little Ben-Franklin	<ul style="list-style-type: none"> • Oxford Technical Solutions RT-3050 unit,; • SICK LIDARS; • Hokuyo LIDARS; • Point Grey Bumblebee Stereo Camera; • Velodyne HDL-64E 	Localization	Localize vehicle pose using Oxford Technical Solutions RT-3050 unit;
			LIDAR Ground/Obstacle Detection	Extract ground plane and obstacles;
			LIDAR Lane Marking Detection	Detect lane markings using LIDAR reflectivity values;
			Dynamic Obstacle Tracking	Track moving objects;
			Vision Perception	Extract road markings using stereo vision;
			Mapping	Fuse perceptual measurements into a map;
5th or 6th	Skynet-Cornell	<ul style="list-style-type: none"> • GPS/INS System; • Velodyne HDL-64E LIDAR; • SICK LIDARS; • IBEO Alasca XT LIDARS • Delphi Radars; • MobilEye Camera; 	Obstacle Detection and Tracking	Detect and track obstacles using LIDAR, Radar and Vision Sensors;
			Localization	Localize the vehicle using GPS/INS system;
			Lane Marking Detection	Detect Lane markings using vision;

Table 2.2: A summary of perception systems for recent notable projects [21] [22] [23] [24] [25]. It is noticed that unlike the DUC vehicles equipped with redundant and expansive sensors, current autonomous vehicle projects try to achieve equivalent with fewer and cheaper sensors.

Research Group	Vehicle Name (if any) - Module	Sensors	Perception Functions	Brief Descriptions
Google	Prius / Lexus	<ul style="list-style-type: none"> • GPS/INS System; • Velodyne HDL-64E; • Radars; • Cameras 	Localization and Mapping	Perform vehicle localization and environment mapping using LIDAR intensity data;
			Object Recognition	Recognize different types of objects based on the 3D data from Velodyne;
			Moving Object Tracking	Track moving objects;
			Traffic Light Detection	Detect the traffic lights and their states;
CMU	Cadillac SRX	<ul style="list-style-type: none"> • GPS/INS System; • 2D LIDARs; • Radars; • Video Camera; • FLIR Camera 	Localization	Localize the vehicle in the road network using lane marking and road shape features, and GPS/INS system;
			Object Detection	Object detection using LIDAR, and vision;
			Lane Marking Detection	Lane marking detection using vision;
VisLab	BRAiVE - Hyundai Sonata	<ul style="list-style-type: none"> • GPS/INS System; • Firewire A Cameras; • Point Grey Cameras: FireFlyMV, Dragonfly2; • Hokuyo LIDARs • IBEO Lux LIDAR, • Hella IDIS LIDAR; • Radars 	Traffic Sign Detection	Detect traffic signs;
			Obstacle Detection	Detec obstacles using LIDAR and stereo vision;
			Object Recognition	Detect various objects like pedestrians, vehicles, etc.
			Terrain Mapping	Identify drivable area;
			Lane Marking Detection	Detect lane markings using stereo-cameras;
			Scene Classification	Classify the environment into different driving scenes;
Oxford	RobotCar – Nissan LEAF	<ul style="list-style-type: none"> • 2D LIDARs; • Stereo Cameras; • INS System 	Localization	Localize the vehicle using push-broom LIDAR or stereovision;
			Dynamic Object Detection	Detect and tracking dynamic objects;
			Environment Understanding	Learn the semantic model of the road environment;
Stanford	Junior - Volkswagen Passat	<ul style="list-style-type: none"> • Applanix POS-LV 220/420 GPS/IMU; • SICK LMS LIDAR; • Velodyne HDL-64E; • BOSCH Radars; • Point Grey Cameras: Ladybug3, Point Grey Flea2, Grasshopper. 	Mapping and Localization	Generate high-resolution intensity-based ground map, and localize the vehicle relative to it;
			Object Recognition	Recognize and track obstacles using Velodyne data;
			Traffic Light Detection	Detect traffic lights and their states using vison;
			Generic Sign Detection	Locate the position and orientations of road signs;

2.1.3 Discussion

The focus of this thesis is on developing the perception functions for vehicle autonomous driving. While it is impossible to cover all these topics in my Ph.D work, three fundamental and interesting questions are selected, i.e. localization, object recognition, and environment understanding. Localization is the problem of estimating the position of the vehicle, which provides the vehicle its spatial relationship to the surrounding environment as well as the destination. The localization ability is the most fundamental requirement for almost any mobile robot. Object recognition is the problem of detecting the existence of certain types of objects, ranging from static to dynamic entities. For a vehicle to safely navigate through an urban environment populated with various static structures and dynamic objects, robust object recognition becomes a critical requirement. While the abilities of localization and object recognition stand out as the most basic and necessary abilities for autonomous driving, environment understanding aims to help the vehicle really “understand” the environment that it lives in, endowing it with higher-level intelligence.

In our researches, we aim to realize the above perception functions with the idea of minimal sensing. Except for the odometry system, the major sensors of the vehicle are only one webcam and two planar LIDARs, as summarized in Table 1.1. We show that it is feasible to realize the studied perception functions with only such sensors. Existing literature of these perception topics will be reviewed in the following sections.

2.2 Localization

Mobile robot localization is the problem of determining the pose of a robot relative to a given map of the environment [16]. It is one fundamental requirement for vehicle autonomy.

Localization problems can be categorized into different types from different perspectives. Considering the availability of the initial position, localization can be distinguished as three types: position tracking, global localization, and the kidnapped robot problem. Considering the environment that a robot navigates, it can be distinguished as static environment localization and dynamic environment localization. From the viewpoint of motion control for localization, it can be categorized into the passive type and the active

type. From the viewpoint of the number of robots involved, it can be classified as single-robot localization and multi-robot localization. The taxonomy introduced here provides a good way to understand the natures and difficulties of different localization problems.

Existing approaches to localization can be categorized into four major streams: the GPS/INS fusion technique, the road matching technique, map-aided localization, and the SLAM (Simultaneous Localization and Mapping) technique. These approaches will be reviewed and discussed in detail.

2.2.1 GPS/INS Fusion Approach

In the past decades, researchers spent much effort on the fusion of a Global Positioning System (GPS) and an Inertial Navigation System (INS) to estimate vehicle position [33] [34] [35].

There are currently three GPS systems available, the American GPS, the Russian Glonass, and the China Beidou Navigation Satellite System. The European Union is developing their Galileo system, which is set to be completed by 2020. The basic operational idea of GPS is that a receiver measures the propagation time of satellite signals, and further calculate its distances to the satellites. These distances are called range estimates. When three or more satellites are available, position of the receiver can be determined by means of triangulation [36]. From the working manner of the GPS, it is found that the positioning accuracy depends heavily on the accuracy of the range estimates, which are subjected to common mode errors (such as ionospheric radio signal propagation delays, satellite clock and ephemeris errors) and non-common mode errors (such as multipath radio signal propagation, and receiver noise). To compensate the common mode errors, differential GPS (DGPS) technology is introduced. A DGPS uses a network of ground-based reference stations to calculate the difference between the positions indicated by the satellite system and the known fixed positions, and provides correction information for receivers in the coverage. There are also some satellite-based augmentation systems (SBASs) that use geostationary satellites for the same purpose.

However, even though the GPS accuracy is improved by the above augmentation techniques, a stand-alone GPS is still vulnerable to various types of noises, and has to be fused with other types of information to obtain desired accuracy, integrity and continuity. An Inertial Navigation System (INS) serves this purpose. An INS usually

includes one computational unit and one platform or module containing accelerometers, gyroscopes, or other motion-sensing devices. The sensing platform is usually an Inertial Measurement Unit (IMU) consisting of 3 accelerometers and 3 gyroscopes, providing linear acceleration and angular velocity information. The INS is usually provided with its initial position and velocity, and computes its own position and velocity by integrating the acceleration information [37]. The error of an INS comes from its use of integration: a small error in the linear acceleration or angular velocity will be integrated into an unbounded large drift in the position estimate. It is also found that the position error is proportional to linear acceleration multiplied by the square of time, and to angular velocity error by the cube of time [38].

The properties of an INS are complementary to those of GPS systems [36]. First, INS systems are self-contained and do not depend on external information which may be noisy or unavailable at some time, as opposite to a GPS. Second, an INS can provide position and angle updates at a quicker rate than a GPS. Third, the position error of an INS is unbounded, which can be bounded by the GPS input. The complementary facts listed above lead to the fusion of the GPS and the INS, which contributes to a more accurate, reliable and robust positioning system. An Extended Kalman Filter (EKF) is the traditional algorithm to fuse the two sources of information. The key idea underlying the EKF approximation is linearization, through which state transition and measurement functions are linearized to calculate corresponding minimum mean square error (MMSE) estimation. While the EKF method provides an easy solution to fuse the INS and the GPS, its goodness depends on the magnitude of noise and the nonlinearity of the two functions. The localization accuracy may be bad and the estimation may even diverge [16]. Some nonlinear filtering algorithms are proposed to solve this problem, such as the Unscented Kalman Filter (UKF), the Particle Filter (PF), and so on [39].

2.2.2 Road-Matching Approach

The performance of current GPS/INS systems rely heavily on the GPS signal quality. While an integrated system works well in open areas, its estimation can be erratic in urban areas due to severe satellite signal blockage and the multipath propagation effect. Road-matching algorithms are proposed to counter this problem. The basic idea of road matching is to treat road constraints of vehicle motion as observations. The road is

conceived as a line segment, with no specific lane width information. By checking the driven-on road segment against a road map, additional localization information can be derived.

Najjar *et al.* in [40] propose a road-matching localization algorithm, using Belief Theory for road selection and Kalman Filtering for recursive estimation. In the first stage, vehicle position is predicted according to odometry information, and corrected by GPS measurement, if available. In the second stage, with this estimate, the credible roads are selected based on multiple criteria combined by Belief Theory. Once a plausible road segment is selected as the driven-on road from the database, information from this road segment will be treated as an observation and used to update the predicted position from the first stage. Guivant *et al.* proposed one road-matching method with a particle filter in [41]. The basic idea is to penalize off-road particles based on road network information. Some other similar studies can be found in [42].

These road-matching algorithms achieve good localization in a global fashion, and have already been applied in some advanced driver assistance systems (ADAS). However, they are not designed to generate accurate position relative to road surfaces or road boundaries. In this sense, the localization is coarse-level localization, which may be inadequate for a fully autonomous vehicle performing complex tasks on a road surface.

2.2.3 Map-aided Approach

While the merit of road-matching techniques can be attributed to the fact that they rely on GPS as the one and only exteroceptive sensor, this fact also brings its inability to perform high-precision localization. To reach this goal, researchers have to use additional sensors. Local features are extracted to help realize precise position estimation, together with a prior feature map. Algorithms of this type are usually classified as map-aided localization.

In [43], single side curb features are extracted by a vertical LIDAR to improve vehicle localization together with one road boundary map. This map is learned beforehand in the form of line segments. This research is extended in [44], where a two-step algorithm is proposed. In the first step, a road-matching technique is applied to get preliminary position estimation at the coarse level. In the second step, the road curb is extracted as a local feature and incorporated into an EKF scheme to correct the estimated pose. While

these algorithms reduce the lateral localization error considerably, they help little in the longitudinal direction.

In [45], lane markings are utilized as the local features, which are extracted from the reflectivity values of LIDAR scans. One digital lane marking map is acquired beforehand. Performance of the algorithm is similar to those in [43] [44]. The lateral position estimation is precise, but the longitudinal error can only be reduced where the road curvature is big. Similar work can be found in [46].

Hentschel *et al.* in [47] use outer walls of solid buildings as local features. The landmark information is extracted from a 3D point cloud with the novel idea of Virtual 2D Scans [48]. The algorithm is tested in a campus environment, with the referenced map provided by the German land registration office. In [49], the group of researchers managed to apply the algorithm with the OpenStreetMap Geodata. The algorithm showed its effectiveness through experiments. However, possible absence of building features and slow update rate limit its effectiveness and usage.

Miller *et al.* in [50] use a particle filter to fuse GPS/INS information with map-referenced, vision-based road information for vehicle localization in challenging GPS environments. Three types of lane information are extracted from cameras. Similar to the previous algorithms, one map of accurate road lanes is provided. Experiments reveal that the algorithm improves significantly the quality of traditional GPS/INS positioning solutions, both when GPS signals are available and particularly when they are unavailable. However, the favorable condition of an accurate road line map is not always possible, and under bad illumination conditions this algorithm may not work.

Levinson *et al.* in [51] [15] utilize road surface reflectivity for precise localization. A particle filter is used to localize the vehicle in real time with a 3D Velodyne LIDAR. The algorithm first analyses the laser range data, and extracts those points cast on the ground. Then reflectivity measurements of these points are correlated to a map of ground reflectivity to update particle weights. One assumption underlying this algorithm is that road surfaces remain relatively constant, which may not hold in some cases. Besides, the need for costly 3D LIDAR sensor limits its usage.

Baldwin *et al.* in [52] utilizes accumulated laser sweeps as local features. The algorithm first generates a swathe of laser data by accumulating 2D laser scans from a tilted-down LIDAR. Then the swathe is matched to a prior 3D survey by minimizing

an objective function. This algorithm demonstrates its accuracy and robustness in GPS-denied areas. Although the algorithm proposed does not require an accurate 3D model of the environment, we argue that an accurate and consistent prior is desired when the localization is integrated with other navigation functions.

2.2.4 SLAM Approach

While the above map-aided methods achieve good performances, they assume the availability of precise prior maps, which however is not always guaranteed. Able to acquire the environment map and recover the robot pose at the same time, SLAM (Simultaneous Localization and Mapping) approaches are usually employed for the localization as well as the mapping purposes when a robot explores a new place.

From a probabilistic perspective, there are two types of SLAM approaches: online SLAM and full SLAM [16]. Online SLAM tries to estimate the posterior over the momentary pose together with the map, i.e. $p(x_t, m | z_{1:t}, u_{1:t})$, where x_t is the pose at time t , m is the map, and $z_{1:t}$ and $u_{1:t}$ are the measurements and controls. Full SLAM tries to estimate the entire path $x_{1:t}$ with the map, i.e. $p(x_{1:t}, m | z_{1:t}, u_{1:t})$. As in most probabilistic robotic problems, a state transition model $p(x_t | x_{t-1}, u_t)$ and a measurement model $p(z_t | x_t, m)$ are required to recursively update the desired probability distribution. SLAM has been a highly active research area in the last ten years, wherein many approaches have been developed [53] [54].

SLAM with an Extended Kalman filter (EKF) is the earliest and maybe the most influential algorithm. The EKF-SLAM algorithms belong to the online SLAM category, using maximum likelihood data association. They construct a combined state vector y_t of the robot pose x_t and the map m to estimate them simultaneously. This type of algorithm uses feature-based maps, and makes a Gaussian noise assumption for robot motion and perception. One of the key problems with EKF-SLAM lies in its computational cost. In the measurement correction step, the state vector and the joint covariance matrix are updated. Due to the dimension of the covariance matrix, the computational cost of this step grows quadratically with the number of landmarks. This limitation makes plain EKF-SLAM algorithms unsuitable for large-area mapping. Another problem with EKF-SLAM is its fragility to mismatching in data association [55]. This issue arises from the operation that only one landmark with the maximum likelihood is associated. If the as-

sociation is incorrect, the whole estimation process after this misalignment is influenced.

The FastSLAM algorithm applies a Rao-Blackwellized Particle Filter (RBPF) to solve SLAM problems, introduced by Montemerlo *et al.* in [56]. FastSLAM factorizes the SLAM posterior into two separate parts:

$$p(y_{1:t}|z_{1:t}, u_{1:t}, c_{1:t}) = p(x_{1:t}|z_{1:t}, u_{1:t}, c_{1:t}) \prod_{n=1}^N p(m_n|x_{1:t}, u_{1:t}, c_{1:t})$$

where $y_{1:t}$ is the combined state vector of the robot path $x_{1:t}$ and the map m . This decomposition is validated by the fact that features are conditionally independent from each other given the robot path. With the idea of RBPF, the above probability distribution is represented by a set of particles. Each particle carries its own path estimation, and the whole set of estimators for individual features. In the estimation process, the robot trajectory is estimated by the particle filter, while each map feature is estimated by an extended Kalman filter inbuilt in each particle.

FastSLAM outperforms the traditional EKF-SLAM in three respects. Firstly, it is computationally more efficient. FastSLAM uses a separate low-dimensional EKF for each individual feature, rather than using a single Gaussian to estimate the joint distribution of all features. Secondly, it handles the data association problem better. Unlike EKF-SLAM, which keeps a single best association, FastSLAM makes the data association decisions on a particle-by-particle basis. By virtue of multiple hypotheses, FastSLAM is more robust to the data association problem. Thirdly, particle filters ensure FastSLAM's ability to cope with nonlinearity in the robot motion model. The disadvantage of FastSLAM stems from degeneration of particle diversity which denotes the variety of all particles. Diversity is important, and it is always desired to maintain maximum diversity. While EKF-SLAM maintains correlation between different map features explicitly in the covariance matrix, FastSLAM maintains it through its diversity in particle sets. If the diversity decreases too much, accuracy of the algorithm will suffer. The FastSLAM algorithm is improved in [57], where measurement information z_t is also incorporated into the proposal distribution. This ensures that fewer particles are eliminated in the resampling step. Grisetti *et al.* manage to apply a similar idea to grid mapping in [58] [59].

Another important SLAM algorithm is GraphSLAM, which is a full SLAM algorithm rather than an online one [60]. GraphSLAM is essentially an information-theoretic technique. It treats control actions and measurements as constraints, and represents these

constraints in a sparse graph. GraphSLAM constructs a target function as the sum of these constraints. Minimizing this function will yield the maximum likelihood map and a corresponding set of robot poses. GraphSLAM is computationally more efficient and can acquire maps many orders of magnitude larger than EKF-SLAM. Besides, due to the fact that GraphSLAM has access to all data at the same time, data association can be more accurate than the EKF one. The limitation of this algorithm is that the constraint graph grows linearly over time, while EKF-SLAM has no such dependence.

Since SLAM has been a popular topic, many algorithms have been proposed. Range sensors, like sonar and LIDAR, have been the dominant sensors in the SLAM history. Recently, however, vision has gained more and more popularity in the appearance-only SLAM branch. Historically vision has been used for topological mapping and place recognition [61]. Color histograms are extracted as the global features of each room. In [62] Newman *et al.* utilize visual appearance signatures to detect and close the loop in 3D SLAM. M. Saedan *et al.* [63] present a vision SLAM algorithm with an omnidirectional camera in a hybrid map representation. More exciting research in [26] develops a biologically inspired method with vision for appearance-only SLAM, which demonstrates its ability of online localization and mapping during a 66 km journey through a suburban road network. Cummins *et al.* in [64] [65] propose a new algorithm of fast appearance-based mapping (FAB-MAP) based on the “Bag-of-Words” idea in the computer vision community. This algorithm proves to be both robust and computationally efficient. The above research progress shows that appearance-only SLAM with vision can be good a complement to metric SLAM methods.

Another trend in the SLAM area is to extend the SLAM idea from 2D to 3D environments. While 3D SLAM may be considered a straightforward extension, its complexity increases significantly due to the dimensions of the robot motion and measurement models. Kummerle *et al.* in [66] propose to use a multi-level volumetric surface map for 3D SLAM. Another representative research can be found in [67], where Nuchter *et al.* apply the method of Iterative Closest Point (ICP) scan matching for highly precise mapping.

2.2.5 Discussion

In our project, the ability of precise localization is pursued. While both the map-aided technique and SLAM technique are able to provide precise pose estimation, the map-

aided technique is adopted as our run-time localization technique. The considerations are as follows. Firstly, compared to the SLAM technique, which involves the high-dimensional map building problem and is hence computationally costly, the map-aided technique concentrates on the low-dimensional pose estimation problem, and is computationally much lighter and more feasible for online processing. Secondly, the SLAM technique is mainly developed for the exploration task in a new environment, and is neither necessary nor efficient for autonomous vehicles which probably navigate the same environment every day. In fact, the SLAM technique is usually applied to generate the map of an environment in an offline manner, which will then be used as the prior knowledge for online map-aided localization. (It should be mentioned that although some SLAM methods are applied in our research to generate metric maps, they are not the focus of this thesis and will not be covered in detail.)

In short, to avoid using expensive GPS/INS devices as well as to guarantee localization accuracy and efficiency, the map-aided technique is chosen in our research. Typical features of urban road environments are studied, and utilized for the localization purposes. In our initial research, road boundary features are extracted and utilized. While existing map-aided algorithms also utilize road features like curbs or lane markings for position estimation [45] [43] [44] [46], they only localize the vehicle well in the lateral direction of the road, but the accuracy in the longitudinal direction is not guaranteed. To counter this problem, we introduce an “intersection feature” as a complement to the curb feature, and utilize the combined curb-intersection feature of a road network for the localization purpose. Our algorithm achieves accurate estimation results in both the lateral and the longitudinal directions. Chapter 3 presents this localization method.

Although our initial work achieves satisfactory results, it is limited to the road segments where curbs exist. To make the localization more general and accurate, we turn to the use of general vertical surfaces in the typical urban environment. The algorithm achieves equivalent accuracy compared to the work in [51] [15], however, with a much cheaper 2D LIDAR sensor which has reduced sensing ability. Details of this work are presented in Chapter 4.

2.3 Object Recognition

Object recognition is the task of detecting and identifying the objects in the environment from sensor measurements. It has always been an active research domain, not only in the robotics community, but also in the communities of computer vision, machine learning, etc. For an autonomous vehicle, to reliably recognize other static/dynamic objects is a prerequisite for interacting with the environment and performing safe navigation. In this section, a general review of object recognition is first presented. While the topics of road detection and moving object recognition are more relevant to my research, studies on these two topics are then performed in detail.

2.3.1 A General Review

A tremendous amount of research has been performed in the field of object recognition, which may be categorized into different types with different criteria. According to the sensory modalities that are used, object recognition approaches can be classified as the vision-based approach [68] [69] [70] [71] [29], the LIDAR-based approach [72] [73] [74] [75] [76] [77] [78], the Radar-based approach [79] [80], and so on. In autonomous vehicle applications, vision and LIDAR are the two most important sensory modalities for object recognition, where the vision category can be further divided into the monocular/stereo types, and the LIDAR category can be divided into the 2D/3D types. While each sensory modality has its own advantages and disadvantages, in some applications multiple sensory modalities are fused to achieve better recognition performance [81] [82] [83]. Methods of this type can be viewed as the fused-modality approach.

According to the types of objects to be recognized, object recognition can be roughly categorized as the recognition of static structures, and the recognition of dynamic human agents. This categorization is based on the observation that an environment is composed of two distinct parts, i.e., the static structures fixed to it and the dynamic human agents living in it. The recognition of the static structures is focused on the roads [68] [84] and other traffic-related facilities [85] [86] [87], while the recognition of the dynamic part covers all the types of human agents in the traffic, such as pedestrians [29] [76] [73] [81], vehicles [69] [74] [88] [89], etc. The recognition of static structures and dynamic agents

is equally important for robot safe navigation.

While object recognition is a broad research topic, our attention is focused on two specific problems that are more relevant to vehicle autonomous driving, i.e., road detection and moving object recognition. Road detection has been a popular research topic for decades, and researchers have proposed various techniques to address this issue. A detailed review of road detection is presented in the following Section 2.3.2. Compared to the problem of road detection, the recognition of dynamic human agents appears much more challenging. The main difficulty comes from the high intraclass variance among human agents [69]. To be more specific, pedestrians may be dressed in different colors, vehicles may have different shapes and sizes, etc. To robustly and efficiently recognize these human agents remains a challenging problem nowadays. In our research of “dynamic” human agent recognition, a reduced problem of “moving” object recognition is studied: while every human agent has the potential to move - noted as “dynamic”, our attention is focused on recognizing those entities actually moving. A detailed review of moving object recognition is presented in Section 2.3.3.

2.3.2 Road Detection

Road surfaces are traversable areas on which vehicles can safely navigate. Road detection is one basic requirement for autonomous vehicle driving in the urban road environment. Road detection can provide not only the guidance for vehicle path planning and low-level control, but also the contextual information of the local environment for other perception purposes like vehicle detection and tracking. Given its importance, the problem of road detection has been widely studied. Current research into road detection mainly falls into two categories, i.e., road marking detection and road surface-boundary detection.

Road marking detection has been studied for years in the context of Autonomous Driver Assistance Systems (ADAS). Researchers aim to detect and locate road markings on the road, and utilize the results for lane departure warning, adaptive cruise control and other purposes. Among all the different types of markings, lane markings are the dominant markings on the road surface, and have attracted most of the research interests. Aly [68] presents a real-time algorithm to detect lane markings on urban streets. An Inverse Perspective Mapping process is first applied to generate a bird’s-eye view

of the road surface from the original image. The transformed image is then filtered by a horizontal Gaussian filter and then thresholded to extract pixels corresponding to the vertical lanes. In the final step, Hough Transform and RANSAC methods are used to get a mathematically compact representation of the detection results. Similar work can be found in [90] [91] [92]. Compared to the tremendous efforts spent on lane marking detection, very little research has been carried out for other types of markings, such as arrows and zebra crossings, which however also convey important traffic guidance information, and deserve more attention. Vacek *et al.* [93] use a template matching method to detect arrow markings. Li *et al.* [94] use marking shape information to detect both lanes and arrows. Se [95] deals with zebra-crossing detection by grouping concurrent lines in the camera image.

While marking detection is only applicable to the structured roads with painted markings, road surface-boundary detection can be applied to both structured and unstructured roads. LIDARs have played a dominant role in this area. Thrun *et al.* use LIDARs to carry out Probabilistic Terrain Analysis (PTA) in desert driving in [20] [96]. The driving terrain is represented by a 2D grid map, and the grid cells are then classified into different terrain types according to the height differences in their local neighborhoods. Zhang proposes a road boundary detection algorithm for urban roads in [84]. A Gaussian differential filter is applied to the range values of each laser scan, and road boundary points are extracted as local maxima of the filter response. In other similar research, Cramer *et al.* applied Hough Line for scan segmentation and feature extraction in [97], while Kodagoda *et al.* achieved the same goal by using an EKF filter in [98] [99]. Our previous work also proposes a simple road boundary detection algorithm using a tilted-down LIDAR in [6]. One common feature of the above algorithms is that they all operate on a single individual 2D scan for road boundary detection. Stereo vision is another sensory modality that has been used for road boundary detection. Related work can be found in [100] [101].

2.3.3 Moving Object Recognition

Moving object recognition helps autonomous vehicles recognize and live with other dynamic agents. Existing work in moving object recognition decomposes the problem into two distinct sub-tasks: detection and classification. The former aims to discern the exis-

tence of moving objects, while the latter aims to recognize the objects' identities. Considering the ways of moving evidence detection, we categorize existing methods into two types, the tracking-based type and the SLAMMOT (short for simultaneous localization, mapping, and moving object tracking) type.

Tracking-based methods (e.g. [81] [102]) work at the object level: they first segment a laser scan into multiple segments, which are considered as the measurements of different objects; the segments are then fed into trackers to estimate their positions and velocities; objects exceeding a defined speed or displacement are reported as moving objects. The accuracies of these methods are mainly determined by the tracking process, which has to solve the notorious data association problem and may easily fail in cluttered environments.

Unlike the tracking-based methods, SLAMMOT methods detect moving objects at the atomic level [103] [104] [105]. An occupancy grid map of the local environment is created through a SLAM process and the changes in the grid cell's occupancy statuses indicate the existence of moving objects. Compared to the tracking-based methods, the SLAMMOT techniques have two major advantages. Firstly, they are more robust to ego-motion estimation errors (which are compensated by the SLAM process). Secondly, they don't have to address the tracking problem. However, the computational cost associated with SLAM is usually high, leading to low update frequency. This is undesired for robots such as autonomous vehicles, which move at relatively high speeds. Besides, since these methods assume that robots move on a flat ground, they are not applicable to bumpy road environments.

In both types of methods discussed above, object classification is performed independently from motion detection, either before or after one object is identified as "moving". The classification process usually relies on the sparse geometric features of the 2D segments, and appears vulnerable to similar-looking background noise. To achieve better performance, classification results of measurements received at different times can be fused for continuous estimation, with the premise that object tracking is carried out accurately.

2.3.4 Discussion

Object recognition is one of the fundamental requirements for vehicle autonomous navigation. Among all the research associated with it, the topics of road detection and moving object recognition will be addressed in this thesis.

Road Detection. Both the marking detection problem and surface-boundary detection problem will be studied in this thesis. For road marking detection, although there has been a large amount of research addressing this problem, researchers tend to develop dedicated methods to detect certain specific types of markings, and there is a lack of a general approach which supports all the detection purposes. For this reason, we develop a general framework for marking detection and analysis, and the proposed approach is presented in Chapter 5.

For the purpose of road surface-boundary detection, we propose a recognition method using accumulated 3D data. While most existing algorithms operate on individual 2D scans, they usually have strong assumptions about the sensing scenario, which limit their applications. Furthermore, these algorithms fail to capture the temporal relationships between adjacent measurements, and appear vulnerable to noise. However, our method employs 3D perception techniques, and is able to handle the above problems. Chapter 6 presents our algorithm in detail.

Moving Object Recognition. While two types of moving object recognition methods have been reviewed, both of them have their limitations: the tracking-based methods have to solve the data association problem, and may fail in cluttered environments; the SLAMMOT method has to perform SLAM during the detection, which limits its speed. In this thesis, we introduce a spatial-temporal approach for moving object recognition, which does not rely on tracking or SLAM, and is able to perform reliable real-time recognition in both clear and cluttered environments. The algorithm is presented in Chapter 7.

2.4 Environment Understanding

For vehicle autonomous navigation in the urban environment, object recognition functions are developed to detect and identify different objects of interest, as discussed in the previous sections. The detection is performed online in the local neighborhood of the

own vehicle, to help it handle various momentary conditions reactively. While the detection is generally short-term and object-oriented, some long-term understanding about the environment is also desired. Environment understanding targets acquiring long-term knowledge about the environment. This knowledge, also called the model, serves as the prior information for robot autonomous navigation, which can be used to help vehicle localization, path planning, and all the other purposes.

The knowledge of an environment can be about any dimension of its properties, such as its spatial layout, its temperature, the types of objects placed in it, etc. In our research, three dimensions of knowledge important for autonomous navigation are identified: the **metric dimension** about an environment’s geometric layout, the **semantic dimension** about the semantic meanings of different places, and the **activity dimension** about the activity patterns of human agents living in it.

In traditional robotics studies, enormous efforts have been spent on the problem of metric mapping to derive a metric model of the environment. One popular approach of metric mapping is SLAM, as reviewed in Section 2.2.4. The output of metric mapping is a metric model of the environment, which captures its geometric layout and is mainly used for localization. Nowadays researchers are trying to augment traditional metric maps with some high-level knowledge, such as semantic information and activity information, to help the robot to really understand its environment. While metric mapping is out of the scope of this thesis, our attention is put on learning the semantic and activity knowledge. Related work on semantic mapping and activity learning is reviewed in the Section 2.4.1 and Section 2.4.2, respectively.

2.4.1 Semantic Mapping

Semantic mapping, which is the process of learning the semantic model of an environment, has become a popular research topic in recent years. A semantic map can not only facilitate human-robot interaction, but also help a robot perform high-level reasoning and planning. In the past few years, various methods have been proposed for semantic mapping. Depending on the sources of semantic information, these methods can be roughly classified into three categories: the appearance-based approach, the object-based approach, and the activity-based approach.

The appearance-based approach is the most popular approach in semantic learning

research, where semantic knowledge is acquired by interpreting appearance features from sensory data. In [106], O. M. Mozos *et al.* use geometric features from a planar laser range finder for indoor place classification. This work is extended to incorporate vision features for better and finer classification in [107]. Similar researches have been done by Pronobis *et al.* in [108]. In [109], I. Posner *et al.* use fused vision and 3D laser data for semantic labeling of urban scenes. Visual features and 2D/3D geometric features are extracted and fed into a hierarchical classifier for scene recognition. In [31], S. Sengupta *et al.* present a dense semantic mapping method for an urban street environment, based on vision features extracted from a sequence of street-level imagery. Some other appearance-based semantic mapping studies can be found in [110].

Unlike the appearance-based approaches where semantics is directly learned from sensor readings, the object-based approaches infer the semantic meaning of an environment by checking the occurrence of key objects inside. In [111], C. Galindo *et al.* infer the semantic type of a room by detecting the typical objects in it. In [112], S. Vasudevan *et al.* propose to perform place classification using not only object count information, but also the position relationships between objects.

The activity-based approach learns the semantic knowledge of an environment based on agent activities in it. Compared to the extensive literature of the appearance-based approach, less research is found in this category. In [113], D. F. Wolf *et al.* build a 2D semantic grid map according to the occupancy status of the space by dynamic entities. Activity-related features are extracted to classify a place into two semantic types, “street” or “sidewalk”. In [114], D. Xie *et al.* present a method to localize functional objects that affect people’s behaviors in surveillance videos. In [115], G. Li *et al.* infer the furniture types in a room based on human activity recognized from wearable sensors. A pre-learned activity-to-furniture model is used.

2.4.2 Activity Learning

While an environment serves as the space for different agents to perform different activities, these activities usually follow certain patterns, which can be learned and aggregated into an activity map. The activity information can be considered the dynamic knowledge of the environment, knowledge of which will bring multiple benefits: for example, given the activity knowledge, a robot can understand the traffic flow at a certain place, infer

social rules and perform human-like planning and controlling; the activity knowledge of human agents can also help the robot to realize robust tracking, predict their intentions, and improve interaction with them; etc.

Activity learning is not a new topic in the computer vision community, where researchers have proposed various methods to learn the human motion patterns in an environment. N. Johnson *et al.* in [116] model the probability distribution of pedestrian trajectories through a vector quantization method using a neural network. Swears *et al.* [117] use hierarchical clustering of Hidden Markov Models to learn motion behaviors in video surveillance. Ellis *et al.* [118] apply an algorithm of Gaussian Process Regression (GPR) to model pedestrian trajectory patterns. Some other representative work can be found in [119]. However, most of the above algorithms use a stationary camera and assume the observability of complete trajectories, which is not a valid assumption for mobile robot applications.

Lookingbill *et al.* in [120] use a helicopter to learn the motion patterns of moving objects on the ground. Objects are tracked with multiple particle filters, with each particle containing the position and velocity information. The activity map is represented by a 4-dimensional histogram $h(x, y, v, \theta)$ of the particles, which is indexed by x-y locations in the camera plane and represented by velocity magnitude v and direction θ . Bennewitz *et al.* [121] manage to learn motion patterns using mobile service robots. Laser range finders are mounted on mobile robots to record human trajectories. The Expectation Maximization (EM) algorithm is applied to cluster different types of motion, and learn the corresponding patterns simultaneously. Sehestedt *et al.* [122] propose a method to learn the human motion patterns in an office-like environment based on Sampled Hidden Markov Models (SHMM). The advantage of this method is allowing online and unsupervised learning.

2.4.3 Discussion

Besides metric mapping, semantic mapping and activity learning are attracting more and more research interest. While the three dimensions of knowledge are highly correlated with each other, existing studies usually focus on one single specific individual dimension, and there is no research utilizing their correlations for knowledge inference. In this thesis, we propose the representation of a multi-dimensional grid map, and argue that the

environment properties of different dimensions are correlated and hence can be learned from each other. As an implementation of this idea, we develop a semantic mapping approach by analyzing the activity patterns of human agents. Details of this work are presented in Chapter 8.

2.5 Summary

This chapter studies the history and current status of autonomous vehicle technology, summarizes the fundamental perception requirements for autonomous driving in the urban environment, and discusses the ongoing research trends. Three perception problems are selected and reviewed in detail: localization, object recognition and environment understanding. Our researches on these three perception tasks are the main body of this thesis, and are discussed in the following chapters.

Chapter 3

Curb-Intersection Feature based Monte Carlo Localization

One of the most prominent features on an urban road is the curb, which defines the boundary of a road surface. An intersection is a junction of two or more roads, appearing where no curb exists. The combination of curb and intersection features gives a complete picture of the urban road network, and can be exploited to improve a vehicle’s localization. This chapter presents a curb-intersection feature based Monte Carlo Localization (MCL) method for precise vehicle localization in the urban road environment.

3.1 Introduction

Localization is one fundamental requirement for vehicle autonomous navigation. Local features have been utilized to realize precise vehicle localization, as reviewed in Section 2.2. In [43] [44] [45] [46], researchers propose to utilize curb features for vehicle localization. However, due to the lack of the longitudinal localization information in the curb features, these methods only localize the vehicle well in the lateral direction of the road, but help little in the longitudinal direction. To counter this problem, we introduce an “intersection feature” as a complement to the curb feature for the localization purpose. The intersection features appear at junctions of roads, where no curb exists. While curb features mostly help localize a vehicle laterally in the road, intersection features carry rich longitudinal information. The complementary nature of these two kinds of features makes them well suited for localization. With this idea, we propose a curb-intersection

feature based Monte Carlo Localization algorithm for precise vehicle localization. The proposed algorithm is implemented using only a single tilted-down 2D LIDAR and the odometry system, and achieves accurate estimation results in both the lateral and the longitudinal directions.

The contributions of our research are two-fold. Firstly, we introduce a new “intersection feature” as a complement to the curb feature for vehicle localization. Secondly, we propose a novel idea of “synthetic LIDAR” to represent the curb-intersection features, which enables us to use the standard measurement models of laser sensors for accurate and robust localization.

The remainder of this chapter is organized as follows. Section 3.2 gives an overview of the localization system. In Section 3.3, the extraction of curb and intersection features is introduced. Section 3.4 presents our idea of synthetic LIDAR. Section 3.5 provides details of the curb-intersection-based MCL method. Experiment results and analyses are presented in Section 3.6. Finally, Section 3.7 concludes this chapter.

3.2 System Overview

The localization system mainly consists of three parts: curb-intersection feature extraction, synthetic LIDAR construction, and the Monte Carlo Localization process, as shown in Figure 3.1. A tilted-down 2D LIDAR is utilized to extract the curb and intersection features from urban roads. Synthetic LIDARs are constructed by encoding the extracted curb-intersection features into the format of laser scans. Vehicle pose estimation is carried out through a Monte Carlo Localization process, where particles are propagated in the prediction step with the odometry information, and the particle weights are updated in the correction step with the measurements from the synthetic LIDARs. Particles are resampled in the resampling step to avoid the degeneration problem.

3.3 Curb-Intersection Feature Extraction

There are numerous studies on road boundary detection. One method is the use of a tilted-down LIDAR for curb detection. Cramer *et al.* [97] applied Hough Line for scan segmentation and feature extraction, while Kodagoda *et al.* [98] [99] achieved the same

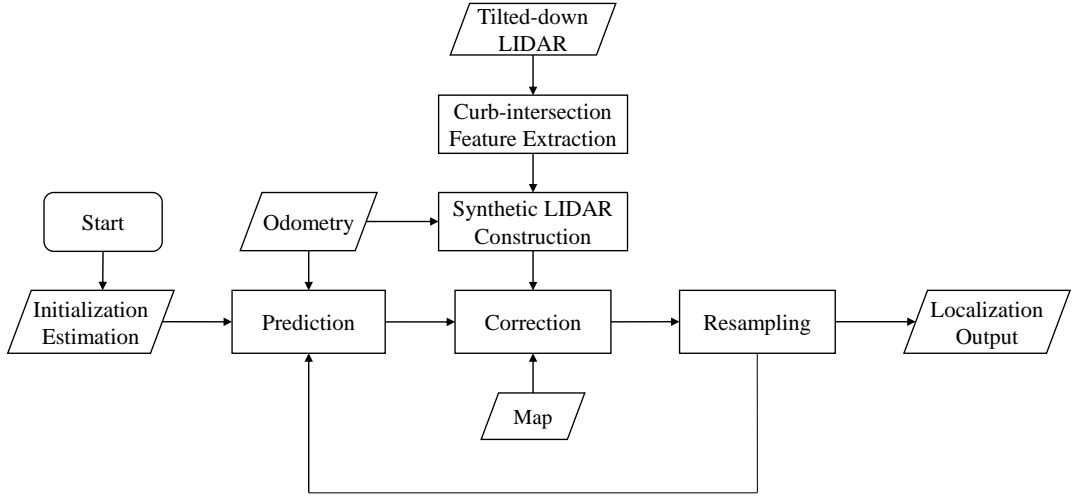


Figure 3.1: Curb-intersection feature based localization flowchart

goal by using an EKF filter. This section presents an intuitive two-step method to detect both curb and intersection, which proves to be efficient and robust. Similar work can be found in [84].

3.3.1 Segmentation of Laser Scan

In the first step, one single laser scan is segmented into several pieces, by virtue of its typical laser range-angle characteristics on-road.

Figure 3.2 shows the model of sensing on-road. One LIDAR sensor is mounted at point O , with its titled-down angle as α and mounting height as H . O' is the projection of O on the road surface, P is the center of the line of intersection between the laser scan and the road surface, and the distance between point P and O' is the look-ahead distance of the tilted-down LIDAR. The angle between the vehicle's heading direction and the road is denoted as φ . The angle of the laser beam is denoted as θ . As presented in Figure 3.2, laser beams from point O are cast onto different planes, i.e. road surface plane, curb plane, road shoulder plane, and so on. From this model, a piecewise function can be derived to represent the relationship between the beam angle and range value, with each interval corresponding to an individual plane. Without involving too many details, a simplified formula can be represented as:

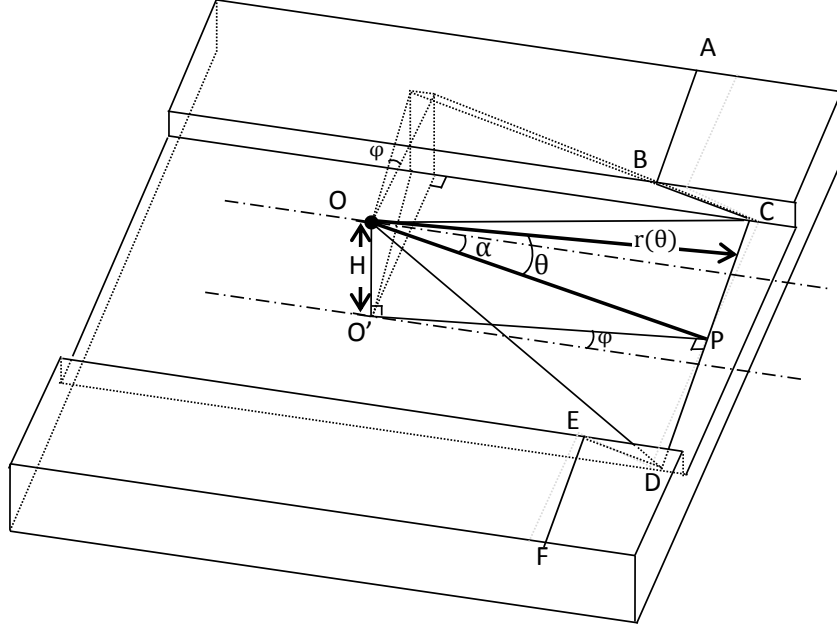


Figure 3.2: Model of LIDAR sensing on-road

$$r(\theta) = \begin{cases} \dots & \dots & \dots \\ R_{leftCurb}(\theta) & \text{for } \theta_C \leq \theta \leq \theta_B \\ R_{roadSurf}(\theta) & \text{for } \theta_D \leq \theta \leq \theta_C \\ R_{rightCurb}(\theta) & \text{for } \theta_E \leq \theta \leq \theta_D \\ \dots & \dots & \dots \end{cases} \quad (3.1)$$

Due to the piecewise nature of function $r(\theta)$, a second-order differential filter can be implemented to detect the edges:

$$r_f(\theta) = \sum_{i=-5}^{i=-3} r(\theta + i \times \mu) + \sum_{i=3}^{i=5} r(\theta + i \times \mu) - \sum_{i=-2}^{i=0} r(\theta + i \times \mu) - \sum_{i=0}^{i=2} r(\theta + i \times \mu) \quad (3.2)$$

where μ is the angular resolution of the LIDAR sensor, and $\theta \in [-\pi/2 + 5\mu, \pi/2 - 5\mu]$. Boundary points are extracted as local maxima or minima in the filter response plot, and their values should exceed a certain threshold, as shown in Figure 3.3.

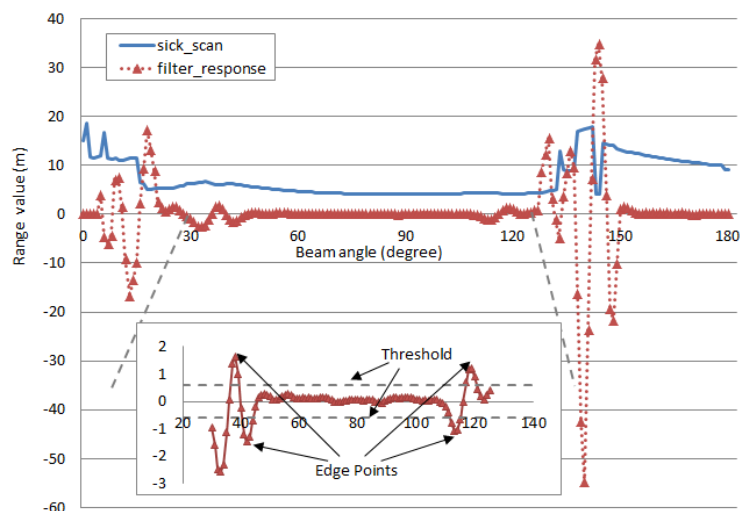


Figure 3.3: Raw LIDAR reading and filter response

3.3.2 Classification of Scan Segments

In the second step, scan segments generated are fed into a sequential classification process:

1. The road surface segment, shown as line CD in Figure 3.2, is selected first. It is always located between the two edge points nearest to the center of the sensor.
2. Curb lines, (BC and DE of Figure 3.2), are searched for subsequently, based on point C and D determined from the former step.
3. The remaining segments are other features off the road.

Some restrictive criteria are applied during the above steps, such as road width, curb height, etc. Specifically, to extract a valid curb feature, the length of segment CD should be bigger than the minimum value of road width, the curb height of segment BC (or DE) should be within a certain range, and the number of laser points on BC (or DE) should be over a certain threshold, etc. Only when all these criteria are satisfied is the classification result considered valid. Thus most noise like vehicles and pedestrians gets filtered. One typical classification result is shown in Figure 3.4. Among these classified laser segments, the curb segments are saved for further usage.

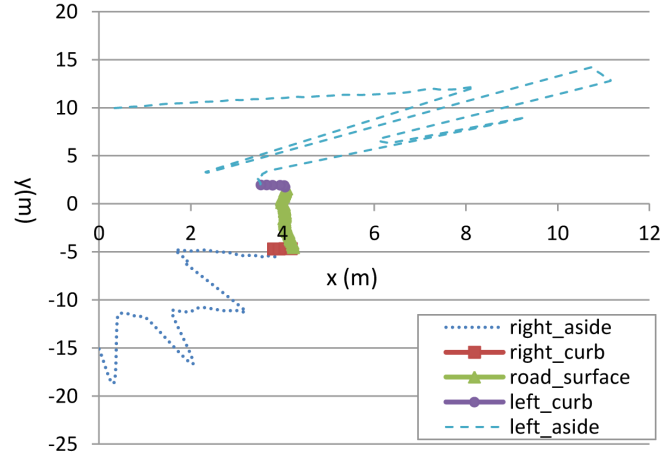


Figure 3.4: Scan segment classification

3.3.3 Intersection Feature

It should be clarified that, a fixed maximum detection range is defined for the above curb extraction algorithm. If the distance of the curb edge (C or D) to the projected center O' exceeds this range, the curb features are deemed unreliable, and will yield a “no-curb” result. For this reason, the above algorithm does not apply to road intersections, where the curbs are too far away, or there may be no curb at all. However, the “no-curb” result actually carries significant localization information, which tells the vehicle that it has arrived at some critical points in the road network. To embody this kind of information, a novel “intersection feature” is introduced.

As shown in Figure 3.5, an intersection feature is represented by a synthetic beam \overrightarrow{PR} (or \overrightarrow{PL} if it points to the left), with its direction perpendicular to $\overrightarrow{O'P}$ and its distance the maximum detection range of curb extraction. It should be clarified that feature extraction of left and right sides are independent, enabling both curb and intersection feature extraction at a T-junction. Implementation of curb-intersection features for Monte Carlo Localization will be discussed in the following sections.

3.4 Synthetic LIDAR Construction

3.4.1 Synthetic LIDARs in the Horizontal Plane

As discussed in the previous section, the features extracted here are curb segments (line BC and DE), or synthetic beams (\overrightarrow{PR} and \overrightarrow{PL}). Because the curb and intersection

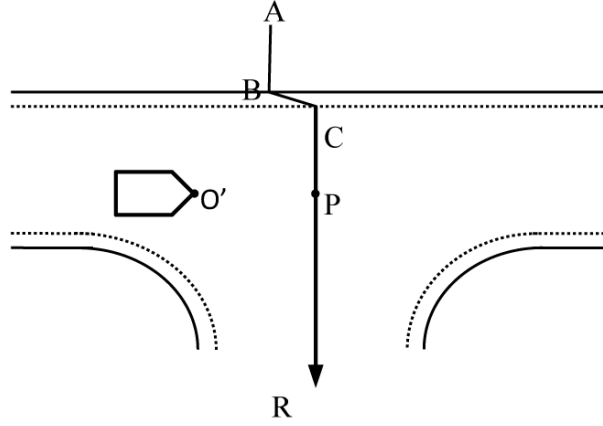


Figure 3.5: Curb-intersection feature at a T-junction

features are extracted using a tilted-down LIDAR in the 3D space, building their measurement models as well as utilizing them for localization is not an easy task. To simplify the task, we project the extracted features in the horizontal plane ($z = 0$), and introduce the novel idea of “synthetic LIDAR” to encode the features into the format of laser scans, as illustrated by Figure 3.6.

For curb features, segments BC and DE are projected as $B'C$ and DE' . Let Q_i be a random point on BC (or DE), and its image on $B'C$ is Q'_i . A synthetic laser beam can be conceived of as $\overrightarrow{O'Q'_i}$, as shown in Figure 3.6(a). In this way, a synthetic planar LIDAR (LIDAR-V1) can be built, with its origin located at O' , and the laser beams ending at the projections of the extracted curb lines, as illustrated by Figure 3.6(b). This LIDAR is somehow exotic: it can only see curb lines (since its beams are only constructed based on the extracted curb points), and its beam angle is not evenly spaced (while the angles of the original laser beams in the slanted plane \overrightarrow{OQ} are evenly spaced, it can be proved that the angles of the synthesized beams $\overrightarrow{O'Q'_i}$ are not due to the projection process). The full scale range of this synthetic LIDAR is the maximum detection range for the curb. With similar ideas, LIDAR-V2 centered at P can be modeled for intersection features. LIDAR-V2, different from LIDAR-V1, has at most two synthetic beams, \overrightarrow{PR} and \overrightarrow{PL} , with its range values always the maximum detection range for curb.

3.4.2 Scan-Assembled Synthetic LIDARs

With two synthetic LIDARs established, the MCL problem using curb-intersection features is reduced to a common MCL problem with planar LIDARs. However, because

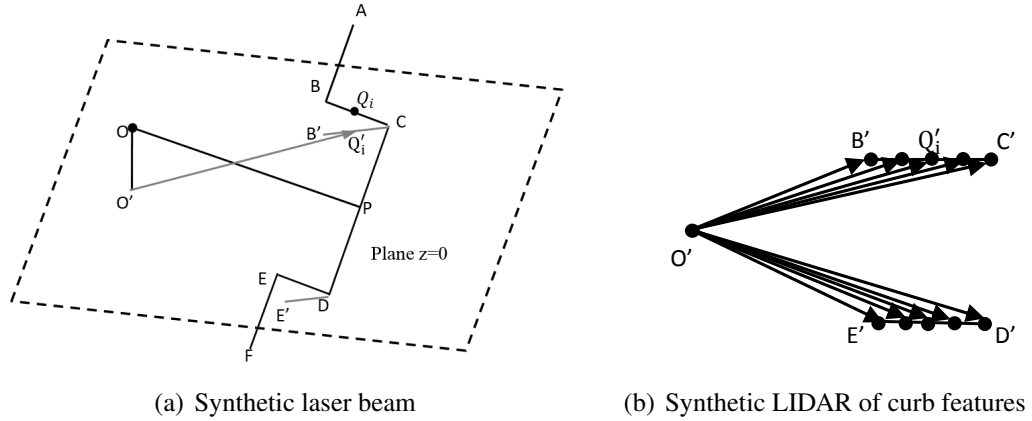


Figure 3.6: Synthetic LIDAR construction

scans of the synthetic LIDARs carry much sparser information than real ones, it is advisable to assemble several scans at different times into one. The assumption validating this operation is that odometry remains accurate within a short distance interval. The new assembled synthetic scan can then be treated as a normal laser scan, and fed into the MCL processing.

During the scan assembling of LIDAR-V1, to reduce the computational cost, only two curb points (C and D) are retained for each scan. Curb points recorded at different times are then translated into the latest LIDAR coordinate, serving as endpoints of synthetic laser beams cast from a new synthetic LIDAR, denoted as LIDAR-VSA1. As for LIDAR-V2, two synthetic beams are recorded at different times, together with their casting origins. The new synthetic assembled LIDAR is even more exotic: it is composed of several 2-beam (or 1-beam at a T-junction) range finders, with each finder mounted at different positions with different angles. When the beam number of LIDAR-VSA1 (or LIDAR-VSA2) exceeds a certain Assembling Threshold, one synthetic measurement is published. Finally, we get two new synthetic LIDARs: LIDAR-VSA1 for curb features, and LIDAR-VSA2 for intersection features, as shown in Figure 3.7.

3.5 Monte Carlo Localization Algorithm

3.5.1 MCL Overview

In this chapter, Monte Carlo Localization (MCL) is applied to estimate the vehicle pose. MCL is a probabilistic localization algorithm based on Bayes Theorem and the Monte

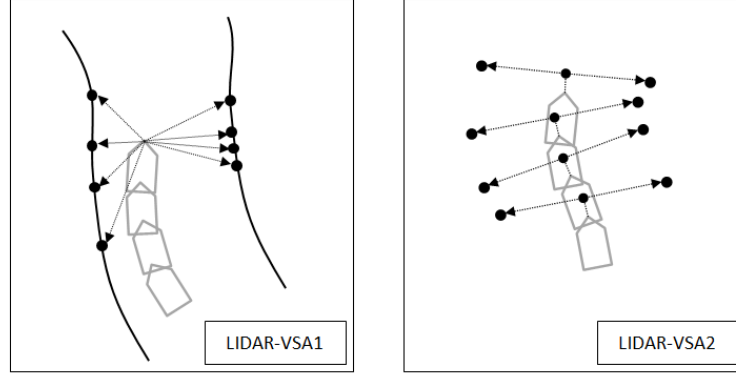


Figure 3.7: Assembled synthetic LIDARs

Carlo method. A thorough study is made by Sebastian Thrun *et al.* [16]. The belief $bel(x_t)$ in MCL is represented by a set of M particles $x_t^{[m]}$, and each particle is paired with an importance weight $w_t^{[m]}$:

$$bel(x_t) \sim \{x_t^{[m]}, w_t^{[m]}\}_{m=1}^M \quad (3.3)$$

MCL estimates the position of the vehicle recursively by repeating the following steps:

1. Prediction: a new set of particles $\{x_t^{[m]}, w_t^{[m]}\}_{m=1}^M$ for time t is generated with $\{x_{t-1}^{[m]}, w_{t-1}^{[m]}\}_{m=1}^M$ and the control u^t , according to a certain motion model $p(x_t|u_t, x_{t-1})$.
2. Correction: the importance weight of each particle in $\{x_t^{[m]}, w_t^{[m]}\}_{m=1}^M$ is adjusted with new measurements z^t , according to a certain measurement model $p(z_t|x_t, m)$.
3. Resampling: the particle set $\{x_t^{[m]}, w_t^{[m]}\}_{m=1}^M$ will be resampled when necessary. After resampling, the distribution of the particles approximates $bel(x_t)$.

3.5.2 Pseudo-3D Odometry Motion Model

In the prediction step, a motion model is applied to propagate particles for the prior belief distribution $\overline{bel}(x_t)$. Generally, a 2D motion model in [16] is enough. Even for a vehicle moving in a 3D world, we solve the localization problem on its horizontal projection plane, as shown in Figure 3.8. Here we extend the 2D motion model to a Pseudo-3D one, by introducing a pitch noise part.

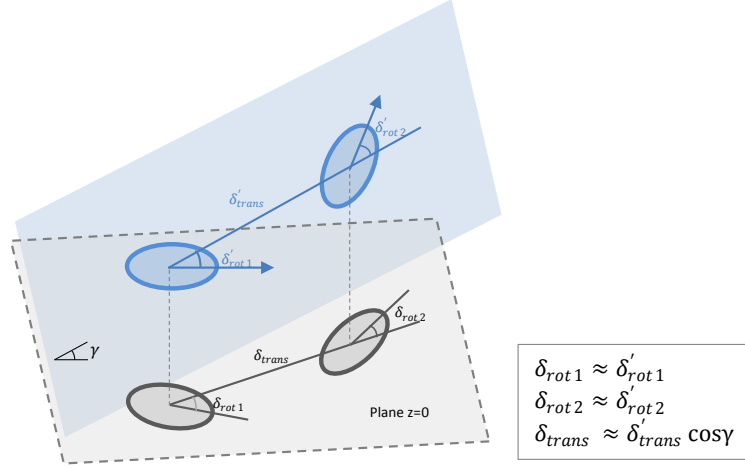


Figure 3.8: Pseudo-3D localization

Table 3.1: Pseudo-3D Odometry Sample Motion Model (u_t, x_{t-1})

1. $\hat{\delta}_{rot1}$	$= \delta_{rot1} - \text{sample}(\alpha_1 \delta_{rot1}^2 + \alpha_2 \delta_{trans}^2)$
2. $\hat{\delta}_{trans}$	$= \delta_{trans} - \text{sample}(\alpha_3 \delta_{trans}^2 + \alpha_4 \delta_{rot1}^2 + \alpha_5 \delta_{rot2}^2 + \alpha_5 \gamma^2 \delta_{trans}^2)$
3. $\hat{\delta}_{rot2}$	$= \delta_{rot2} - \text{sample}(\alpha_1 \delta_{rot2}^2 + \alpha_2 \delta_{trans}^2)$
4. x'	$= x + \hat{\delta}_{trans} \cos(\theta + \hat{\delta}_{rot1})$
5. y'	$= y + \hat{\delta}_{trans} \sin(\theta + \hat{\delta}_{rot1})$
6. θ'	$= \theta + \hat{\delta}_{rot1} + \hat{\delta}_{rot2}$
7. <i>return</i> x_t	$= (x', y', \theta')^T$

Table 3.1 represents the Pseudo-3D Odometry Sample Motion Model. This model is used for sampling from $p(x_t|u_t, x_{t-1})$ with relative motion information on the horizontal plane. Here γ denotes the pitch angle, δ_{rot1} the initial rotation on the projected plane, δ_{trans} the translation and δ_{rot2} the second rotation. More details can be found in [16].

3.5.3 Curb-Intersection Measurement Model

In the correction step, importance weights of particles get adjusted based on measurements and related measurement models. A measurement model is used to adjust the importance weight of each factor. It is formally defined as a conditional probability distribution $p(z_t|x_t, m)$, where x_t denotes the robot pose, z_t denotes the measurement at t , and m is the map of the environment, according to [16]. In our algorithm, z_t is curb and intersection features, and m is an occupancy grid map of the road boundary. By applying the idea of “synthetic LIDAR”, the curb-intersection feature is converted into a synthetic

laser scan, which permits us to use common LIDAR models for the measurement.

LIDAR-VSA1 (for curb features) adopts the “Likelihood Field Range Finder Model”, considering its computational efficiency and less sensitivity to noise. However, LIDAR-VSA2 (for intersection features) has to adopt the “Beam Range Finder Model” due to its working manner. As mentioned in previous sections, intersection features are represented by a set of synthetic beams with maximum range values, meaning that no curb is met along their virtual light paths. Only through ray tracing in the “Beam Model” can this working manner get properly interpreted.

3.5.4 Practical Considerations

MCL Estimation Frequency. In this chapter, an MCL estimation loop is triggered by the arrival of synthetic measurements. Whenever an assembled synthetic scan from LIDAR-VSA1 or LIDAR-VSA2 is available, a prediction step is performed in a retrospective manner, followed by a correction step with the new incoming measurement. In this sense, the frequency of the synthetic scan is quite important. To control the frequency of LIDAR-VSA1 and LIDAR-VSA2, we can either control the frequency of curb-intersection feature extraction, or control their Assembling Threshold. We find it is always advisable to obtain curb-intersection features when the vehicle moves, and suspend the process when stopping. The Assembling Threshold is determined by trading off the MCL response speed and robustness.

Algorithm Robustness. The robustness of MCL is a key issue. A reasonably high Assembling Threshold will help the algorithm to resist measurement noise. Actually, before curb-intersection is fed into MCL, we adopt the temporal EKF method in [98] to reduce measurement noise. The temporal filter is applied after curb extraction and before the scan assembling operation. In the filter update step, if the Mahalanobis distance between the detected curb and the predicted one exceeds a certain threshold, the newly detected curb will be considered as noise and discarded. This EKF method helps to eliminate minor noise like pedestrians and small cars.

Another strategy that we apply to increase algorithm robustness is the injection of random particles [16]. When the vehicle is locally lost or the measurement is badly corrupted for sometime, the short-term average of particle importance factors will be

remarkably decreased. In this case, a fraction of particles will be generated around the predicted position, and spread according to a uniform distribution within a certain range.

Map-Incorporated Prediction. For vehicle localization on urban roads, one assumption is that vehicles are not likely to drive off-road. This assumption allows us to penalize those erratic particles by decreasing their weight importance. In this way, map information is also incorporated into the prediction step, which makes our localization more robust.

3.6 Experiments

3.6.1 Experimental Setup

Our test bed is a Yamaha G22E golf cart with various sensors, as shown in Figure 3.9. (Note that the sensor configuration is different from the latest one shown in Figure 1.1. While the sensor configurations have been changing during the development of the vehicle, the exact configuration at the time of the experiment is shown.) We use one SICK LMS 291 LIDAR for curb and intersection detection. It is mounted in the front, with a tilted-down angle of 18 degrees. One wheel encoder (Scancon-2RS) and one IMU (3DM-GX3-25) are mounted on the cart to provide necessary odometry information (distance, pitch and yaw). The proposed algorithm is tested online. In the experiment, the golf cart is driven manually on a hilly road at the campus of the National University of Singapore, from point S to G, as shown in Figure 3.10. Several big slopes are involved along the way, with the maximum height difference over 10 meters. The average speed in the test is about 3.5 m/s. The reference road map is an occupancy grid map manually generated from a vector-format road map provided by the Land Transport Authority (LTA) of Singapore and a satellite map. The size of this road map is 200 meter by 240 meter, with grid resolution of 0.1 meter, as shown by Figure 3.11.

3.6.2 Experimental Results

In the test, the golf cart is given a rough initial position at S, and driven for about 430 meters to G. The localization results are shown in Figure 3.10. The blue lines denote the road boundary. The red line marks the localization result of the curb-intersection



Figure 3.9: Yamaha G22E golf cart mounted with various sensors

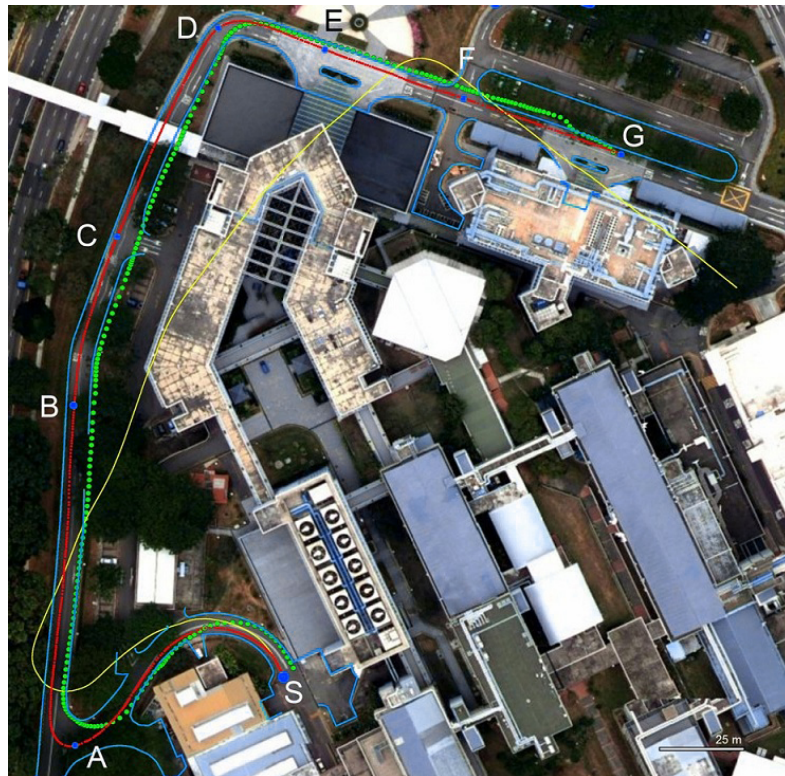


Figure 3.10: Localization results

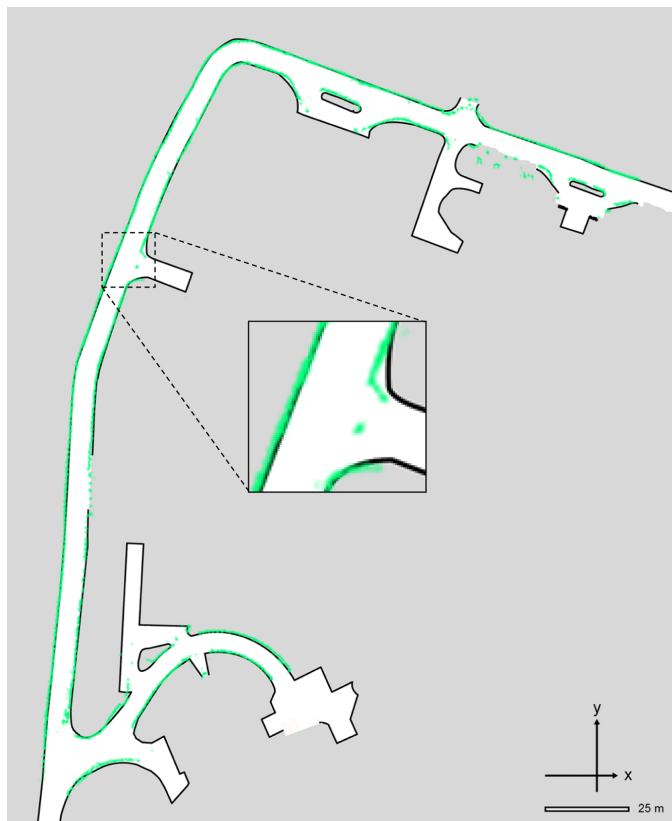


Figure 3.11: Road boundary map and detected curb features

feature-based MCL, and the raw odometry trace is shown by the yellow line. For comparison, we also give the localization result from one state-of-the-art GPS/INS module (Ublox EVK-6R) with the green dotted line. From Figure 3.10, it is apparent that the dead-reckoning odometry drifts a lot after a certain distance. Even when it is fused with GPS, the INS/GPS trajectory tends to fall outside of the road boundary. Because our algorithm incorporates road surface information, it helps to correct the odometry and yield a fairly decent estimation. Figure 3.11 shows the occupancy grid map of the road boundary. The green points represent the curb features detected in the experiment, overlaid on top according to the localization results. Some unexpected points in the figure are measurement noise.

To evaluate the localization result, estimation errors of position and attitude are calculated against ground truth values. We rely on our occupancy grid map to get the ground truth. When the ground truth is needed, vehicle position relative to the road network is measured carefully and marked onto the map image. By counting the pixels in the image, the ground truth can be calculated easily. The vehicle was driven manually to the selected points marked in Figure 3.10 and the errors in location estimate are listed in

Table 3.2: Localization error at several marked points

Marked Points	A	B	C	D	E	F	G
Position Error (m)	0.20	0.55	0.06	0.20	0.32	0.06	0.08
Orientation Error (deg)	< 3						

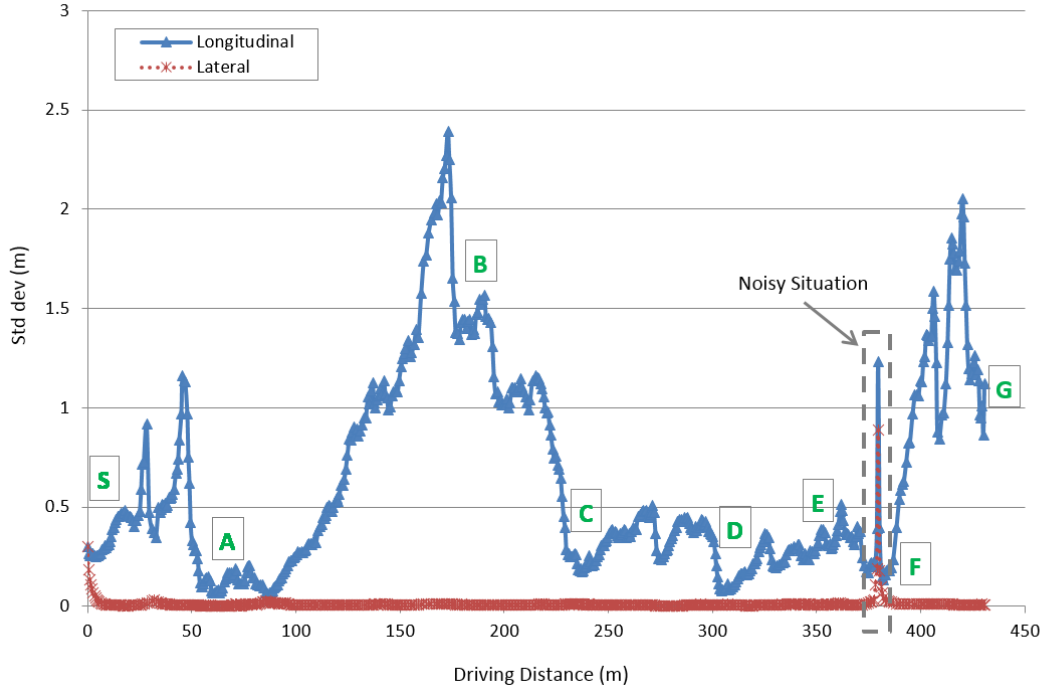


Figure 3.12: Position estimation standard deviation

Table 3.2. It can be seen that position error of our algorithm is usually small, less than 0.6 meter; and the orientation estimation is quite accurate, less than 3 degrees derivation from the ground truth.

From Table 3.2, one can also observe that position errors at some critical points of intersections and turnings (like A, C, D, F) are much smaller than that of the straight road (like B). The phenomenon can be explained from the estimated standard deviation of particles. Figure 3.12 shows “estimation standard deviation” vs “driving distance” in road longitudinal and lateral directions. During the whole test, lateral estimation standard deviation remains small, which means particles are confident about the lateral position. However, the longitudinal standard deviation changes remarkably along the drive, which determines the accuracy of localization.

During the trip from A to B, the longitudinal standard deviation increases first, due to consistency of the road boundary. When the road represents a small curvature, curb features embodying this information will reduce the longitudinal standard deviation.

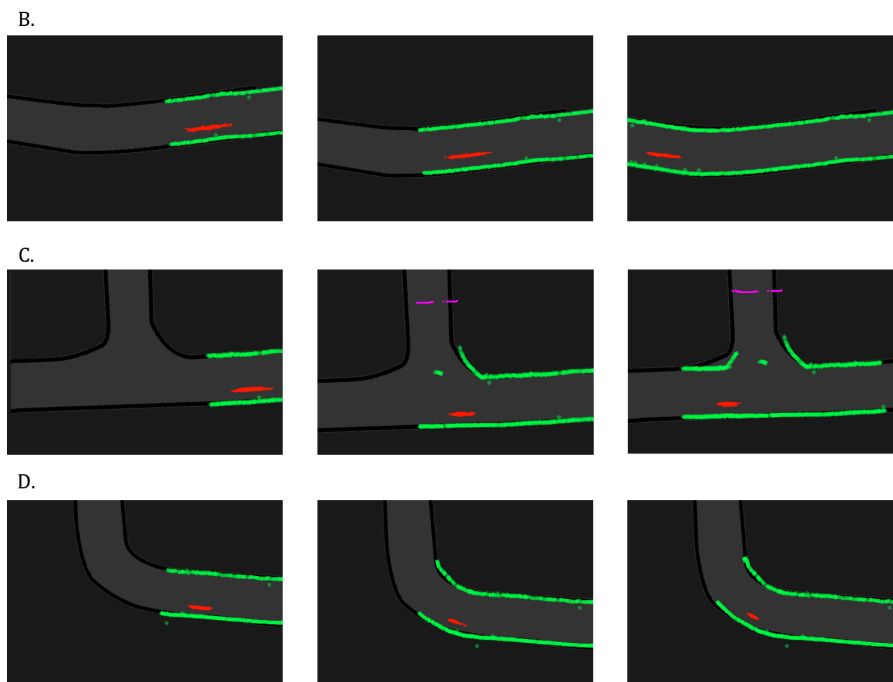


Figure 3.13: Typical particle behaviours near some marked points in Figure 3.10

Thanks to the look-ahead distance of the tilted-down LIDAR, the vehicle will sense this information before it actually arrives there. When the vehicle is approaching the intersections and turnings (like A, C, D, F), the particles are condensed significantly by the detected intersection and the tightly curved curb features. The longitudinal standard deviation at these points is usually less than 0.4 meter. Hereby it can be concluded that, while curb features on straight roads help to estimate the lateral position, the intersection and tightly curved curb features contribute very much to the longitudinal positioning. Figure 3.13 shows the typical particle behaviours around point B, C, D. The red arrays are particles, green lines are the detected curb features, and the purple dots visualize the endpoints of the extracted intersection features.

In the experiment, there is one situation where measurement noise becomes severe, when the vehicle is passing by an intersection at F. As mentioned in Section 3.3, injection of random particles is performed to overcome this “noisy situation”. This operation leads to an increase of the estimation standard deviation, as reflected in Figure 3.12. As long as some new reliable measurements come in, the particles quickly converge, and the localization quickly recovers from the bad situation. Actually, although light measurement noise happens from time to time in the test, the localization is hardly disturbed. The robustness of this algorithm is proved.

Besides the manual drive, we conducted another simple semi-autonomous drive to test our localization algorithm. The vehicle is required to navigate from point S to G by following a predefined route. While the throttle and brake are controlled manually, the steering is controlled by an on-board computer. It turns out that the localization is accurate enough for the vehicle to reach its target smoothly.

3.6.3 Autonomous System Demonstration

As a part of the overall goal of attaining mobility on demand, we conducted an autonomous system demonstration in July 2011, where we had guests request the vehicle to navigate from a pickup location to pre-specified drop-off locations shown in Figure 3.14.

The autonomous vehicle localization was performed purely using curb-only road features. At the pickup and drop-off points where few curb features exist, patches of 2-D occupancy planar maps were used to augment the road network map. Secondary planar LIDAR sensor readings were incorporated to achieve higher accuracy in localization at these critical points. During the course of the demonstration, the autonomous vehicle serviced almost 10 requests from the guests, running over 7 km. The curb-only localization failed at an inclined T-junction 2 times over the whole demo. The reason for failure was determined to be lack of curb features and planar maps at the intersections and T-junctions, which prompted us to include such intersection features, resulting in the localization scheme presented in this chapter. Since then we have covered over 50km in autonomous runs during various demonstrations using the curb-intersection MCL algorithm without failing in any segment of the route. This included situations where the curb detection was hampered briefly by traffic. However such events were detected as no-information cases and recovered from once the sensory occlusion was overcome.

More details of our experiments and demonstrations can be found from Video (4)(5) in Appendix B.

3.7 Summary

This chapter introduces a Monte Carlo Localization algorithm based on the curb-intersection feature, which is extracted through a two-step procedure. A novel idea of “synthetic LI-

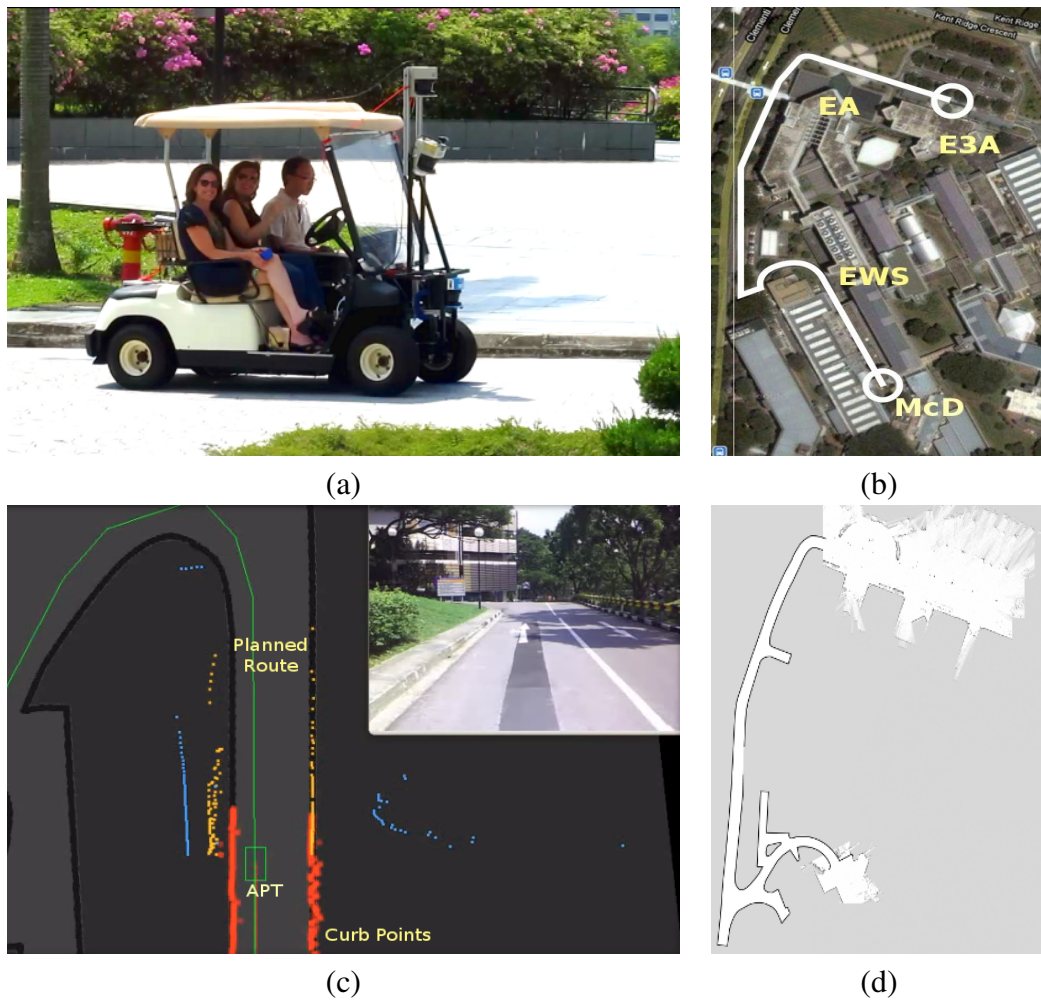


Figure 3.14: Autonomous system demonstration: (a) vehicle in operation, (b) pickup-dropoff points, (c) snapshot of curb localization estimate, (d) curb map augmented by planar patches.

DAR” is applied to represent the curb-intersection features, which enables us to use the standard measurement models of laser sensors. An occupancy grid map for the road boundary is used as prior knowledge. From experiment results, our algorithm proves to be accurate and robust. Although the longitudinal estimation standard deviation may increase on a long straight road, it will not influence much the control of vehicle motion. The look-ahead distance in the feature extraction can help localize vehicles accurately before they reach critical places like junctions and turnings.

Compared to existing approaches that purely use curb features for vehicle localization [43] [44] [45] [46], we introduce an “intersection feature” as a complement, and utilize the combined curb-intersection feature for better localization. Our algorithm achieves accurate estimation results in both the lateral and the longitudinal directions. The contributions of our algorithm also include the way we represent and utilize the curb-intersection features, which are encoded into the format of laser scans with the idea of “synthetic LIDAR”, enabling us to use the standard measurement models of laser sensors for accurate and robust localization.

However, there are two limitations of the proposed localization algorithm: firstly, it is only applicable to the urban roads where curb-intersection features exist; secondly, while the algorithm relies on a road map as a prior, an accurate road map may not always be available, and it is laborious to generate it manually. To counter these limitations, in the following work, we extend the idea of “synthetic LIDA” from curb-intersection features to general vertical surface features, which is applicable to all kinds of urban environments. Chapter 4 introduces the improved localization algorithm.

Chapter 4

Synthetic 2D LIDAR for Precise Localization in 3D Urban Environment

The previous chapter introduced our curb-intersection feature-based localization algorithm. While the algorithm achieves satisfactory results, it is only applicable to road segments which have curbs as the boundaries. To counter its limitations as well as to achieve better localization accuracy, we turn our attention to other salient features in the urban environment, such as building contours, lampposts, etc. The common traits of these artificial objects (or structures) are that they all have vertical surfaces.

This chapter extends our idea of “synthetic LIDAR” from curb-intersection features to general vertical surface features, which are extracted from a 3D rolling window [8]. (In our research, we use the same notion of “synthetic LIDAR” for both the LIDARs built with the curb-intersection features and the one built with the vertical surface features. If not explained explicitly, the notion in this chapter is referred to the latter case.) The basic assumption is that many surfaces in the urban environment are rectilinear in the vertical direction. The interest points are extracted from the rectilinear surface, and then projected on a virtual horizontal plane to form a synthetic LIDAR. The synthetic LIDAR serves as a bridge between the real-world 3D environment and the virtual horizontal 2D plane. With the idea of synthetic LIDAR, algorithms for 2D localization can be easily adapted to the 3D problem. We develop a Monte Carlo Localization algorithm with the notion of synthetic LIDAR, and demonstrate its accuracy and robustness through experiments.

The rest of this chapter is organized as follows. Section 4.1 discusses the idea of pro-

jecting the 3D world to the 2D plane, and gives an overview of the localization system. Section 4.2 addresses the construction of synthetic LIDAR. The localization algorithm is introduced in Section 4.3, and experiment results are presented in Section 4.4. Section 4.5 concludes the chapter.

4.1 Localization on a Virtual Plane

4.1.1 Projecting the 3D world to the 2D plane

Robot localization on a planar surface has been studied for decades and many algorithms have been proposed. The 2D scan-matching algorithm may be the most popular choice due to its accuracy and robustness [123]. However, it cannot be directly applied to vehicles moving in the 3D world. Since outdoor roads can be very hilly at times, laser points from a planar LIDAR may cast onto the road surface, rather than the desired vertical objects, as discussed in [52]. Our previous research in Chapter 3 utilizes a tilted-down LIDAR to extract road boundary features on urban roads, and then uses these features for vehicle localization. While this algorithm achieves satisfactory results, it is only applicable to road segments where curbs exist. Actually, there are many other salient features in the urban environment that can benefit localization. What features to extract, how to extract them, and how to feed them into the localization scheme are questions to be further addressed. It is acknowledged that 3D range data are usually desired to extract features for a robot navigating in the 3D world [67]. In our research, we use a tilted-down LIDAR to generate a 3D point cloud of the environment in a push-broom configuration. Rather than directly applying 3D scan-matching to the raw data [52], we try to extract features from the 3D point cloud, and use the vertical features for localization. The assumption of our method is that the urban environment is rich in vertical surfaces, such as curbs, walls of buildings, and even vertical tree trunks.

The “vertical world” assumption, also called the “2.5D assumption”, is a popular assumption used in many works in the literature. Harrison *et al.* in [124] propose a method to generate high-quality 3D laser range data while the robot is moving. By exploiting the assumption of a vertical world, useful information (e.g., roll and pitch angles) can be inferred. Kohlbrecher *et al.* [125] achieve 2D SLAM and 6-DoF (Degree of Freedom) pose estimation with only a single 2D LIDAR and an IMU. Although not explicitly ex-

plained, the underlying assumption in the work is that the environment contains many vertical surfaces. Weingarten *et al.* in [126] use this assumption to realize fast structured environment reconstruction. In our method, since outdoor environments may have more arbitrary-shaped objects than structural environments, a classification step has to be taken before applying the vertical assumption. In the classification procedure, laser points cast on the vertical surfaces are extracted based on surface normal estimation. When the tilted-down LIDAR sweeps the environment, some vertical surfaces will be swept from bottom to top in consecutive laser scans. If we take a bird's-eye view of this scanning process and project the vertical features onto a virtual horizontal plane, it is exactly the same as a robot with a horizontal LIDAR moving on a 2D surface. From a mathematical point of view, the vertical surface constrains how laser points at different heights should match with each other. With the above intuition, the idea of synthetic LIDAR is proposed. A synthetic LIDAR is a planar 2D LIDAR on the projected virtual plane, where the endpoints of its laser beams are the projected points from the vertical surface in the 3D environment.

The idea of synthetic LIDAR helps to solve the 3D localization problem on a 2D plane. Although a vehicle is moving in the 3D world with 6-DoF, generally speaking, a ground-based vehicle is mostly interested in its 2D pose vector (x, y, yaw) . By projecting the 3D vertical features onto a virtual plane, a 2D occupancy grid map can be built by marking those vertical features. This way, an a-priori map can be obtained using SLAM with the idea of synthetic LIDAR. It should be clarified that our algorithm only applies to an environment with only one traversable level. For cases with more traversable levels such as multistory garages or highway overpasses, some 2.5D or full 3D algorithms may be used [127].

4.1.2 System Overview

The localization system mainly consists of two parts, the 3D perception part to extract key feature points, and the 2D localization part to solve the localization in the horizontal plane. The synthetic LIDAR serves as a bridge to connect the 3D world and the 2D virtual plane, as shown in Figure 4.1.

The system uses an IMU and a wheel encoder to provide 6-DoF odometry information, a 2D tilted-down LIDAR to provide laser scans, and an occupancy grid map as

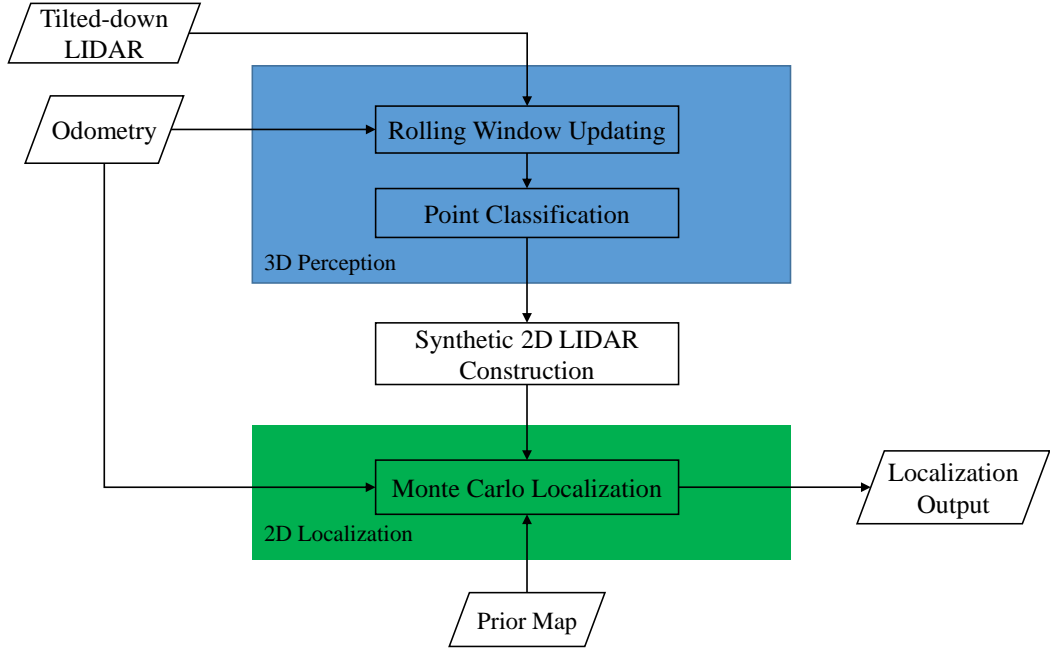


Figure 4.1: Synthetic LIDAR based localization flowchart

prior knowledge for localization. Simple dead-reckoning is used to obtain the odometry information. Assuming the distance measured by a wheel encoder at the n -th time step is r_n , and the rotation is given by a pitch θ and a yaw Ψ , the change in position of the vehicle is given by:

$$\begin{pmatrix} \Delta x_n \\ \Delta y_n \\ \Delta z_n \end{pmatrix} = \begin{pmatrix} r_n - r_{n-1} \end{pmatrix} \begin{pmatrix} \cos \theta_n \cos \Psi_n \\ \cos \theta_n \sin \Psi_n \\ -\sin \theta_n \end{pmatrix} \quad (4.1)$$

The 3D perception process assumes that the odometry system is accurate enough in a short time period, and accumulates the 2D laser scans for 3D range data. A classification procedure is then applied to extract interest points from the accumulated data. The extracted laser points are then projected onto a virtual horizontal plane (by ignoring their z values), and a synthetic 2D LIDAR is constructed. The 2D localization fuses odometry information from odometry and measurements from the synthetic 2D LIDAR in a Monte Carlo Localization scheme. With a prior map of vertical features generated beforehand, localization in the 2D horizontal plane is achieved.

4.2 3D perception

One of the requirements of the synthetic LIDAR is the adaptability to different types of environment in urban scenarios. We achieve this by properly extracting interest points from a reconstructed environment model, before the synthetic LIDAR is built.

To be able to recognize features that are perpendicular to the ground, an accurate model of the world is necessary. There are numerous ways that allow building an accurate environmental model, which include nodding LIDAR and Velodyne. As introduced in Section 4.1, a single tilted-down planar LIDAR enables the reconstruction of the environment accurately by sweeping across the ground surface. This is an attractive solution since it is low-cost and only requires rigid mounting of the sensor. It also allows efficient computation for feature extraction, as discussed later.

4.2.1 3D rolling window

A 3D rolling window is used to accumulate different scans recorded across a short distance. The size of the window is flexible and the rolling window forms a local snapshot of the 3D environment. It moves together with the vehicle, where new incoming scans are added into the window, and the old samples get discarded. Let w denote the window width, i denote the 2D scan index, p_i the points in the i th scan, n the latest scan index, and β the control distance. P_n is the 3D point cloud updated by the newest scan n , which is accumulated according to

$$P_n = \bigcup_{k=n-\lfloor w/\beta \rfloor} \{p_k, \dots, p_n\} \quad n > \lfloor w/\beta \rfloor \quad (4.2)$$

As shown in Figure 4.2, a new scan is only inserted when sufficient distance β is reached. This prevents the rolling window from getting redundant points at the same place. There are many ways to control the processing of the collected 3D data. The way used in our work is to process the rolling window when the vehicle traverses a certain distance interval w .

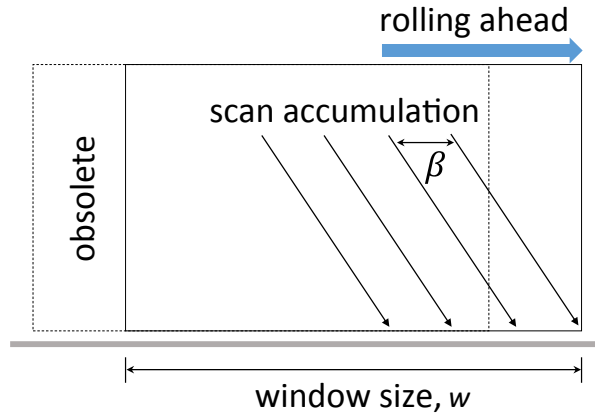


Figure 4.2: 3D rolling window

4.2.2 Point Classification

To extract features that are perpendicular to the ground, estimation of surface normal is used. While many methods exist [128], we use the normal estimation method proposed by [129]. It is based on first-order 3D plane fitting, where the normal of each point in the space is approximated by performing least-square plane fitting to a point's local neighborhood P^K [130]. The plane is represented by a point x , its normal vector \vec{n} and distance d_i from a point $p_i \in P^K$, where d_i is defined as:

$$d_i = (p_i - x) \cdot \vec{n} \quad (4.3)$$

By taking $x = \bar{p} = \frac{1}{k} \sum_{i=1}^k p_i$ as the centroid of p_k , the values of \vec{n} can be computed in a least-square sense such that $d_i = 0$. The solution for \vec{n} is given by computing the eigenvalue and eigenvector of the following covariance matrix $C \in \mathbb{R}^{3 \times 3}$ of P^K [131]:

$$C = \frac{1}{k} \sum_{i=1}^k (p_i - \bar{p}) \cdot (p_i - \bar{p})^T, C \cdot \vec{v}_j = \lambda_j \cdot \vec{v}_j, j \in \{0, 1, 2\} \quad (4.4)$$

where k is the number of points in the local neighborhood, \bar{p} as the centroid of the neighbors, λ_j is the j^{th} eigenvalue with \vec{v}_j as the j^{th} eigenvector. The principal components of P^K corresponds to the eigenvectors \vec{v}_j . Hence, the approximation of \vec{n} can be found from the smallest eigenvalue λ_0 . Once the normal vector \vec{n} is found, the vertical points can then be obtained by simply taking the threshold of \vec{n} along the z axis, e.g. 0.5. This can vary depending on how noisy the sensor data are.

To find the local neighborhood points efficiently, KD-tree [132] is built from all the points obtained from the rolling window and a fixed radius search is performed at each point. Although the surface normal can be calculated as a whole, performing normal calculation at each point in the rolling window can be very expensive. To further reduce the computation complexity, two successive rolling windows are maintained. The idea is illustrated by the following equation:

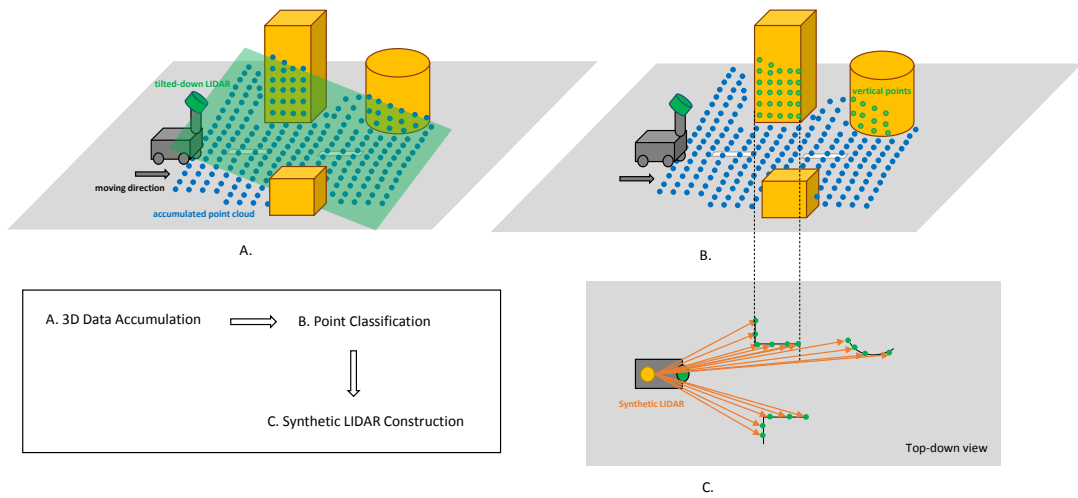
$$P_{n+1}^{\phi} = P_n^{\phi} \cup \Phi(P_{n+1} \setminus P_n) \quad (4.5)$$

where Φ can be any points classification function, P^{ϕ} consists of the processed points and P contains the raw points. This way, surface normal calculation is only required for the much smaller rolling window $P_{n+1} \setminus P_n$. In other words, this ensures that classification will only be performed on the newly accumulated point cloud and the processed points from the previous instance can be reused.

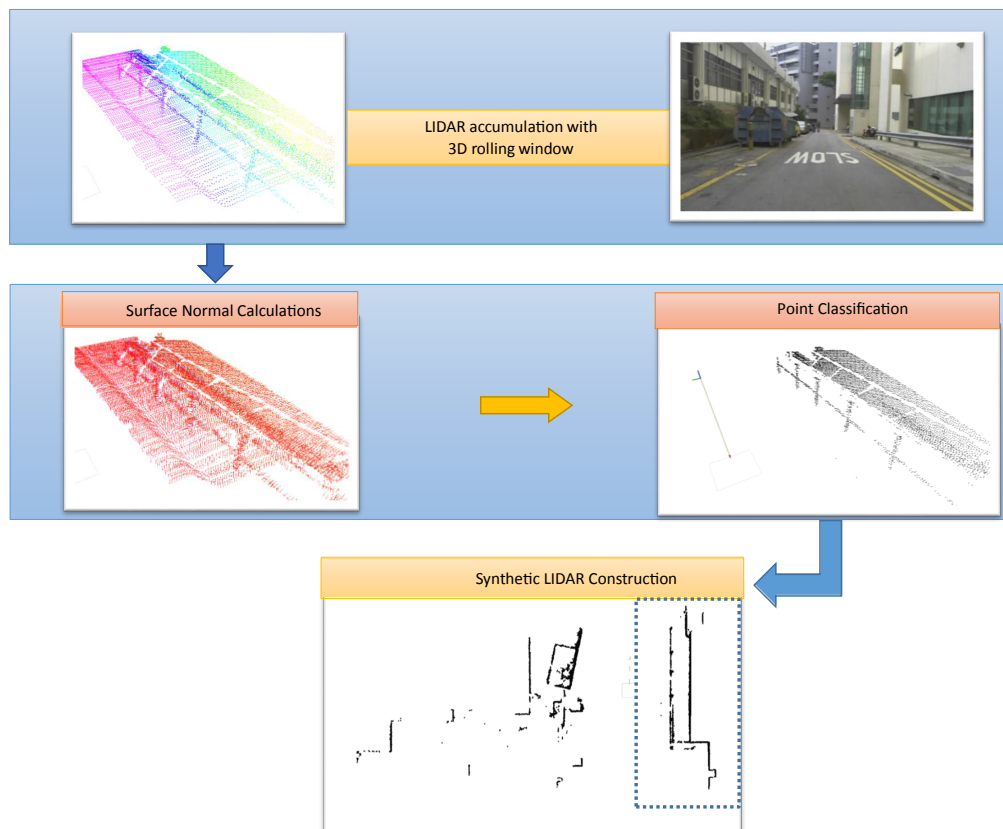
4.2.3 Synthetic LIDAR construction

The classified points consist of the collection of interest points in 3D. For the construction of synthetic LIDAR, the interest points in 3D are projected onto the virtual horizontal plane ($z=0$). It can be seen that this synthetic LIDAR has a very special trait: the ability to “see through” the obstacles. This is possible since interpretation of points is done in 3D. The construction of synthetic LIDAR is completed by placing the virtual sensor at the base of the vehicle and performing transformation of all the interest points from odometry to the vehicle’s base.

The construction of synthetic LIDAR can be illustrated by a cartoon illustration, as shown by Figure 4.3(a). In the illustration, the tilted-down LIDAR is shown by a green cylinder mounted on a vehicle, which sweep the environment as the vehicle moves forward. A point cloud is accumulated with the idea of 3D rolling window, where each point is denoted as a blue dot. In the point classification process, we extract points which are cast on the vertical surfaces of the environment, and visualize them as the green dots. In the process of LIDAR construction, a synthetic LIDAR is constructed with its origin placed at the base of the vehicle and its laser beams ending at the projections of the extracted “vertical points”.



(a) a cartoon illustration



(b) a real-world example

Figure 4.3: Construction of synthetic LIDAR

A real-world example is also provided to visualize the construction process, as shown by Figure 4.3(b). In this figure, we also visualize the intermediate results of surface normal calculation, where the normal at each point is represented by a tiny red arrow. In our application, the operations of the point cloud are carried with the Point Cloud Library (PCL) [133], which provides many useful functions for 3D perception.

4.3 Online Localization

4.3.1 MCL Localization

Our algorithm adopts the Monte Carlo Localization (MCL) scheme to estimate the vehicle pose. MCL is a probabilistic localization method based on Bayes Theorem and the Monte Carlo idea [16]. The core of MCL is a particle filter, where the belief of vehicle position is maintained by a set of particles. MCL mainly consists of three steps, prediction, correction, and the resampling. For the motion model which is required for the prediction step, Pseudo-3D odometry motion model from Chapter 3 is used. The choice of measurement model is discussed in the following.

4.3.2 Synthetic LIDAR Measurement Model

To incorporate the measurement into localization, a measurement model is needed for the synthetic LIDAR. The “Likelihood Field Range Finder Model” is adopted for the synthetic LIDAR. Our considerations are as follows. Since the endpoints of virtual beams are the projection of interest points from vertical surfaces, it is possible that different points from different vertical surfaces may have the same angle. In other words, there exist two laser beams with the same angle while having two different range values. For this reason, synthetic 2D LIDAR is a peculiar LIDAR that only detects vertical surfaces, and can also see through these surfaces. In light of this, the likelihood model which only requires the endpoints of laser beams is well suited for the synthetic LIDAR.



Figure 4.4: Vehicle testbed

4.4 Experiments

4.4.1 Experiment Setup

Our test bed is the Yamaha G22E golf cart used in Chapter 3, but with a slightly different sensor configuration. The hardware configuration is shown in Figure 4.4. The tilted-down LIDAR mounted in the upper-front is a SICK LMS-291 LIDAR for localization. A 4-layer LIDAR, SICK LD-MRS400001 is mounted at waist level for obstacle detection. Both rear wheels of the golf cart are mounted with encoders that provide an estimate of the distance traveled. An Inertial Measurement Unit (IMU) MicroStrain 3DM-GX3-25 is mounted at the center of the rear axle to provide orientation information of the vehicle. The localization algorithm is tested on the Engineering Campus of the National University of Singapore, where the road is up-and-down and many high buildings exist off the road.

A prior map is first generated with Graph-SLAM techniques by using the synthetic LIDAR as the input. To perform pose optimization, Fast Laser Interest Region Transform (FLIRT) [134] is used as a front end to detect loop closure. Then, the fully optimized pose is recovered using the optimization library from [135]. To evaluate the quality of the recovered map built from synthetic LIDAR, the map is projected onto a satellite map, as shown in Figure 4.5. The map shows consistency with good correlation with the satellite map, with an area of about $550\text{ m} \times 487\text{ m}$. Although there are discrepancies towards



Figure 4.5: Mapping of the NUS Engineering Campus

the left side of the map due to uniform longitudinal features along the road, the overall topology is maintained. This shows that the map can be used for accurate localization.

4.4.2 Experiment Results

The synthetic LIDAR is able to perform at a rate of 50 Hz output on a laptop with a Core i7 processor, showing that the synthetic LIDAR can be used to perform a real time localization. The localization results are shown in Figure 4.6. Judging from the prior map, the localization result from our algorithm always aligns with our driving path where a parallel line with the road boundary is clearly shown in the long stretch of road. Since our algorithm does not rely on GPS, our estimation still performs well near areas crowded by tall buildings. Note that in the experiment, a rough initial position is given and hence localization is mostly concerned with pose tracking. However, the system is able to cope with small kidnapping problems, e.g. brief data error from the LIDAR, since the odometry system is still able to provide information. Should a large kidnapping occur, e.g. the vehicle was moved in between placed without turning on the localization module, a rough initial position may be provided to speed up the convergence rate.

Figure 4.7 shows “localization standard deviation” vs “driving distance”. The standard deviation of angle is generally less than 1° , as shown in Figure 4.7(a). Figure 4.7(b)



Figure 4.6: Localization results during a manual drive

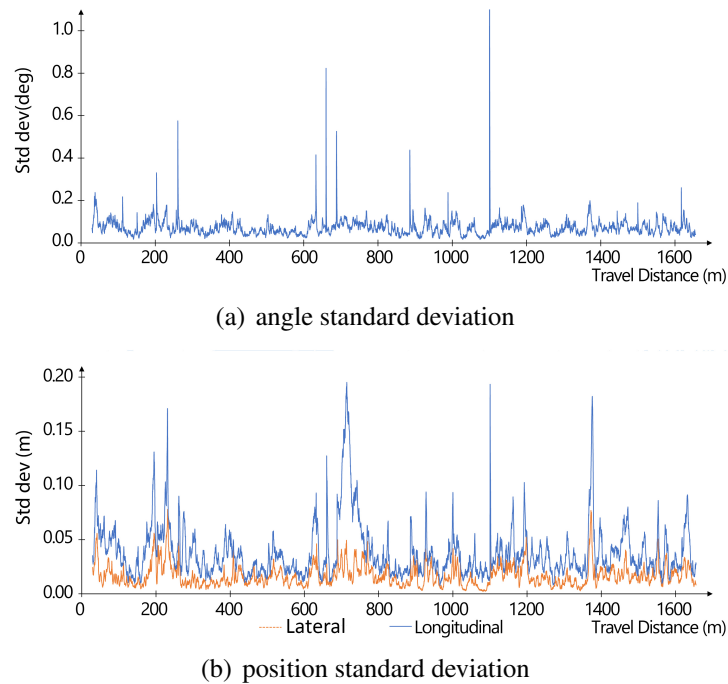


Figure 4.7: Localization standard deviations

shows “position standard deviation” vs “driving distance” in longitudinal and lateral directions relative to the vehicle. It is shown that during the whole test, standard deviations in both directions remain small. The worst estimations occur longitudinally, at a value of about 0.2 m. This suggests the localization algorithm has high confidence about its pose estimation. At the same time, it is also seen that the lateral standard deviation is generally smaller than the longitudinal one. This is inline with the fact that in an urban road environment, features in the lateral direction are much richer than those from the longitudinal direction, as discussed in our previous work in Chapter 3. By comparing Figure 4.7(b) to Figure 3.12, it can be easily observed that the localization performance in the longitudinal direction is much better than that in the previous work. The improvement can be attributed to the usefulness of vertical surface features other than the curbs.

This proposed method has since been used to perform autonomous navigation similar to the previous work. To show that the localization results are consistent, two autonomous runs are performed as shown in Figure 4.8. The golf cart is given a path to follow with the direction from A to E. To validate the precision of the localization, 5 checkpoints are selected where per-pixel absolute difference between 2 grayscale images captured from each autonomous run is performed. The larger the grayscale difference is, the darker the output image. To ensure the same lighting condition, the two autonomous

runs were performed consecutively on the same day. 5 smaller images from Figure 4.8 are the results of the visual validation. The bright images give strong evidences that the localization system is able to provide precise position repeatedly. Do note that the dark spots present at images B, C and E are the natural results from moving objects. This shows that precise navigation can be achieved with only a single 2D LIDAR.

More details of our experiments and related applications can be found from Video (6)(7) in Appendix B.

4.5 Summary

This chapter proposes the use of synthetic LIDAR constructed from vertical surface features for precise localization in a 3D environment. Vehicle position estimation is conducted in a Monte Carlo Localization scheme, based on synthetic LIDAR measurements and odometry information. We demonstrate the accuracy and robustness of our localization algorithm in a driving test. Since this method utilizes the general vertical surface features rather than only the curb-intersection features, it is able to achieve better localization performance than the algorithm introduced in Chapter 3.

The contributions of our localization algorithm presented in this chapter are two-fold: firstly, compared to the state-of-art algorithm using 3D LIDAR [15], our method achieves equivalent performance, however, with the much reduced sensing ability of a 2D LIDAR; secondly, similar to the algorithm in Chapter 3, the idea of “synthetic LIDAR” enables us to use the standard laser model to solve the localization problem on the projected 2D plane, both efficiently and precisely.

To sum up, Chapter 3 and Chapter 4 introduce our localization methods with only a tilted-down LIDAR and odometry information. By utilizing the typical features in the urban road environment, the proposed localization algorithms achieve good accuracy as well as robustness. In our future work, we are looking forward to developing a generic framework that incorporates multiple sensory modalities for localization, in order to further improve the accuracy and robustness.

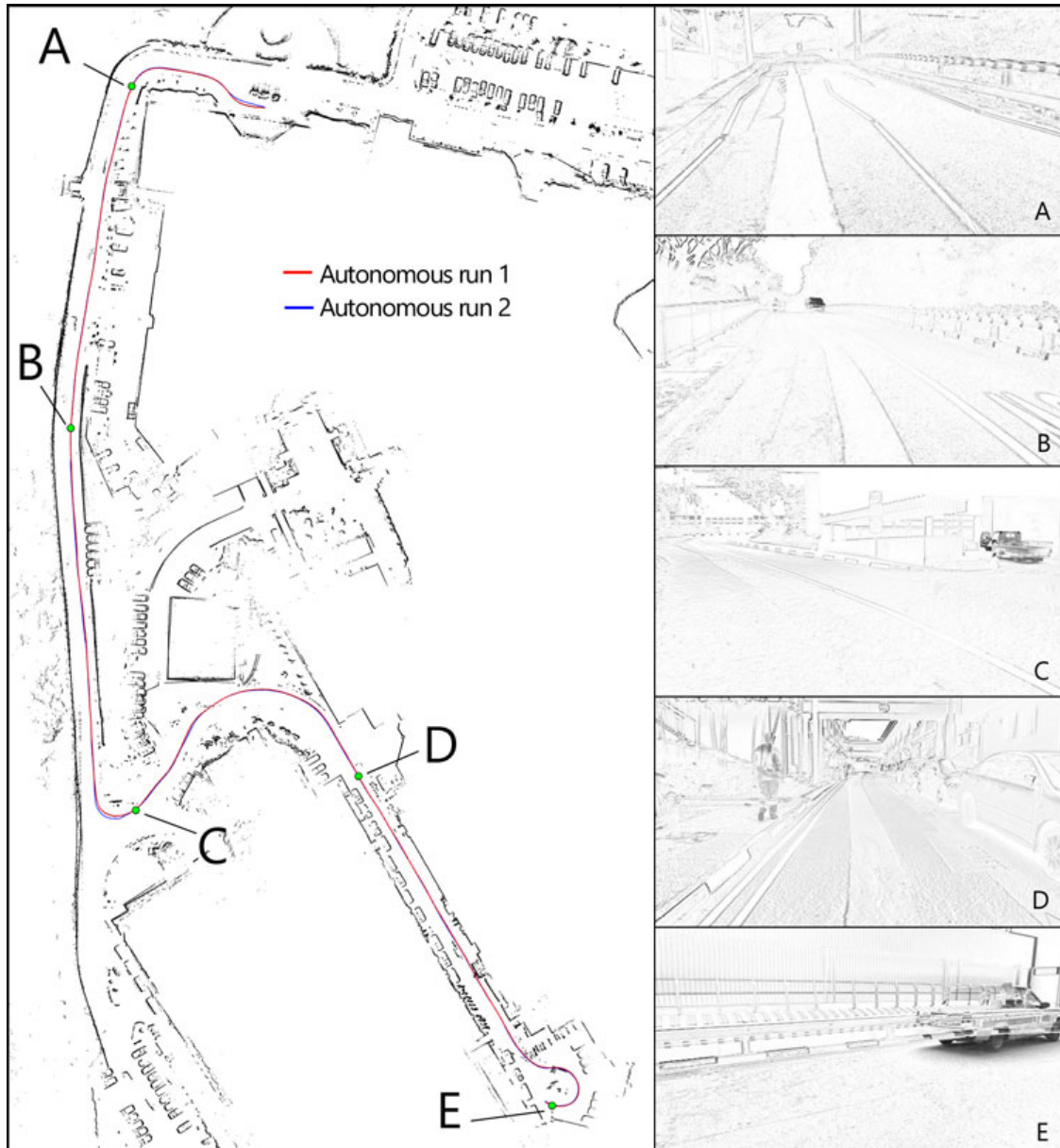


Figure 4.8: Autonomous navigation with synthetic LIDAR. Images on the right from top to bottom correspond to visual validation of localization repeatability from checkpoint A to E.

Chapter 5

A General Framework for Road Marking Detection and Analysis

Road detection is the problem of detecting and locating drivable roads, which is a basic requirement for vehicle autonomous navigation. As reviewed in Chapter 2, two major streams of research are found around this topic, i.e. road marking detection, and road surface-boundary detection. This chapter deals with the problem of road marking detection. The topic of road surface-boundary detection is addressed in Chapter 6.

5.1 Introduction

Road marking detection has been a popular research topic in the context of Autonomous Driver Assistance Systems (ADAS). Road markings are paintings on road surface to provide traffic guidance information for vehicles and pedestrians. Common road markings include lane markings, arrows, zebra-crossings, words, etc. Researchers aim to detect and locate these road markings, and utilize the results to guide vehicle autonomous navigation. This chapter introduces a general framework for road marking detection and analysis using vision, which is able to support various marking types.

Lane markings are the dominant markings on the road surface, and there has been a lot of research to detect lane markings, as reviewed in Section 2.3.2. The detection of other types of markings is also studied, such as arrow markings, pedestrian crossings, and so on [68] [93] [95]. However, existing approaches are all case-specific, and there is a lack of a general framework which supports all the various types of markings. Our

research in this chapter targets meeting this demand.

Our basic idea is to extract all the marking contours indiscriminately from image processing, and then send them into different modules for dedicated classification and analysis. Each type of marking will have its own dedicated classifier, which extracts its marking contours of interest, and filters out the rest. The recognized marking will be further fed into an analysis process to extract its guidance information. In some cases, road markings come as a group, such as zebra-crossings and words. For markings of these types, two rounds of classifications will be involved. Firstly, their contour components will be recognized at an individual level. Then the recognized components are clustered as groups, and a second round of classification will be carried out at the group level. After the marking groups are recognized, traffic guidance information will be extracted from them in the subsequent analysis procedure. Unlike the previous work, which only deals with certain specific types of markings, our proposed method is general enough to support a variety of markings.

The rest of this chapter is organized as follows. Section 5.2 introduces the overview of the proposed framework. Section 5.3 discusses the image processing to extract marking contours. Section 5.4 presents the detection and analysis modules for four marking types. Experimental results are shown in Section 5.5. Finally, Section 5.6 concludes the chapter.

5.2 Framework Overview

The framework of marking detection and analysis is shown in Figure 5.1. Its input is the camera image, and the output is various marking information. This framework can be roughly divided into two parts, the image processing part for marking contour extraction, and the classification-analysis part for contour recognition and analysis.

In the image processing part, input image is first transformed through an Inverse Perspective Mapping (IPM) process to get a bird's-eye view of the road surface. Image binarization is then applied to the transformed image to extract the foreground pixels. The binarized image is further segmented into different parts with their contours extracted. While each contour represents one single marking candidate, their contours are sent to the classification-analysis part for further processing.

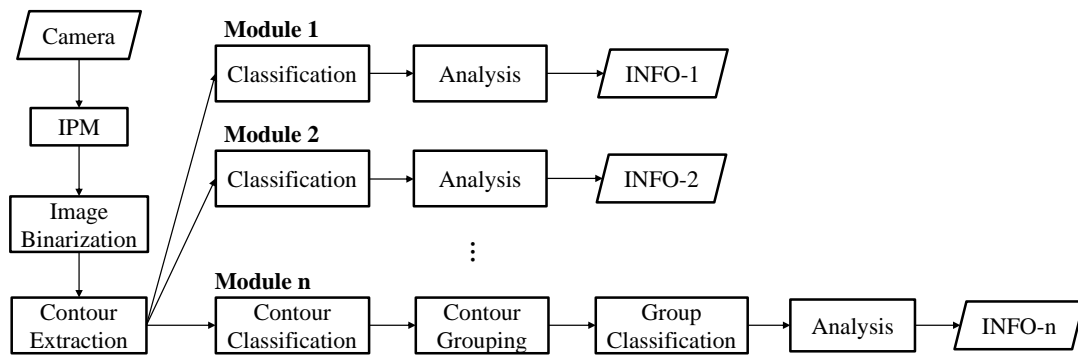


Figure 5.1: A general framework for road marking detection and analysis

The classification-analysis part is composed of several parallel classification-analysis modules. These modules are independent from each other, with each module dedicated to its corresponding type of marking. In one module's classification step, contours are classified into $(c + 1)$ classes, where c is the class number of markings in this type, and the additional "1" represents other irrelevant types of markings and noise. For example, there are c types of arrow markings on urban roads. In the "Arrow" module, its classifier will treat markings of other types as noise, and try to identify these c classes of arrow markings. The identified markings are then analyzed to extract useful guidance information. In some cases, road markings come as a group, such as the markings of zebra-crossings. In such cases, one individual marking component carries no meaningful information. It is desired to cluster the recognized contour components together, and recognize them as an entity.

5.3 Image Processing

5.3.1 Inverse Perspective Mapping (IPM)

Inverse Perspective Mapping (IPM) is an image transformation method to get a bird's-eye view of the road surface [136]. When a forward-looking camera captures an image, shapes of road markings are usually distorted due to perspective projection. An IPM process can be applied to remove this distortion, and recover the original shapes for classification purposes. Figure 5.2 illustrates the basic idea of IPM. For a fixed-mount on-board camera, its pose in the vehicle coordinate frame is usually known, and hence

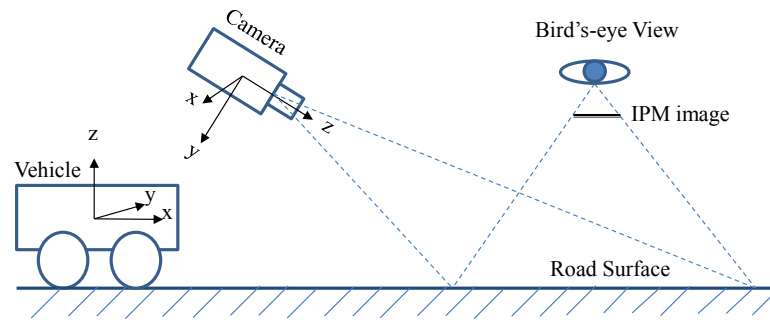


Figure 5.2: Inverse Perspective Mapping

its pose relative to the road surface. Together with its intrinsic parameters from a prior calibration process, the perspective transformation matrix from road surface to camera image can be calculated. An inverse operation of this perspective matrix will restore the original road surface, and represent it as an IPM image.

An example of the complete image processing flow is shown by Figure 5.3. Figure 5.3(a) shows the original image, and Figure 5.3(b) illustrates the image after IPM transformation. We select our Region of Interest (ROI) to be the lower part of the original image, as shown by the blue quadrilateral. During the transformation, besides the IPM ROI, we can also define the pixel-to-meter ratio between the transformed image and real road area. This ratio bridges the position of an object in the IPM image, and its position in the real world.

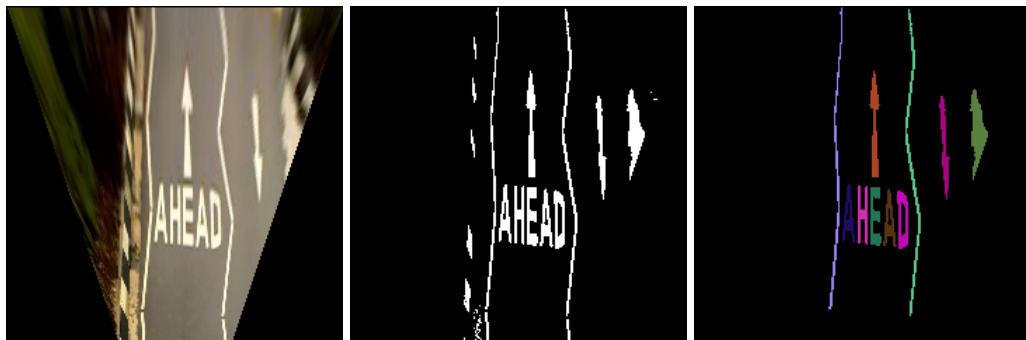
5.3.2 Image Binarization

After the IPM process, the transformed image is converted into an 8-bit grayscale image for the operation of binarization. We combine the global thresholding and local adaptive thresholding methods to generate the binary image. Since road markings usually have higher brightness than the background surface, a global thresholding step with an appropriate threshold value will naturally distinguish the foreground markings and the background surface. We select a fixed threshold value for this global thresholding step.

To guarantee that the global thresholding always works well, we implement one camera exposure control method [137] to maintain a stable grayscale histogram for the IPM image. While the global thresholding distinguishes the foreground and background pixels in a global manner, some details of the IPM image may be lost due to unbalanced exposure. We use the adaptive thresholding method to recover these local details. In our



(a) original image



(b) IPM transformation

(c) image binarization

(d) contour extraction

Figure 5.3: An example of image processing

implementation, the IPM image is independently processed by the two types of thresholding methods, and the resulted images are then fused together in one bitwise “AND” operation. Figure 5.3(c) shows the resulting image after binarization.

5.3.3 Contour Extraction

After the image binarization process, the foreground pixels are segmented into different components. Since the contour of a component carries all its geometric information, we use the contours to represent the extracted components. Our operations of recognition and analysis will be carried out with the geometric information of the contours. Before sending the contours for further processing, a prefiltering step is applied to remove the tiny pieces of contours from background noise. Figure 5.3(d) shows the extracted contours by visualizing them in different colors.

5.4 Contour Classification and Analysis

The image processing procedure extracts a set of contours, which comes from various road markings as well as background noise. This set of contours needs to be further classified and analyzed to extract useful traffic guidance information. Our framework uses independent classification-analysis modules to deal with different marking types. In each classification-analysis module, a classifier is trained to recognize its dedicated marking type. Some specific analysis will then be carried out for this type of marking.

A Support Vector Machine (SVM) [138] is used for the classification purpose, with each classifier trained independently for each single type of marking. Geometric features of each contour are extracted to compose its feature vector, as summarized in Table 5.1. The selected features are rotation-invariant, and carry adequate geometric information for the classification operation.

One interesting issue of contour classification arises when certain types of markings appear as a group. These types of markings include zebra-crossings, road surface words, etc. After the classification of contour components at the individual level, recognized components are then clustered into groups, where a second-round classification will be carried out to recognize the marking groups.

In this section, we study four types of common markings, and their classification-

Table 5.1: Geometric features of marking contours

Feature Name	Dims.	Description
Hu Moments	7	<i>The seven Hu Moments of a contour, which are scaling and rotation invariant.</i>
Weight	1	<i>The length in pixels of a contour.</i>
Bounding Rectangle	2	<i>The minimum-area bounding rectangle of a contour is calculated, and its two side lengths are selected as features.</i>
Approximate Polygon	1	<i>A polygon approximation is applied to a contour, and the vertex number of the polygon is selected as a feature.</i>

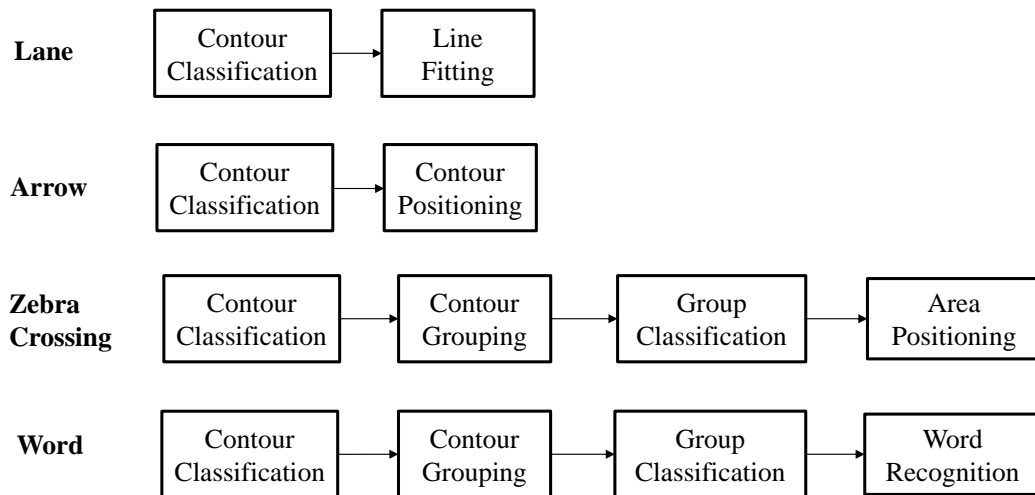


Figure 5.4: Four marking modules

analysis modules are shown in Figure 5.4. While the classifications of different markings use the same method, analyses on them are quite different: for a lane marking, our interest is to extract its mathematical representation as a line; for an arrow, our interest is to determine its type, and calculate its pose relative to the vehicle; for a zebra-crossing, we want to recognize its existence and determine its position and covering area; for a word on the road surface, we want to infer its meaning given a learned dictionary.

5.4.1 Lane Module

Lane markings are used to denote road lanes, and are the most dominant markings on the road surface. A binary classifier is trained for lane detection, to determine whether one marking is a lane or not. To get a line representation of the recognized marking, we

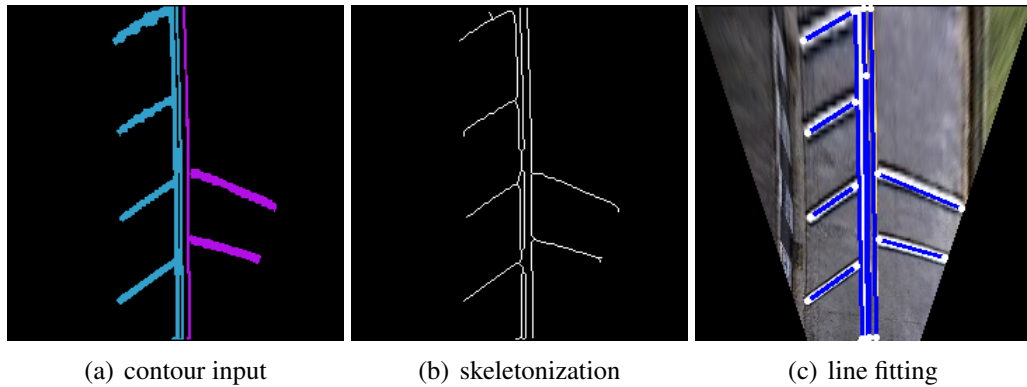


Figure 5.5: Lane processing

first perform a thinning operation [139] to get its skeleton with single-pixel thickness. The Random Sample Consensus (RANSAC) [140] method is then applied to fit a line on the skeleton pixels.

It is possible that multiple lane markings are connected to each other, and extracted as a single marking. To deal with this situation, an iterative process of RANSAC is performed for multiple-line fitting. In the iteration, after a line is fitted by RANSAC, its inliers are removed from the skeleton pixel sets, and the remaining pixels are used for the next iteration step. The iteration process is terminated when no more “good” lines can be fitted. Here the criteria for “good” are that a line should have enough supporting pixels, and its length should exceed a certain threshold. Figure 5.5 shows an example of lane contour processing, where Figure 5.5(a) shows the contour input, Figure 5.5(b) visualizes the skeletonization for the recognized lane contours, and Figure 5.5(c) illustrates the RANSAC line fitting result on the IPM image. This iterative RANSAC method works well to extract multiple straight lanes when they are connected to each other. As for curved lanes, they will be approximated by several connected straight lanes.

5.4.2 Arrow Module

Arrow markings usually appear nearby T-junctions or turnings of the road network, and provide direction guidance for drivers. While the appearances of arrow markings may vary in different countries, we study 7 types of common markings from our local road, as shown in Figure 5.6. To recognize these arrows, an 8-class classifier is trained. The additional class denotes other irrelevant markings and noise.

For the recognized arrows, we want to calculate their poses (x , y , θ) relative

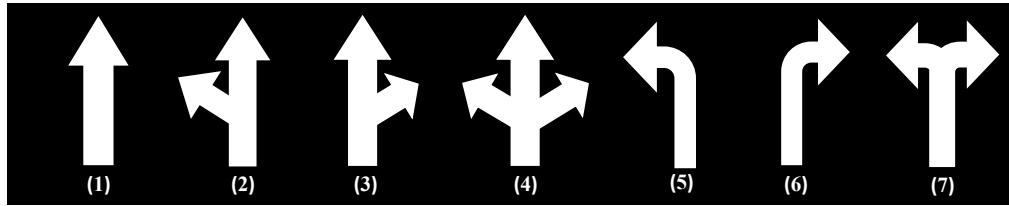


Figure 5.6: Common arrow types

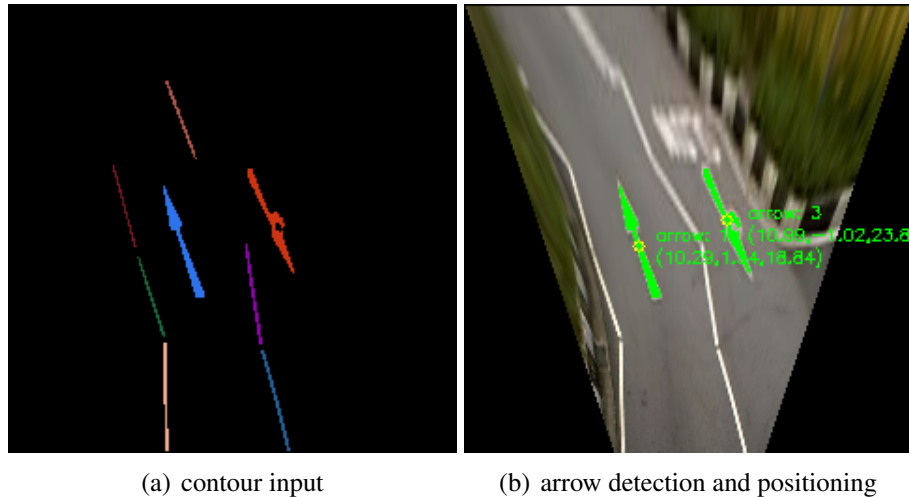


Figure 5.7: Arrow processing

to our vehicle. The arrow position of (x, y) is represented by its contour's centroid position, which is first calculated in the IPM image with “pixel” as its unit, and then transformed into the real world coordinates using the IPM pixel-to-meter ratio mentioned in the former section. Arrow angle θ is denoted as the angle of its contour's principal axis [94]. While there are 2 principal axes for a 2D contour, the one corresponding to a smaller angle is chosen, considering that an arrow is usually parallel to a vehicle's driving direction. Figure 5.7 shows an example of arrow processing. In Figure 5.7(b), two arrows are recognized and colored in green with their poses in the vehicle coordinates.

5.4.3 Zebra-Crossing Module

A zebra-crossing is an area for pedestrians to go across the road. The recognition of a zebra-crossing area can alert drivers to possible pedestrians. Its most distinguishing feature is the alternating dark and light stripes. In the zebra-crossing module, a binary classifier is trained to recognize the light stripes. Then the extracted strip contours are clustered as a group. A second round of classification is then carried out at the group level to recognize possible zebra-crossings.

In the clustering/grouping step, multiple criteria are used to calculate the distance measurement between two contours c_1 and c_2 , denoted as $Dist(c_1, c_2)$. The first criterion is about the parallelism between two contours. While the contour of a zebra-crossing strip can be approximated by a rectangle, $Angle_dist(c_1, c_2)$ denotes the angle between the long sides of their rectangles. The second criterion deals with the collinearity of the rectangle short sides, which is reflected by the measurement $Col_dist(c_1, c_2)$. $Col_dist(c_1, c_2)$ is calculated as the maximum distance from points in one short side to the line of the other. The third criterion is about the distance between the centroids of two contours. For two neighboring zebra-crossing stripes, the centroid distances of their contours should be around 2 times the rectangle width. We use $Centr_dist(c_1, c_2)$ to present this knowledge, which is calculated as the centroid distance minus 2 times the rectangle width. The synthetic distance $Dist(c_1, c_2)$ is defined as:

$$Dist(c_1, c_2) = a_1 Angle_dist(c_1, c_2) + a_2 Col_dist(c_1, c_2) + a_3 Centr_dist(c_1, c_2) ,$$

where a_1, a_2, a_3 are weighting parameters. When $Dist(c_1, c_2)$ is less than a fixed threshold T , two contours are clustered into one group. T can be learned through a supervised learning process.

After contour grouping, clusters of contours will be classified at a group level using a second classifier. The second classifier is a binary classifier to determine whether one cluster of contours is a crossing, which can be trained based on the group features. In our application, the group feature is simply the contour number in one cluster, as a zebra-crossing usually has multiple light stripes. The covering area of one recognized zebra-crossing is calculated by finding a minimum-area enclosing rectangle for all its stripes. Figure 5.8 shows an example of zebra-crossing processing.

It should be mentioned that our zebra-crossing detection algorithm works well even in the presence of partial occlusion. By choosing a relatively small value for parameter a_3 when calculating $Dist(c_1, c_2)$, two adjacent crossing contours can be clustered together even when some crossing contours are occluded in between. As long as enough contours are grouped in one contour cluster, this cluster will be recognized as a zebra-crossing.

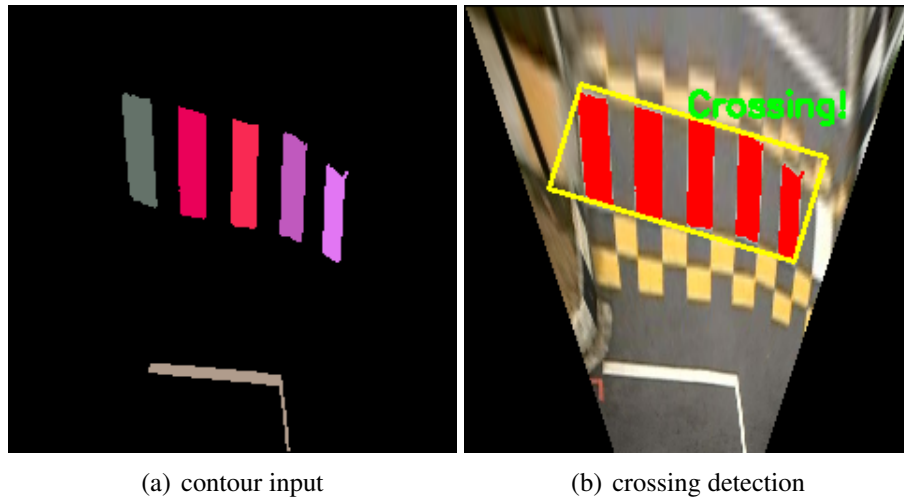


Figure 5.8: Zebra-crossing processing

5.4.4 Word Module

In an urban road network, another important type of markings is the words. These words are usually used to give speed advice for drivers, to indicate a speed bump ahead, etc. The letters of these words are usually in uppercase, with uniform height and width. In the word module, we first apply a binary classifier to extract the letter contours, then group them, and finally recognize them as a word at the group level.

In the first step, a binary classifier is trained to classify the contours into two types, the letter contour and the non-letter contour. While letter contours usually have uniform size and some similar geometric features, non-letter contours are either too big or too small, with irregular shapes. This observation is utilized by the classifier to distinguish letter and non-letter contours. It should be clarified that we do not perform character recognition at this step, but leave it to the end of the word module. Similar to the processing of zebra-crossing, our second step is contour grouping. While some noise may remain from the classification step, the grouping process can discard most of the remaining non-letter contours. After contour grouping, a simple classification will be performed on the extracted groups. Only the groups with enough letter contours will be treated as possible words, and fed into the word recognition process.

Tesseract [141] is one popular open source Optical Character Recognition (OCR) engine, and we use it for the word recognition purpose. In our application of road word recognition, it is found that Tesseract achieves much better performance recognizing a single letter than a word. For this reason, we use it to recognize individual letters inside

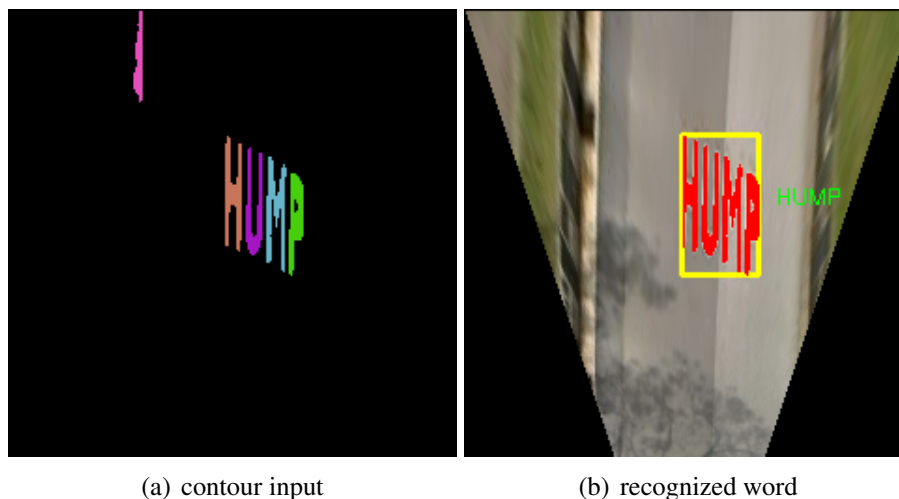


Figure 5.9: Word recognition

a contour group, and fuse the results together to recognize a complete word using a Bag of Words (BoW) model [142]. In our application, a “word” inside the “bag” is actually an uppercase English letter, and the “bag” is an English word shown by the markings. Our dictionary is composed of the 26 uppercase English letters. We construct a 26-entry vector as the feature vector of one word, with each entry of the vector representing the occurrence of the corresponding letter. With this feature vector, a multi-class classifier can be trained to recognize the different words. Figure 5.9 shows an example of word recognition.

5.5 Experiments

We use a common webcam (Logitech Pro Webcam C910) to acquire 640×360 images at 30 fps . It is mounted in the front of the vehicle, at the height of 1.5 m and with a tilting-down angle of 10 degrees . We define our IPM image to be 240×240 , with its ratio to the real word as 20 pixel/m . Although the IPM image covers a square area in front of the vehicle, the effective detection zone of our algorithm is an isosceles trapezoid, with its bottom width 4.0 m , top width 12.0 m and height 12.0 m . The bottom side of this trapezoid is 3.0 m ahead of the camera mounting position. Our algorithm is programmed in C++ with OpenCV [143], and is able to perform real-time classification and analysis for various markings, except for the word ones. Our experiments were carried out on the campus of the National University of Singapore, which is one typical urban road environment.

Table 5.2: Classification performance for road markings

Classifiers for markings	Lane	Arrow	Crossing	Word
labeled markings	2720	6912	732	490
weighted precision(%)	92.8	96.5	97.2	91.8
weighted recall (%)	92.8	96.4	97.2	91.8

Table 5.3: Confusion matrix of arrow classification

		Prediction Outcome							
		Noise-0	Arrow-1	Arrow-2	Arrow-3	Arrow-4	Arrow-5	Arrow-6	Arrow-7
Actual Value	Noise-0	4247	108	21	12	4	5	3	7
	Arrow-1	57	1799	0	0	2	2	0	0
	Arrow-2	4	0	247	17	0	0	0	0
	Arrow-3	6	2	3	151	0	2	0	0
	Arrow-4	0	0	0	0	40	0	0	0
	Arrow-5	0	0	2	0	0	166	0	0
	Arrow-6	0	0	0	0	0	0	72	0
	Arrow-7	0	0	0	0	0	0	0	196

To train the classifiers for different marking types, images are collected beforehand. Our data collection took place on a cloudy day when the lighting conditions were relatively mild. Marking contours in the collected images were labeled manually. Four classifiers were trained independently using SVM with the labeled data. Table 5.2 shows the classification performance of each classifier under 5-fold cross-validation. We analyze the precision and recall rates to better evaluate our algorithm: precision measures what fraction of the detections are actually the studied markings, and recall measures what fraction of the actual studied markings are detected. It can be seen that each type of markings is classified well, with precision and recall rates all above 90%. This result justifies our idea of using contour features for marking classification. While the classifiers for the other 3 types of markings are all binary classifiers, the one for arrow marking is an 8-class classifier. Table 5.3 shows the confusion matrix for arrow classification. From the table, it is found that the first 3 types of arrows are more easily misclassified as noise than the other 4 types. This is because that when seen from a distance, these arrows appear blurred and resemble lane segments.

While the classifications for lanes and arrows are based on individual markings, the recognitions of zebra-crossings and words are performed on the group of contours. As for the zebra-crossings, we try to extract the groups with enough contour members. Since the precision of crossing contour classification is high, the precision of its group classification is also high. The precision of the zebra-crossing classification is more

Table 5.4: Confusion matrix of word classification

		Prediction Outcom					
		Noise	"AHEAD"	"SLOW"	"STRIP"	"HUMP"	"X-ING"
Actual Value	Noise	53	5	0	1	0	1
	"AHEAD"	5	70	4	4	2	0
	"SLOW"	0	5	22	2	0	1
	"STRIP"	0	0	0	18	0	0
	"HUMP"	0	0	0	0	15	0
	"X-ING"	0	4	0	0	0	76

than 98%. As for the words, OCR is performed at the letter level, and we train a word classifier based on the OCR outputs. Tesseract is used for single-letter classification, and its precision is about 55% in our application. By training a word classifier using a BoW model, the accuracy of word detection is considerably improved. In our experiment, we manage to identify 5 common words on road, and the precision is 88%. The confusion matrix of word classification is shown in Table 5.4.

Besides the accuracy, computation time is another key issue for marking classification and analysis. Our processor is an Intel Core i5-3550. We use separate threads for different operations and test their computation time. Table 5.5 shows the computation time of marking detection and analysis. The procedure of image processing is composed of IPM transformation, binarization, and contour extraction, and its computation time is around 3 ms . The extracted contours are then sent to different modules for processing. While the computation time of each module may vary dramatically for different images with different markings, we record the time when their corresponding contours appear. The lane module generally takes 6 ms to process contours from one single image, while in contrast the arrow module only needs 2 ms . The reason is that the lane module involves contour skeletonization and RANSAC operations, which is computationally intensive. The zebra-crossing module takes 3 ms to classify the contours, cluster them, and find the crossing areas. Our camera is working at the frequency of 30 fps , and the above 3 modules are able to perform online processing for their relevant markings. The word module takes a much longer time compared to the above 3 modules. In our implementation, Tesseract takes around 30 ms to recognize a single character, making the recognition of a word to be more than 150 ms .

To test the robustness of our proposed method, we carried out a series of tests under different lighting conditions. Since camera exposure control is used to the camera for the

Table 5.5: Computation time of different processes

Processes	Image Processing	Lane	Arrow	Crossing	Word
Time (ms)	3	6	2	3	>150

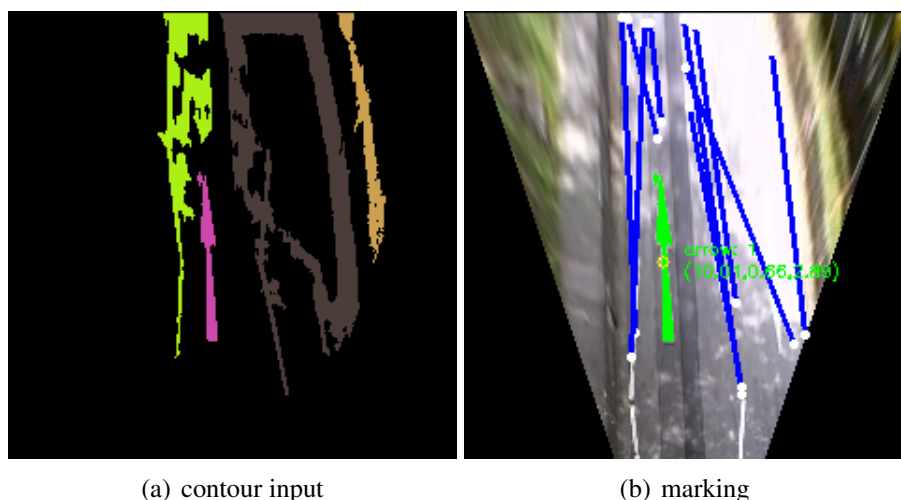


Figure 5.10: Shadow-highlight situation

IPM ROI, our algorithm can handle moderate shadow-highlight effects and perform well in most scenarios. However, when the shadow-highlight effects are too strong, noise will appear and the detection performance will be undermined. Figure 5.10 shows an example where false lane markings are detected due to nonuniform illumination on the road surface. Despite the challenges brought by shadows and highlights, our algorithm shows good performance in most scenarios. Some impressive results are shown in Figure 5.11.

More details of our experiments can be found in Video (8) in Appendix B.

5.6 Summary

In this chapter, we propose a general framework for road marking detection and analysis, which is able to support various types of markings. Marking contours of different types are extracted indiscriminately from an image processing procedure, and sent to respective modules for independent classifications and analyses. Four common types of markings are studied as examples: lanes, arrows, zebra-crossings, and words. Our proposed method shows good accuracy in experiments.

The contribution of our work is the proposed framework for road marking detection and analysis; unlike the existing approaches which are case-specific, the proposed

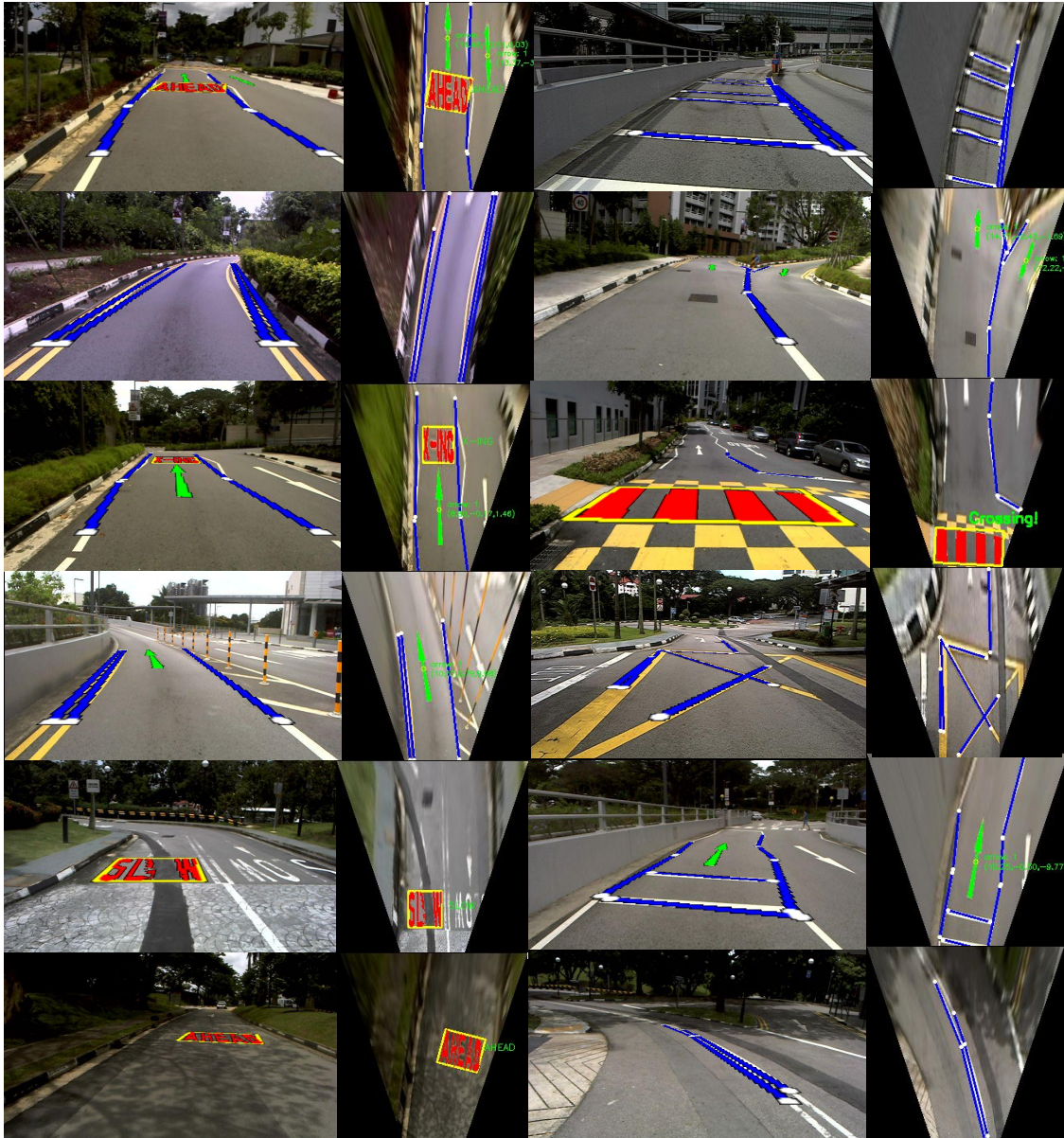


Figure 5.11: Marking detection and analysis results

method is able to detect various types of markings under a uniform framework, and achieves good performance.

In future work, we will try to improve our algorithm in the following three respects. Firstly, a more robust method will be developed for the contour extraction process. Here extraction simply relies on brightness thresholding by assuming that road markings are generally clear and complete, which however may fail when the markings are worn or the shadow-highlight effect is too strong. Some shadow removal algorithms may be applied to alleviate this problem. Secondly, in the current setup, different classification-analysis modules operate in an independent manner, which may generate contradictory classification results. A consensus process will be developed to eliminate the contradictions based on some semantic knowledge, which will help to improve the classification accuracy of the whole system. Thirdly, due to the lack of suitable public datasets, we do not provide comparative evaluation of our method with other state-of-art algorithms, which will be addressed in the future work.

Chapter 6

Road Detection and Mapping using 3D Rolling Window

Road marking detection and surface-boundary detection are the two major research topics in road detection. While the previous chapter introduces our work of road marking detection, it is only applicable to painted roads where markings exist. This chapter addresses the problem of road surface-boundary detection, which however can be applied to the general urban road environment.

6.1 Introduction

LIDARs have played a dominant role in the research on road surface-boundary detection, as reviewed in Section 2.3.2. While most existing methods as well as our initial work (Section 3.2) use a 2D LIDAR for road boundary detection, the algorithms are based on the processing of individual 2D scans, which have several disadvantages to be discussed in the next paragraph.

In this research, we employ the idea of a 3D rolling window for road surface and boundary detection. Unlike algorithms directly processing 2D scans, our method applies 3D perception techniques for the detection purpose, and has several advantages over the 2D algorithms. Firstly, 2D algorithms have a strong assumption about the detection scenario: road boundaries should always intersect the projected laser line on the ground, and there are at most two boundary points in each scan. This assumption generally works well; however, it does not apply to some tricky situations, e.g. where road boundaries are

parallel to the projected laser line. On the contrary, our algorithm using accumulated 3D data does not have this assumption, and is able to handle all the situations. Secondly, our algorithm is able to implicitly utilize the temporal relationship between adjacent scans, and hence will be more robust to noise. This is because our detection makes use of the features of each point, which incorporate the information of their neighboring points from multiple scans. Thirdly, while other algorithms use the tilted-down LIDAR solely for road detection, the 3D data accumulated in our algorithm can also be used for other object recognition purposes.

The method of a 3D rolling window has been introduced in Section 4.2.1. Compared to commercial 3D sensors like Velodyne, this technique provides a cheaper way to acquire 3D data. While a Velodyne can perceive the 3D environment through one snapshot, the rolling window achieves this through temporal data accumulation and is hence suitable for static object perception. Since the accuracy of the rolling window is not yet clear, its probabilistic characteristics will be studied. Considering the characteristics of the rolling window, a cascaded process of road detection is developed with region-growing and classification methods. We further develop a probabilistic framework for road mapping with the detection results.

The remainder of this chapter is organized as follows. Section 6.2 studies the idea of the 3D rolling window. Section 6.3 presents the cascaded process of road detection. Section 6.4 discusses the idea of probabilistic road mapping. The experimental results and analyses are shown in Section 6.5. Section 6.6 concludes this chapter.

6.2 3D Rolling Window

3D perception is the ability to perceive the environment in three dimensions. It is always desired for a robot navigating in the real world, and is getting more and more popular [133]. There are many ways to get 3D data, including using stereo vision, using a 3D LIDAR, and accumulating 3D data from 2D range scanners [144]. We use a tilted-down LIDAR to generate a 3D point cloud of the environment in a rolling-window way. A fixed tilted-down single planar LIDAR enables the reconstruction of the environment by sweeping across the ground surface. In this section, we will introduce the construction of the 3D rolling window, study its probabilistic characteristics, and incorporate the

probabilistic characteristics into the 3D point cloud. (While the construction and maintenance of a 3D rolling window has been introduced in Section 4.2.1, to make this chapter self-contained and easy to read, some details are restated here.)

6.2.1 Construction and Maintenance

A 3D rolling window is used to accumulate different scans received across a short distance. The size of the window is flexible and the rolling window forms a local snapshot of the 3D environment. It moves together with the vehicle, where new incoming scans are added into the window, and the old samples get discarded. Let w denote the window width, i denote the 2D scan index, p_i the points in the i th scan, n the latest scan index, and β the control distance. P_n is the 3D point cloud updated by the newest scan n , which is accumulated according to

$$P_n = \bigcup_{k=n-\lfloor w/\beta \rfloor} \{p_k, \dots, p_n\} \quad n > \lfloor w/\beta \rfloor . \quad (6.1)$$

As shown in Figure 4.2, a new scan is only inserted when sufficient distance β is reached. This prevents the rolling window from getting redundant points at the same place. There are many ways to control the processing of the collected 3D data. The way used in this work is to process the rolling window when the vehicle traverses a certain distance interval w .

6.2.2 Probabilistic Characteristics

The 3D rolling window provides a low-cost method to get 3D data. However, the accuracy of the accumulated point cloud is not yet clear. It is necessary to understand its probabilistic characteristics before using it. While below we give a general analysis for the accumulation process, we are not going to calculate the full distribution of the 3D data. Instead, for our specific application, we only focus on the distribution of each single point in the z direction.

Figure 6.1 shows the coordinate systems during the data accumulation process. In the figure, p_i denotes the laser points from scan i , “Laser- i ” the LIDAR coordinate where points p_i are originally collected, “Baselink- i ” the vehicle-attached coordinate when accumulating scan i , “Baselink- t ” the vehicle coordinate at time t , and “Odom Coordinate”

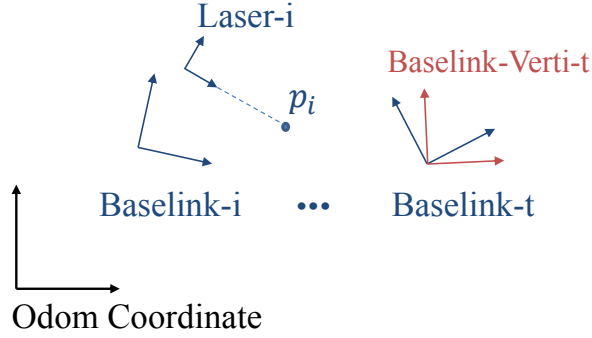


Figure 6.1: Coordinate system for 3D data accumulation

the dead-reckoning coordinate of odometry. For the vehicle-attached coordinate, its origin is at the center of the real wheel axis, with its x-axis pointing forward in the vehicle longitudinal direction, y-axis pointing left in the lateral direction, and z-axis pointing up. When a scan i is received, points p_i are calculated and transformed into the odometry frame to be stored. When data processing is needed at a certain time t , recorded points p_i are transformed from “Odom Coordinate” into the latest vehicle coordinate “Baselink- t ”. This transformation depends on pose estimation for vehicle coordinate “Baselink- t ”. However, pose estimation for the vehicle coordinate may suffer from severe noise in its pitch and roll angles, making the transformed data unreliable to use. A vertical coordinate frame is introduced to avoid this problem. The vertical coordinate frame, denoted as “Baselink-Verti- t ”, has the same origin and yaw angle with “Baselink- t ”, but its roll and pitch angles are set to zero. At time t , the accumulated 3D data are transformed into the “Baselink-Verti- t ” coordinates for further usage.

The data accumulation process can be represented as follows:

$$p_i^t = {}^t_{iB}T {}^{iB}_{iL}T p_i, \quad (6.2)$$

where $p_i^t = (x_i, y_i, z_i, 1)^T$ is the augmented position vector of points p_i in the coordinates of “Baselink- t ”, ${}^t_{iB}T$ denotes the 4×4 homogeneous transformation matrix from coordinate frame “Baselink- i ” to “Baselink- t ”, and ${}^{iB}_{iL}T$ denotes the transformation matrix from “Laser- i ” to “Baselink- i ”. The distribution of p_i^t is determined by the two transformation matrices ${}^t_{iB}T$ and ${}^{iB}_{iL}T$. In the cases where the LIDAR is fixed and ${}^{iB}_{iL}T$ is a fixed matrix, the p_i^t distribution is then only determined by ${}^t_{iB}T$, which is subject to the estimation of

the vehicle pose.

Let X_{iB}^t denote the pose of coordinate frame ‘‘Baselink-i’’ in ‘‘Baselink-Verti-t’’, and \bar{X}_{iB}^t the estimation value for X_{iB}^t from the vehicle odometry system. The probability distribution function of X_{iB}^t can be approximated by a Gaussian Distribution with mean value \bar{X}_{iB}^t , and the covariance matrix Σ_{iB}^t :

$$X_{iB}^t = (x, y, z, \alpha, \beta, \gamma)^T, X_{iB}^t \sim N(\bar{X}_{iB}^t, \Sigma_{iB}^t). \quad (6.3)$$

Here vehicle orientation is represented in Euler angles (roll-pitch-yaw), with α, β, γ denoting yaw, pitch, and roll respectively. Σ_i^t reflects the uncertainty of the dead-reckoning system, increasing with vehicle driving distance and turning angle. When the LIDAR mounting is fixed, p_i^t is only a function determined by X_{iB}^t , which is $p_i^t = f(x, y, z, \alpha, \beta, \gamma)$. By linearizing the function f , distribution of p_i^t can be represented as a Gaussian Distribution:

$$p_i^t \sim N(\bar{p}_i^t, \Sigma_p), \quad (6.4)$$

where

$$\begin{aligned} \bar{p}_i^t &= f(\bar{x}, \bar{y}, \bar{z}, \bar{\alpha}, \bar{\beta}, \bar{\gamma}), \\ \Sigma_p &= F \Sigma_{iB}^t F^T. \end{aligned}$$

F is the Jacobian Matrix of function f with respect to X_{iB}^t . The full distribution for single point p_{i_n} can be calculated with the above equation. In our specific application for road detection, we are only interested in the point distribution in the z axis. The tilted-down LIDAR is fixed in the front of the vehicle and its mounting pose in the vehicle coordinate is denoted as $X_{iL}^{iB} = (x_l, y_l, z_l, 0, \beta_l, 0)$, which determines the transformation matrix ${}^{iB}_L T$. Then we have:

$$z_i = f_z(\beta, \gamma, x_l, y_l, z_l, \beta_l, r, \theta), \quad (6.5)$$

where

$$f_z = -\sin \beta (\cos \beta_l (r \cos \theta) + x_l) + \cos \beta \sin \gamma ((r \sin \theta) + y_l) + \cos \beta \cos \gamma (-\sin \beta_l (r \cos \theta) + z_l) + z .$$

r denotes the range value in the laser scan, and θ is the laser beam angle. To calculate the variance of z , some simplification is adopted for Σ_{iB}^t . Usually the six state variables can be correlated, but here we define it as a diagonal matrix as adopted in [96] :

$$\Sigma_{iB}^t = \text{diag}(\sigma_x^2, \sigma_y^2, \sigma_z^2, \sigma_\alpha^2, \sigma_\beta^2, \sigma_\gamma^2) . \quad (6.6)$$

Considering the fact that β and γ are usually very small, F_z (the Jacobian Matrix of f related to z) can be calculated and approximated as:

$$F_z = (0, 0, 1, 0, \frac{\partial f_z}{\partial \beta}, \frac{\partial f_z}{\partial \gamma}) \quad (6.7)$$

$$\approx (0, 0, 1, 0, -r \cos \theta \cos \beta_l - x_l, r \sin \theta + y_l) .$$

The distribution of the points in the z direction will be:

$$z_i \sim N(\bar{z}_i, \sigma_{z_i}^2) , \quad (6.8)$$

where

$$\bar{z}_i = f_z(\bar{\beta}, \bar{\gamma}, x_l, y_l, z_l, \beta_l, r, \theta),$$

$$\sigma_{z_i}^2 = \sigma_z^2 + (r \cos \theta \cos \beta_l + x_l)^2 \sigma_\beta^2 + (r \sin \theta + y_l)^2 \sigma_\gamma^2 .$$

Given the fact that the width of the rolling window is small, and the road surface is generally horizontal, σ_z^2 is usually very small and negligible. It can be seen that point variance in z is mainly determined by variances of vehicle pitch and roll angles, and points from side beams of LIDAR are more sensitive to roll angles rather than those from the central ones. In our system, both roll and pitch angles are retrieved from one

Inertial Measurement Unit (IMU). Their variances are approximated as:

$$\sigma_\gamma = a_1 \frac{d\gamma}{dt} + a_2 \gamma, \quad \sigma_\beta = b_1 \frac{d\beta}{dt} + b_2 \beta. \quad (6.9)$$

where a_1, a_2, b_1, b_2 are fixed parameters selected in experiments.

6.2.3 Extended Point Cloud

A point cloud is a cloud of points in 3D space, which has been the most popular representation of 3D data. One point may have 3D position information, color, and other features. Rusu *et al.* present a Point Cloud Library (PCL) to facilitate the processing of 3D data in [133], which has been widely used. However, while it is well recognized that 3D data may be noisy, especially in stereo-vision, the probabilistic characteristics of the point cloud are usually not used. In our research, we incorporate point variance σ_{z_i} for noise filtering, and road boundary recognition.

In our application, the 3D point cloud is generated by accumulated 2D laser scans. When laser points are calculated and stored in the point cloud format, information belonging to the accumulation process is lost, such as the serial number (or ID) of the scan that the point belongs to, the beam angle that the point belongs to, etc. However, such information may be useful for the 3D data processing purpose. We try to preserve such useful information, and demonstrate its importance in later sections. One point p' of the extended point cloud is defined to have the following attributes: $p' = (x, y, z, \sigma_{z_i}, laser_ID, beam_angle)$.

6.3 Cascaded Process of Road Detection

This section presents a cascaded road detection process based on region-growing and classification methods. Region-growing is a simple segmentation method which has been used in image processing. Rusu *et al.* generalize it for 3D point cloud segmentation in [145]. Similarly we use the region growing method to extract road surface and boundary from the 3D rolling window. Our assumption is that the road surface is generally smooth in the center, but changes drastically at road boundaries. Surface curvature can be calculated to represent this smoothness information. While road surface

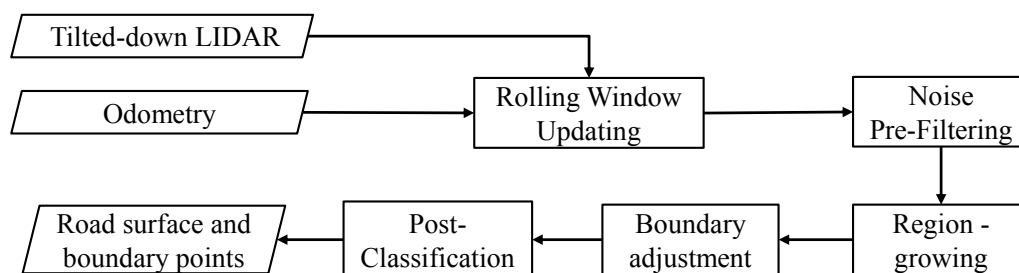


Figure 6.2: Road detection flowchart

is represented by a cloud of points, its curvature near a point can be calculated through an analysis of the eigenvectors and eigenvalues (or PCA - Principal Component Analysis) of a covariance matrix created from the nearest neighbors of this query point, as introduced in [145].

The region-growing method is easy to implement. However, due to the noisy and sparse nature of the accumulated 3D data, some pre-filtering, adjustment and post-classification processes have to be cascaded to yield good detection results. In the first step, most noisy points in the 3D rolling window are filtered by checking the integrity of their corresponding laser scans. The region-growing method is then applied to extract road surface and boundaries, followed by a boundary adjustment procedure. Finally, the adjusted boundary points are further filtered by a classification process. The flowchart of this algorithm is shown in Figure 6.2.

6.3.1 Noise Prefiltering

According to Equation 6.8, point variance in the z direction is mainly determined by variances of pitch and roll. In common cases, a noisy pitch angle contributes to most of the noisy points in the rolling window. When the vehicle passes over some speed bumps, its pitch angle will change drastically, making a large pitch variance according to Equation 6.9. Points from laser scans collected at this time hence have high variance in z , and usually unexpected large curvature values due to misalignment. Since most points from these scans are useless, it is desired to identify these scans and filter them out before applying any road detection algorithm.

One binary classifier is trained using a Support Vector Machine (SVM) to detect the

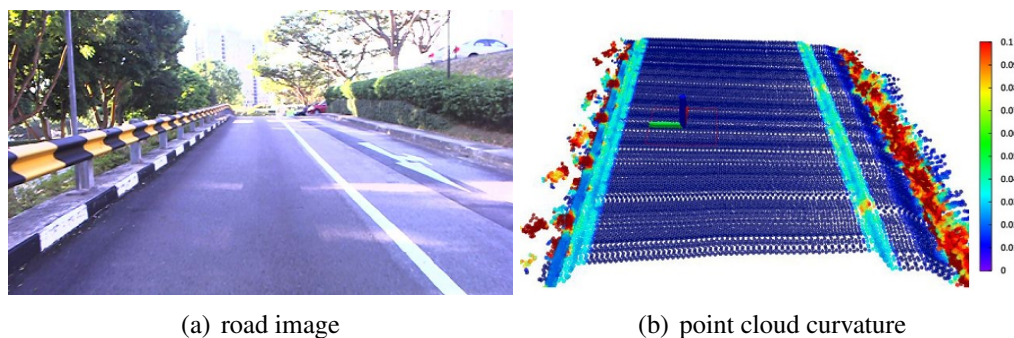


Figure 6.3: Curvature hint for road detection

noisy scans. The response of each laser scan is good or noisy. Its feature vector is composed of two parts. The first part is the angular information acquired from the IMU, including pitch, roll, pitch speed, and roll speed, which will give a description of the temporal vehicle motion status and is important for position variance in the z direction. The second part is an array of curvature values. These curvature values belong to points from the middle part of the scan, which gives us a hint about how many points will be classified as boundary points if the scan is not filtered.

The feature vector can be constructed using the extended format for a point cloud. The “laser_ID” and “beam_angle” attributes in each point can be used to assemble points together and generate the curvature arrays for their corresponding scans. The “laser_ID” can be used to refer to scan-related angular information, which is recorded from the IMU during the data accumulation process. Our training data are labeled manually. During our labeling process, if too many large-curvature points appear where they should not be, the scan is labeled as noisy.

6.3.2 Region-growing Method

Figure 6.3 shows a 3D point cloud generated from the rolling window. It is colored according to a point’s local curvature, which is the curvature of the neighborhood surface. It can be found that curvature of the road surface is generally low, while curvature at the surface edge is high. If we give a seed point to the middle of the road, and connect every point whose local curvature is low, we can extract the whole road surface. In addition, road boundary points are extracted where high-curvature points are encountered in the region-growing process. In our application, the region-growing process is applied to the 3D point data in the rolling window, which works in the vehicle-attached coordinates.

Algorithm 1: Pseudo-code for region-growing algorithm

Input: 3D point cloud P , seed point $p_o \in P$
Output: Road surface point set S , boundary point set B

```

1  $S := \emptyset, B := \emptyset$ ; add  $p_o$  to  $S$ ;
2 for every point  $p_i$  in  $S$  do
3   find its neighbourhood points  $P_i$  within a searching radius  $l$ ;
4   for every point  $p_j$  in  $P_i$  do
5     if  $p_j$ 's curvature  $<$  curvature threshold  $T$  then
6       if  $p_j \notin S$  then
7         add  $p_j$  to  $S$ 
8       end
9     else
10      if  $p_j \notin B$  then
11        add  $p_j$  to  $B$ 
12      end
13    end
14  end
15 end

```

The seed point is selected from the point cloud which is several meters away, right ahead of the vehicle center. The pseudo-code for the region-growing algorithm is shown in Algorithm 1.

6.3.3 Road Boundary Adjustment

In the region growing process, curvature threshold T is a global parameter to be determined. A small threshold value is set to guarantee that no road boundary points are missed. While the region growing method is easy to implement, the selection of small threshold value T will introduce a new problem. Due to the sparsity of collected 3D data, neighborhood points nearby the road boundary will have curvatures comparably large to those of the boundary points. The small threshold T will terminate the region growing process early before it really reaches road boundaries, and extract its neighborhood points as boundary points. In other words, these extracted boundary points (candidates) are inaccurate, and hence the surface points (since the supposed boundary points are actually surface points). Thankfully, since boundary points are the local maximum curvature points, we can adjust the boundary candidates to their local maxima for better accuracy. Figure 6.4 shows the boundary point adjustment, where purple points are boundaries before adjustment, and green ones are the adjusted boundary points. The

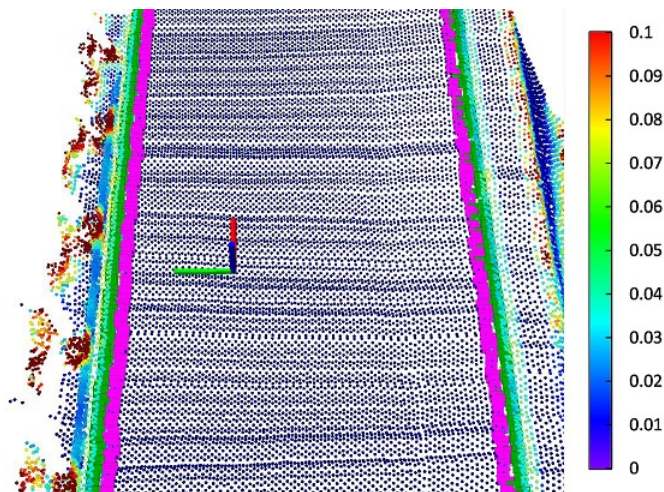


Figure 6.4: Boundary point adjustment

points between the old and new boundaries will be classified as road surface points. The road width after adjustment is slightly larger, which however reflects the actual road width.

6.3.4 Post Classification

In our previous discussion, curvature is used as the only boundary criterion. In the region growing process, if a point being inspected has a curvature less than a certain threshold, it is classified as a surface point; otherwise, it is a boundary point. This simple criterion generally works fine. However, it will lead to some false-positive boundary points where points become too noisy in the z direction, or too sparse. It is desired to take into account point z -variance and local density (number of neighboring points) for boundary detections. Besides, maximum local height difference (maximum height difference between a point and its neighbors) will also help to eliminate spurious boundary points.

We propose to apply a post-classification process to filter the extracted boundary points. One SVM binary classifier is trained for this purpose. The response of each point will be surface or boundary. Its feature vector is composed of its curvature, z -variance, local density and maximum local height difference. The training data are boundary points extracted after boundary adjustment, and are labeled manually. This post-classification process improves the accuracy of the road boundary classification, and hence the road surface extraction. It should be clarified that this process will not help to erase boundary points caused by vehicles or pedestrians. These points can be

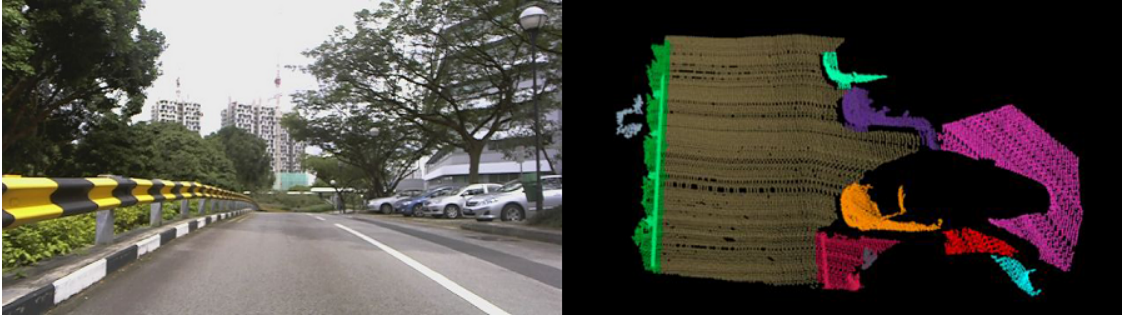


Figure 6.5: Point cloud segmentation example

treated as temporary boundary points, and used for various purposes like online path planning.

6.3.5 Road Detection for Point Cloud Segmentation

The road surface is the most dominant structure in the urban road environment. The detection of the road surface from 3D point clouds can be used for point cloud segmentation of the road environment. Since the road surface serves as a supporting ground for other objects, by first detecting and removing it from the 3D point cloud, points from different objects are naturally separated. Then the remaining part of the point cloud can be segmented and clustered easily. Figure 6.5 shows one example of point cloud segmentation, where different point cloud clusters are shown in different colors. The point cloud segmentation is the by-product of our road detection, and its results can be used for other object recognition purposes.

6.4 Probabilistic Road Mapping

In this section, a probabilistic framework is proposed for global road mapping, with road detection results from above. A global road map can not only be used to help vehicle path planning, but also provide texture information for vehicle localization and other perception purposes [41, 89].

An Occupancy Grid Map (OGM) [16] is used for this mapping purpose. An Occupancy Grid Map is a map that represents a map of the environment by an evenly spaced grid. Each grid cell represents a variable to be estimated. In our application, the variable is a binary variable with two possible values, road surface or road boundary. By

utilizing our high-accuracy localization [8], this mapping problem is simplified from a Simultaneous Localization and Mapping Problem (SLAM) to an Occupancy Grid Mapping problem, which is to estimate the posterior over map given the data: $p(m|z_{1:t}, x_{1:t})$, where m is the map, $z_{1:t}$ is the measurement from time 1 to t , and $x_{1:t}$ is the set of robot poses from time 1 to t . By assuming independence between grid cells, the posterior can be further factorized into $p(m|z_{1:t}, x_{1:t}) = \prod_i p(m_i|z_{1:t}, x_{1:t})$, where m_i denotes a grid cell in map m [16]. An inverse sensor model $p(m_i|z_t, x_t)$ is needed for the above estimation process.

In our application, z_t are the extracted road surface and boundary points in the vehicle-attached coordinates (“Baselink”) at time t , and x_t is the vehicle pose in the global coordinates (“map”). The extracted points z_t can be transformed into map coordinates given the knowledge of x_t . We have $p(m_i|z_t, x_t) = p(m_i|z_{t_i})$, where z_{t_i} is the point that falls into cell m_i . The inverse sensor model is defined as:

$$\begin{aligned} p(m_i = \text{road_surface} | z_{t_i} = \text{surface_point}) &= k_1, \\ p(m_i = \text{road_surface} | z_{t_i} = \text{boundary_point}) &= k_2, \end{aligned}$$

where k_1, k_2 are parameters to be selected in experiments. Generally k_1 should be larger than $(1 - k_2)$, reflecting the fact that there is more noise in boundary points than in surface points. This noise may come from false-positive boundary points, or temporal boundary points caused by vehicles or pedestrians. Given the inverse sensor model, a Static Bayes Model is applied to calculate the posterior $p(m|z_{1:t}, x_{1:t})$.

6.5 Experiments

6.5.1 Experiment Setup

To test the performance of our algorithms, we conducted several experiments. We use the same test bed as in Section 4.4.1. For prompt data processing and reducing computational cost, the width of rolling window w is set to a small value $0.5 m$. Its control distance β is $0.02 m$. We conducted several experiments to test the proposed road detection and mapping method. Our experiment was conducted on the Engineering Campus, National University of Singapore.

6.5.2 Experiment Results

The first experiment is to check the probabilistic characteristics of the 3D rolling window. Parameters in Equation 6.9 are manually set, with $a_1 = b_1 = 0.1$, $a_2 = b_2 = 0.02$. Figure 6.6 shows three typical cases for 3D data collected from the rolling window. The left column of each case shows the corresponding angle plots over data accumulation; the middle column is the angular rate plots; the right column shows the accumulated point cloud colored by the variance value in units of meters, and the red line shows vehicle trajectory over time. In general straight-line traversal, both pitch and roll angles keep relatively stable, making point z-variance small. The quality of the 3D rolling window is generally good at this time, as shown by the upper picture. When the vehicle is making a turn, the absolute value of roll increases (the vehicle is rolling to one side), making laser points from sides of the LIDAR less accurate, as shown in the middle picture. The most noisy situation comes when the vehicle traverses through bumpy road: the vehicle keeps pitching up and down, making 3D points accumulated from this time unreliable, as shown in the lower picture. In real experiments, we found that the noise in pitch angle contributes to the most noisy points. It is desired to filter 3D points collected from this time in a pre-filtering process before applying the road detection algorithm, which is shown in Figure 6.7. In the curvature plot, one light blue swathe of points appearing in the middle of road is noisy. These noisy points are filtered through the pre-filtering process, and visualized in pink. To train the binary classifier using SVM, there are in total 24309 scans collected and labeled, among which 1546 scans are noisy scans. The classifier is trained using 5-fold cross validation, and achieves 99.4% total accuracy.

The second experiment is to test the cascaded road detection process. It takes less than 0.1 second to process the data from the rolling window, whose width is 0.5 m with 2000 to 3000 points. Since our vehicle moves at the low speed of 3.0 m/s, the road detection can be run as an online process. After region-growing and boundary adjustment, road surface and boundary points are extracted. A binary classifier using SVM is applied to filter the extracted boundary candidates. To train this classifier, we labeled in total 14893 candidates, and 2963 of them are false boundary points. Table 6.1 shows the classification performance under 5-fold cross validation. The “Original” column shows the detection results from region growing. Among all the boundary candidates, only 85.3% are the boundary points, while the rest are the misclassified surface points, due

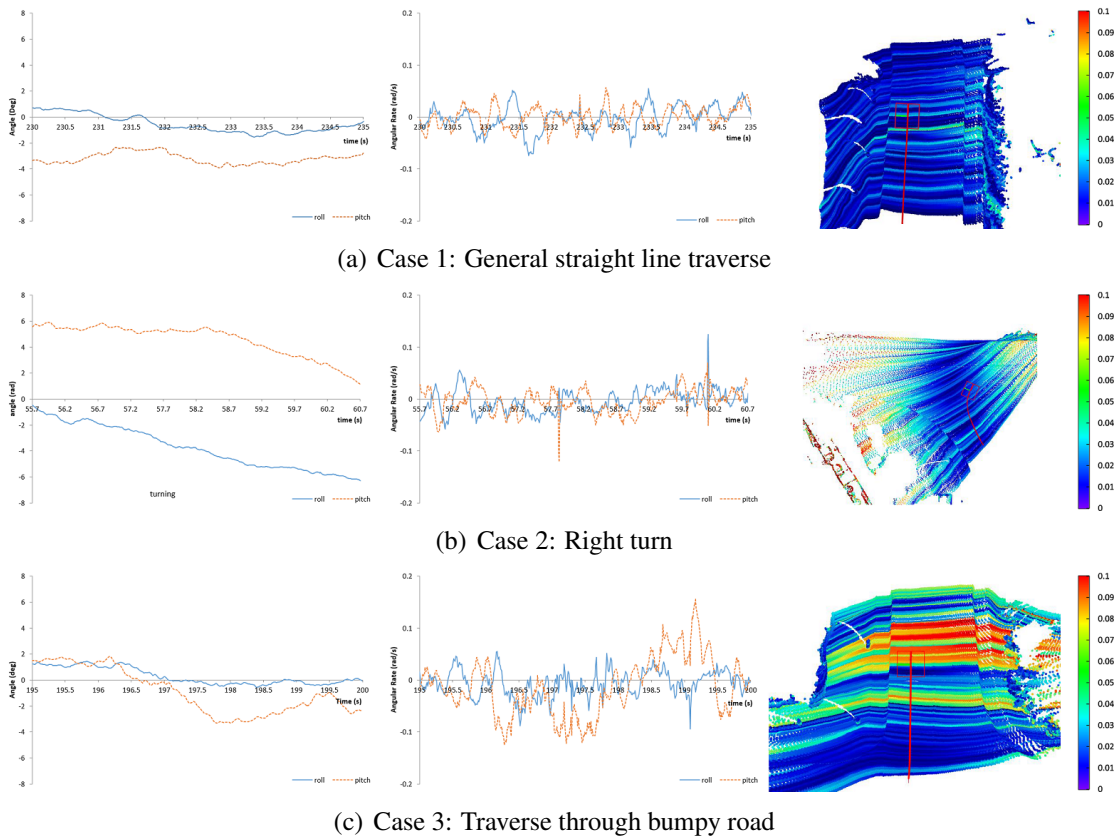


Figure 6.6: Three typical cases of rolling window. The left column of each case shows the corresponding angle plots over data accumulation; the middle column is the angular rate plots; the right column shows the accumulated point cloud colored by the variance value, and the red line shows vehicle trajectory over the time.

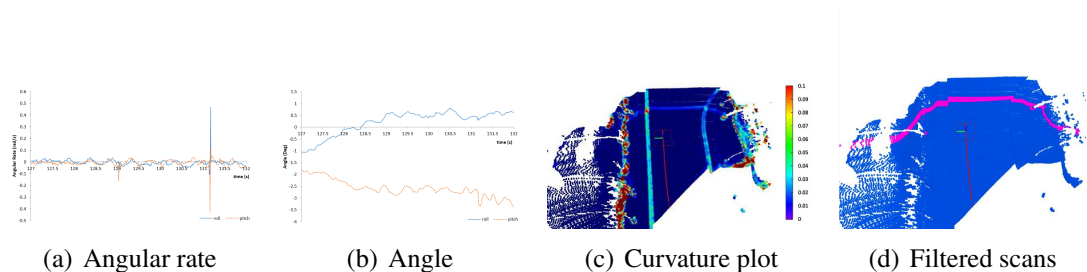


Figure 6.7: Noisy scan rolling window filtering. In the curvature plot, one light blue swathe of points appearing in the middle of road is noisy. These noisy points are filtered through the pre-filtering process and visualized in pink color.

Table 6.1: classification accuracy for boundary candidates

classification methods	Original	OneR	SVM1	SVM2
recall of boundary (%)	100.0	95.8	95.7	98.2
recall of surface (%)	0.0	84.8	88.3	91.8
total accuracy (%)	85.3	94.2	94.6	97.3

to a small curvature threshold value. For comparison, we apply the “OneR” classification method to boundary candidates using only the curvature feature, and 94.2% total accuracy is achieved. This is the best classification performance that can be achieved using only the curvature feature. “SVM1” denotes SVM classification using curvature and z-variance features. It is found that point z-variance helps improve the recall rate of the surface in the boundary candidates considerably. In other words, it reduces the false positive boundary points. “SVM2” denotes classification using full features of curvature, z-variance, local density, and maximum local height difference. The total accuracy of boundary candidate classification can be considerably improved by applying the SVM filtering process. Figure 6.8 shows one instance of improved road detection. The green points are the extracted road boundary, and the yellow points are the road surface. More results of road detection can be found in Figure 6.9.

The third experiment is about road mapping. To generate the road map, our vehicle is driven around the Engineering Campus three times with vehicle localization and road detection running. The localization algorithm helps to transform road detection results in the global “map” coordinates, and the transformed road detection results are used to estimate one occupancy grid road map. For the inverse sensor model, k_1 and k_2 are chosen to be 0.9 and 0.2 respectively. Figure 6.10 shows results of road mapping. The left figure shows one occupancy grid map of the surveyed road with an area of $429\text{ m} \times 475\text{ m}$. Its resolution is $0.1\text{ m}/\text{pixel}$. The road surface is overlaid on a satellite map for comparison, as shown in the right figure. It can be seen that the proposed algorithm mapped out the driven road with good accuracy.

More details of our experiments can be found in Video (9) in Appendix B.

6.6 Summary

In this chapter we present a road detection and mapping algorithm using the 3D rolling window. The probabilistic characteristics of the rolling window are studied. A cascaded

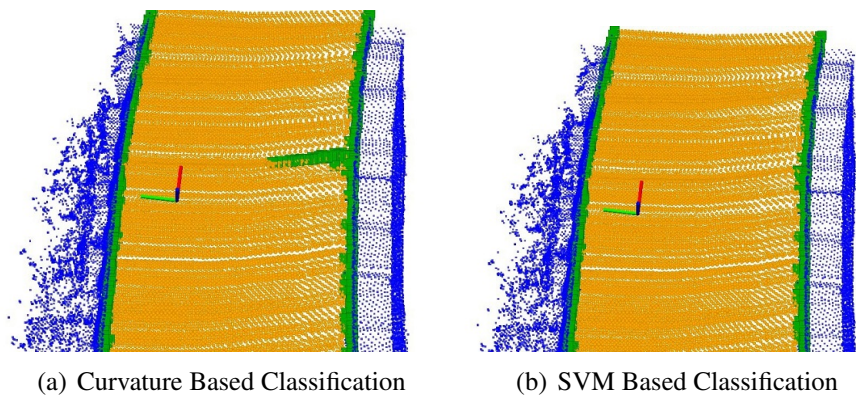


Figure 6.8: Increased classification accuracy through SVM

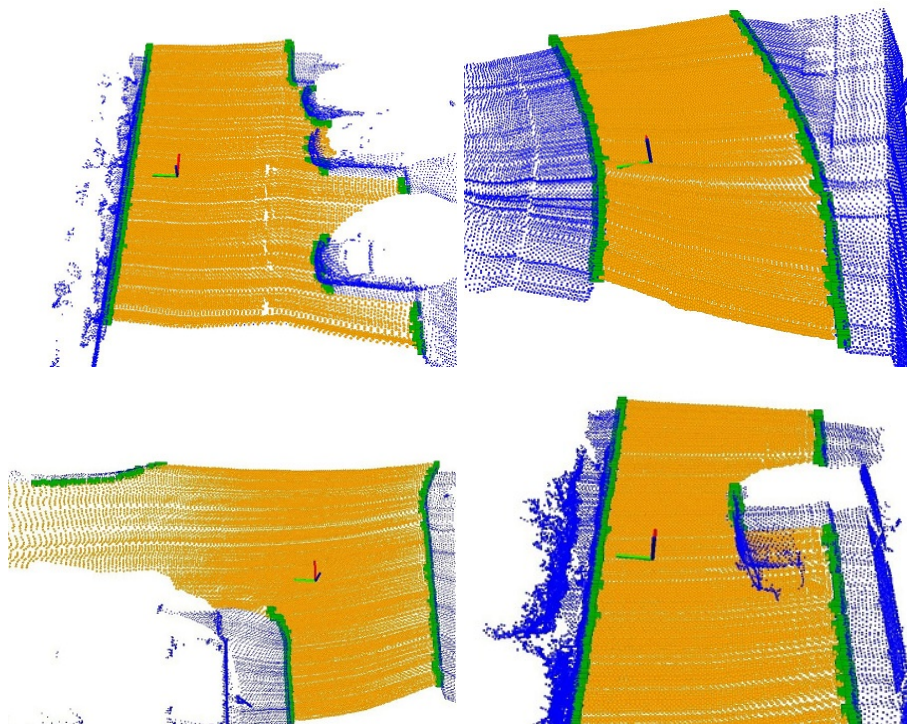


Figure 6.9: Results of the SVM classification of road boundary and surfaces

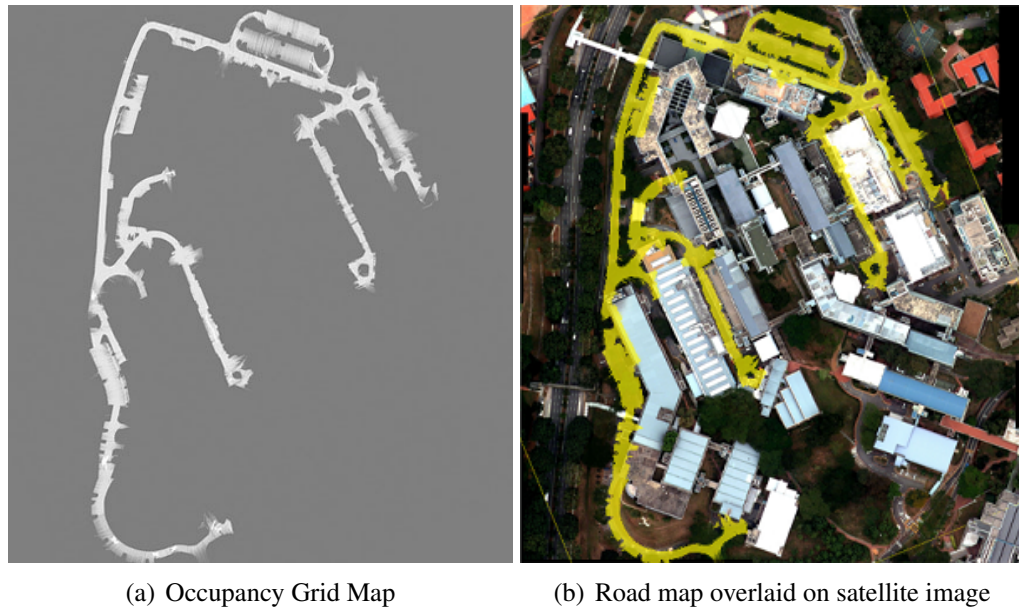


Figure 6.10: Road mapping results

process is developed for road detection using region growing and classification methods. A probabilistic framework is proposed for road mapping purposes. The accuracy of road detection and performance of road mapping are shown through experimental results.

The contributions of our work are two-fold. Firstly, we realize road surface-boundary detection using accumulated 3D data; compared to the existing 2D approaches [84], our algorithm does not have strong assumption on the sensing scenario and is able to handle temporal noise. Secondly, we study the probabilistic characteristics of the accumulated 3D data, which is important for the road detection and other recognition tasks.

Chapter 7

A Spatial-Temporal Approach for Moving Object Recognition with 2D LIDAR

For an autonomous vehicle to safely navigate in the urban environment, it has to be able to detect and interact with other static objects and dynamic human agents. While Chapter 5 and Chapter 6 present our work on static object recognition, to be more specific, road detection, this chapter is going to address the detection of dynamic human agents.

Compared to road detection, the recognition of dynamic human agents appears much more challenging. The main difficulty comes from the high intraclass variance of human agents [69]. For example, pedestrians may be dressed in different colors, vehicles may have different shapes and sizes, etc. To robustly and efficiently recognize these human agents remains a challenging problem nowadays. In our research on “dynamic” human agents recognition, a reduced problem of “moving” object recognition is studied: while every human agent has the potential to move - noted as “dynamic”, our attention is focused on recognizing those entities actually moving.

7.1 Introduction

For an autonomous vehicle’s safe navigation in an urban environment shared by other dynamic agents, the capability of reliable moving object recognition is desired. In this chapter, we propose a spatial-temporal (ST) approach for moving object recognition

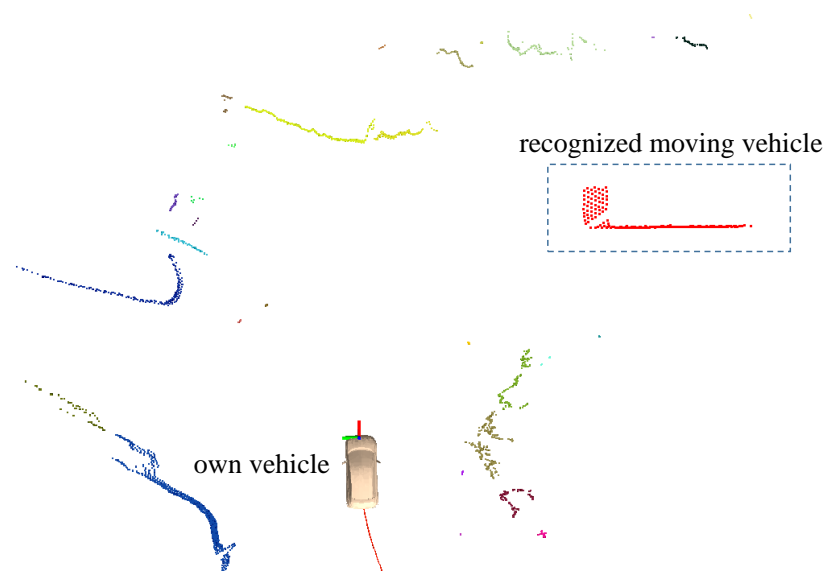


Figure 7.1: One example to illustrate spatial-temporal method. The red-green axes attached to the own vehicle represent the mounted 2D LIDAR. This image also captures a typical snapshot of a cluttered campus environment.

using only modest sensory data. Compared to more elaborate and costly solutions (e.g., outdoor depth cameras and 3D ranger finders), our method works with range readings obtained from a planar 2D LIDAR on a mobile platform. Using only range readings complicates object recognition because information is sparse relative to richer modalities such as vision. Furthermore, noise introduced by ego-motion (and other sources) can make static objects appear dynamic. We show that it is possible to obtain highly accurate object classification via temporal accumulation and a coupled classification process.

Existing work in moving object recognition decomposes the problem into two distinct sub-tasks: detection and classification. The former aims to discern the existence of moving objects, while the latter aims to recognize the objects' identities. As reviewed in Section 2.3.3, existing methods employ either the tracking or SLAM techniques for the moving evidence detection. However, as the tracking techniques may fail in cluttered environments and the SLAM techniques have high computational cost, reliable real-time detection of moving objects remains a challenging problem. In addition, while object classification in existing methods is usually based on individual measurements at each time cycle, it is vulnerable to similar-looking background noise.

A spatial-temporal approach for moving object recognition couples detection and classification into a single process. The basic idea of our approach derives from the observation that accumulated laser scans generally provide sufficient information for the

task. For example, it is difficult to recognize a vehicle from a single scan segment, because of its simple shape contour. However, Figure 7.1 illustrates that in the ST domain, the moving vehicle shows unique geometric features, i.e., a chain of shifted “L” shapes. The uniqueness of these features comes from not only the vehicle’s appearance in the spatial domain, but also from its motion pattern in the temporal domain. We show that these features can be exploited to create accurate classifiers.

Our method consists of three basic steps: (1) laser scans are first accumulated over a certain time window, (2) segmentation is then performed on the accumulated data to generate clusters, and (3) moving objects are finally recognized using the spatial-temporal features of these clusters. Compared to existing methods, our approach does not rely on object tracking nor local environment mapping and hence, it is more robust in cluttered environments and computationally lighter. Furthermore, since detection and classification are conducted in one single process, better recognition accuracy can be achieved. The rationale is that while motion patterns in the T-domain can aid object classification, appearance features in the S-domain can also help determine whether an object is moving (e.g, a bizarre-shaped cluster is more likely to be a static bush rather than a moving vehicle). A coupled process is able to fully utilize the ST information and benefit both sub-tasks.

7.2 Technical Approach

In brief, our method segments and clusters accumulated laser scans in a time-window, extracts relevant spatio-temporal features and then classifies each cluster. Segmentation is performed using a graph-based algorithm in the ST domain and classification is performed using the widely-used Support Vector Machine (SVM). The flowchart of our algorithm is illustrated by Figure 7.2.

7.2.1 Data Accumulation in T-domain

Laser scans are accumulated over a defined time window to collect N scans: $\mathcal{S} = \{s_{t_1}, s_{t_2}, \dots, s_{t_N}\}$, where \mathcal{S} denotes the collected scan set, and s_i each scan component. To represent ego-motion, we record the LIDAR’s pose (according to the robot’s odometry system) at each corresponding time stamp: $\mathcal{X} = \{x_{t_1}, x_{t_2}, \dots, x_{t_N}\}$, where each x_i is

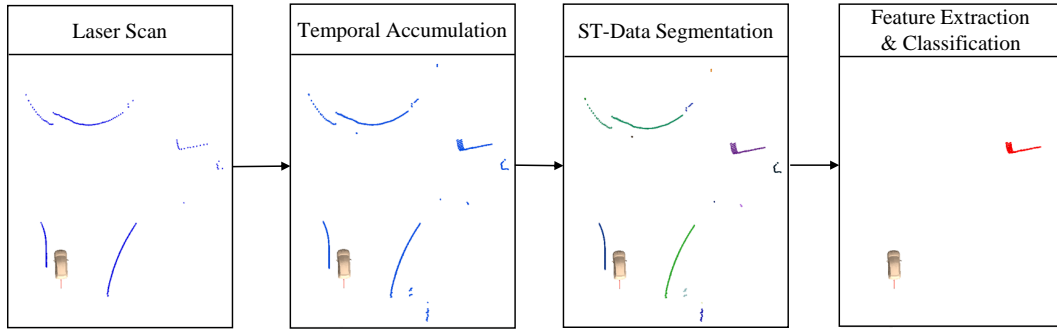


Figure 7.2: Flowchart of the spatial-temporal algorithm. Laser scans are received from the LIDAR sensor, and then accumulated in the temporal domain, visualized as blue points; the accumulated ST-data are then segmented into different ST-clusters, visualized in various colors; feature extraction and classification are performed on each ST-cluster, to recognize the moving objects, colored in red. The clay-colored vehicle model visualizes the own car.

the LIDAR pose corresponding to scan s_i . The accumulated laser scans \mathcal{S} and associated poses \mathcal{X} carry all the raw information required in our system.

7.2.2 Graph-based ST Segmentation

To segment the accumulated scans, we first convert the scan set \mathcal{S} into a point set \mathcal{D} , where each point d_i contains the position information p_i in the robot’s fixed odometry coordinate system, and its collected time t_i . In addition to this information, we maintain the conversion relationship between the scans and the points, such that each point in \mathcal{D} is mapped to its angle and range reading in \mathcal{S} .

We employ the graph-based region merging method [146] for segmentation of the transformed set \mathcal{D} . The advantage of this approach is that it is able to find a segmentation that is neither too coarse nor too fine. In brief, the data are treated as a graph, with points as nodes and edge weights indicating the dissimilarities between nodes. Initially each node is an individual component, and the algorithm performs pairwise region merging iteratively if the minimum edge weight connecting two components is less than the minimum internal difference (a scoring function); see [146] for more details. In our work, the edge weight (dissimilarity measure) between two points in the ST-domain is

the weighted Euclidean distance:

$$\mathcal{E}_w(d_i, d_j) = \|p_i - p_j\| + \alpha \times |t_1 - t_2| \quad (7.1)$$

where α is a weight parameter. Intuitively, this metric ensures that points which are close in both spatial and temporal domains are placed in the same cluster.

After the segmentation process, clusters of points in the ST-domain are obtained. Preliminary results showed that if we used each cluster as unorganized data and simply extracted its statistical features as a whole, performance was degraded, presumably due to a loss of information. As such, we define a ST cluster, denoted as \mathcal{ST} , as a collection of scan segments in a sequence together with their LIDAR poses:

$$\mathcal{ST} = \{z_{t_1}, z_{t_2}, \dots, z_{t_N}, x_{t_1}, x_{t_2}, \dots, x_{t_N}\} \quad (7.2)$$

where z_{t_i} is the scan segment collected at time t_i , and x_{t_i} its corresponding LIDAR pose. Given the segmentation results of data \mathcal{D} , to construct \mathcal{ST} is straightforward.

7.2.3 Spatial-Temporal (ST) features

In this section, we discuss the design of our spatial-temporal features. Recall that \mathcal{ST} not only contains the information about the object’s shape, but also the information related to their motion patterns. We construct our feature vector \mathcal{F} to maximize the amount of original information, while keeping its structure invariant to the scan number N :

$$\mathcal{F} = \{\{\hat{z}_{t_i}\}_N, \mathcal{M}, \mathcal{X}\} \quad (7.3)$$

where \hat{z}_{t_i} is a compressed representation for raw scan segment z_{t_i} , \mathcal{M} is a set of “shape moments” that captures the shape characteristics of the cluster, and \mathcal{X} is the pose set.

The compressed segment (\hat{z}). The compressed segment (CS) approximates each scan segment by a fixed number of key points and selected statistical features. Figure 7.3 illustrates the idea of the compressed segment. Here, we have used the Douglas-Peucker algorithm [147] to find the relevant key points. In addition, the number of points in between each pair of neighboring key points, and the variance of their distances to the line formed by the pair are also incorporated in the feature vector. To represent positional

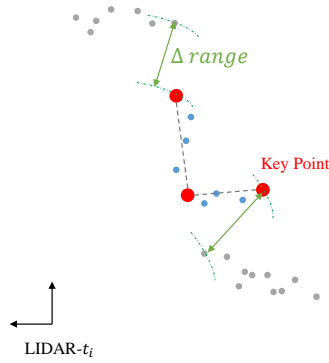


Figure 7.3: Compressed segment.

information relative to the background, range differences between the extreme points and their respective neighboring background points are also used. Table 7.1 summaries all the features in a CS feature vector.

The shape moments (\mathcal{M}). Although the scan segment information is incorporated into the feature vector by way of the compressed segments \hat{z}_{t_i} , some geometric information may still be lost due to compression. To better preserve the information, we project the scan points into the global odometry coordinates, and then extract the Hu-Moments [148] of the contour to convey the shape information of the overall point set.

The pose set (\mathcal{X}). To take into account robot ego motion, LIDAR poses at different time are incorporated into the feature vector. However, rather than using their original pose values in the global odometry frame, we transfer all the LIDAR poses into the latest LIDAR coordinates. This helps remove the irrelevant information of absolute positions and concentrate the classification on the relative movements.

Pose-Variant and Pose-Invariant Feature Sets. Figure 7.4(a) illustrates the spatial-temporal feature vector \mathcal{F} . Note that it captures not only object appearance and movement, but also the information relating to the sensing scenario, such as how far away

Table 7.1: Feature vector of the compressed segment

Feature Name	Description
Key Points	x, y position of the points in the LIDAR coordinates; Intensity values of these points (if intensity values are provided)
Points in between key points	Number of points in between a pair of key points; variances of distances from these points to their key point lines;
Range distances to background	Range differences between two extreme points to their neighboring background points.

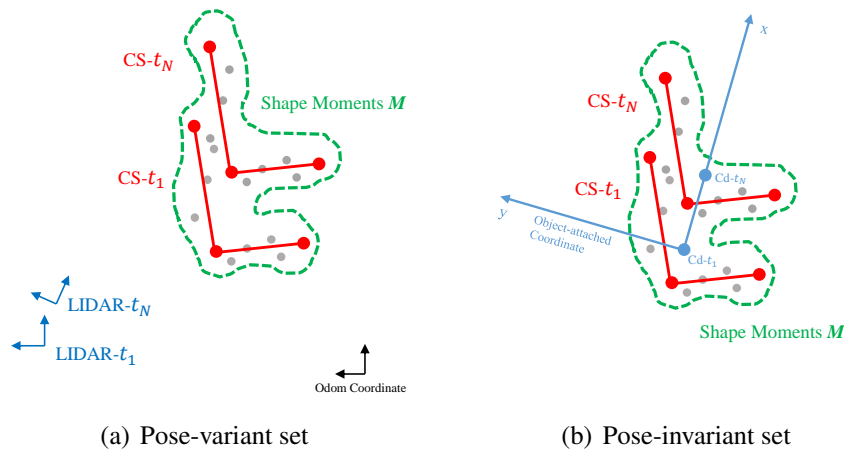


Figure 7.4: Pose-variant and Pose-invariant feature sets

the object is and at what angle. The scenario information is important for multiple reasons. First, the distance to the object affects the number of laser points cast on it, due to LIDAR’s limited angular resolution and detection range. Second, the observation angle on the object determines the measurements, e.g., the side of an object may be occluded when observed from the front. The importance of scenario information for object recognition is demonstrated by the experiment results described in Section 7.4.

Because the compressed scan \hat{z}_{t_i} is defined in the sensor coordinates LIDAR- t_i , \mathcal{F} is pose-variant and a large number of training instances may be needed to cover different sensing situations. For this reason, we also propose a pose-invariant feature vector, where the compressed scan \hat{z}_{t_i} is transformed into an object-attached coordinate, as shown in Figure 7.4(b). Denoting the centroid of LIDAR segment z_{t_i} as Cd_{t_i} , the origin of the object-attached coordinate is defined to be Cd_{t_1} , with its x axis pointing from Cd_{t_1} to Cd_{t_N} . Compared to the pose-variant feature vector, the pose-invariant vector is more general in terms of object positions and orientations, but at the cost of losing scenario information.

7.3 Experiments

The objective of our experiments is three-fold. First, we seek to validate that accumulated scans will result in higher accuracies compared to single scans. Second, we attempt to better understand the effect the length of the time-window has on classification accuracy. Third, we seek to analyze the performance of our designed spatial-temporal



Figure 7.5: Testbed for moving object recognition

features.

Our test bed is a converted iMiev with a 2D LIDAR (SICK LMS 151) mounted on the front of the vehicle, as shown by Figure 7.5. The LIDAR runs at 50 Hz, with 270° FOV. The entire system is developed using the Robot Operating System (ROS) [149]. To test the performance of our algorithm, we conduct experiments in two different environments: a university campus and a highway. The former is a cluttered environment with average vehicle speeds of 10–30 km/h, while the highway is a more “structured” environment with vehicle speeds of 60–100 km/h. In this experiment, we focus on recognizing moving vehicles, but our algorithm is applicable to general-purpose moving object recognition. Ground truths of moving vehicles are obtained via manual labeling for both environments, with 232 positive vehicle samples labeled for the campus environment and 1212 positive samples for the highway one. Note that negative samples are also manually labeled in the experiments, the numbers of which change with the temporal window lengths, as will be shown in the next section.

7.4 Results

We evaluate our algorithm using five different metrics: segmentation ratios, classification accuracy, spatial analysis, performance of different feature sets, and the computational cost. Note that all the analyses are performed with the pose-variant feature vector, except where performances of different feature sets are studied. Major insights of the experimental results will be summarized at the end of this section. More details of our

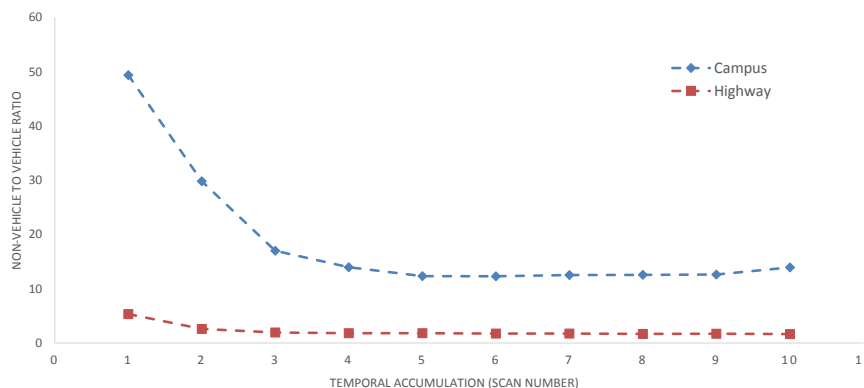


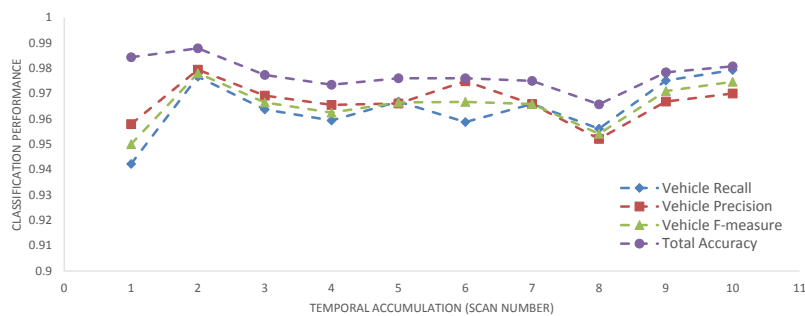
Figure 7.6: Non-vehicle to vehicle ratios in different environments.

experiments can be found in Video (10)(11) in Appendix B.

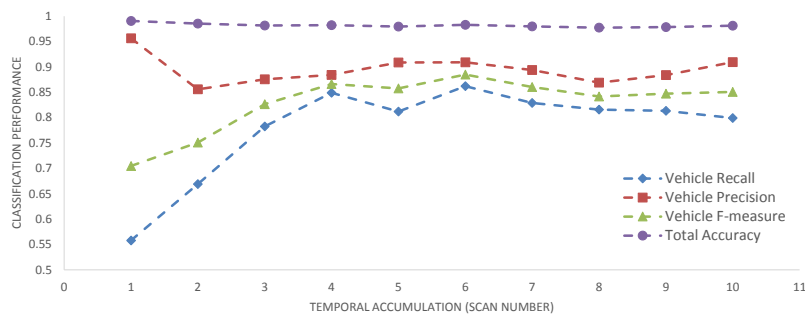
Segmentation Ratios: Figure 7.6 shows the ratios of background clusters to vehicle clusters. Compared to the highway environment, the campus environment is far more cluttered, resulting in a larger number of background clusters. However, the number of background clusters decreases drastically as the number of accumulated scans is increased. This suggests that the temporal accumulation prevents the background from being over-segmented, which as we will see, leads to an improvement in classification accuracy.

Classification Accuracy: Figure 7.7 shows the classification results for moving vehicle recognition under 5-fold cross validation. Note that the classification problem here is a unbalanced binary classification problem, and the number of background clusters varies with the accumulated scan number. For the above reasons, while apparently good total accuracies ($> 97\%$) are achieved in both environments, we analyze the precision and recall rates to better evaluate our algorithm: precision measures what fraction of the detections are actually moving vehicles, and recall measures what fraction of the actual moving vehicles are detected [150].

In the clean highway environment, both the precision and recall rates are high ($> 94\%$) even when using only single scan segments. With a temporal window length larger than two, the SVM attains performances above 97% . In the cluttered campus environment, vehicle detection appears more challenging. However, we observe that it is in this environment that our approach yields the most positive effect. While the precision remains decent ($> 85\%$), the recall rate using no temporal windowing is at a low 55% . The recall rate rises rapidly with the temporal accumulation from 55% to a high of 86%



(a) Highway environment



(b) Campus environment

Figure 7.7: Classification at different environments

(at $N = 6$). The F-measure (F_1 -score) shows a weighted average of the precision and recall, where our algorithm achieves best performance at $N = 6$. As N continues to grow, the performance seems to tail off. We believe this occurs due to the “curse of dimensionality”, which hampers the classification process as the feature vector length grows.

Figure 7.8 shows examples of moving vehicle detection in the two environments ($N = 2$ for highway, and $N = 6$ for campus). The top row shows the images captured from an on-board camera, which is calibrated with the LIDAR sensor, and the bottom row shows the recognition results from the accumulated LIDAR data. Temporal accumulation and ST segmentation are performed to extract individual ST-clusters (shown in different colors), which are then classified to extract the moving vehicles (shown as red blobs). The results are also projected into the camera image for visualization purpose. Since the camera has a much smaller field of view ($\approx 70^\circ$) compared to the LIDAR, some of the results are not shown in the image. Note that a minor misalignment exists between the camera images and the LIDAR data, which is attributed to the time difference between the laser points (accumulated in the past) and the captured image.

From the two examples it is easily observed that the campus environment is much

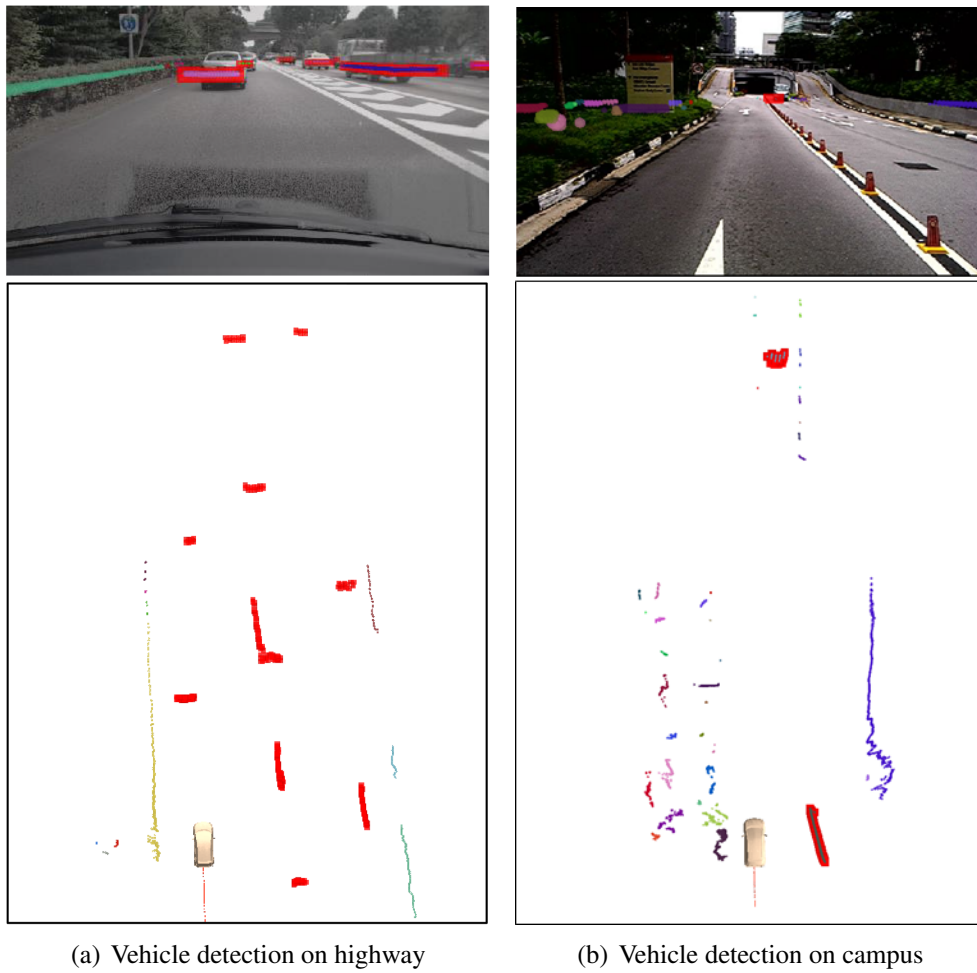


Figure 7.8: Moving vehicle detection examples

more cluttered than the highway. In the highway example, the road barrier and bushes at the two sides are generally neat and consistent, making it relatively easy to differentiate the foreground objects and the background noise. There are 20 clusters extracted in the shown case, with 10 of them recognized as moving vehicles. Compared to the clean highway scenario, the campus environment is much more “dirty”: its background usually consists of various unconnected objects, and the bumpiness of the ground may also cause noise when the LIDAR scans strike the road surface. Given all these challenges, our algorithm is still able to perform robust recognition: 2 moving objects are correctly identified from the 37 extracted clusters in the shown case.

Spatial Analysis: Figure 7.9 presents us with more insights into the performance of our algorithm from a spatial perspective ($N = 2$ for highway, and $N = 6$ for campus). Since our test locations are left-hand drive, many of the vehicle samples in our collected training data are at the front and right sides of the iMiev. In Figure 7.9(b), we see that high vehicle detection errors occur at the boundary of the LIDAR FOV, where only parts of the clusters are observed. Other errors take place over > 20 meters away from the LIDAR center. We posit that this is due to the fact that when observing objects from a distance, the LIDAR readings are occluded by other objects or missing due to low reflectivity. Importantly, the results show that in the vicinity of the LIDAR, the detection accuracy is nearly 100%, which is essential for safe navigation.

Different Feature Sets: In our research, we compare the performance of our designed features with existing feature sets proposed in the literature. Together with our designed pose-variant and pose-invariant features, we include three more feature sets: Ensemble of Shape Functions (ESF) [151], Viewpoint Feature Histogram (VFH) [152], and Ultrafast Shape Recognition (USR) [153]. Unlike our feature sets extracted from compressed scan segments, these three methods operate on 3-D spatial data. Here, 3-D data are constructed by shifting the accumulated points (point set D in Section 7.2.2) in the z direction (the shifted distance is proportional to the elapsed time from when they are received to the latest time).

To assess the performances of different feature sets, the same temporal window length is used, with $N = 2$ for highway and $N = 6$ for campus. Our results are shown in Table 7.2. It is observed that the pose-variant and pose-invariant features outperform the 3D feature sets, which are designed for dense 3D data and appear unsuitable for

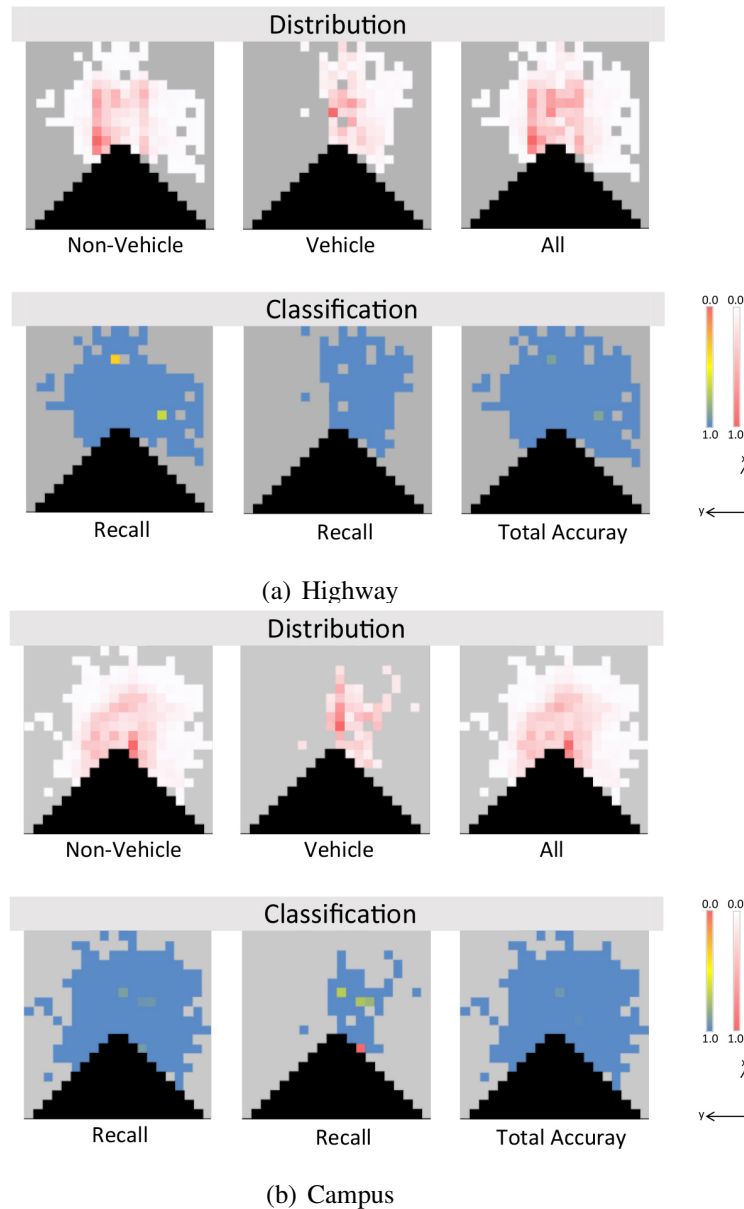


Figure 7.9: Overall vehicle detection performance. The center of each plot is the LIDAR origin, with LIDAR orientation shown by the legend. Each pixel in the figures represents a $5 \times 5 m$ grid place. The grey pixels are places where insufficient samples were collected, and the dark areas are places beyond the LIDAR FOV. In the distribution plots, the density value of each cell represents the number of collected samples in this place, which is normalized by the largest value.

Table 7.2: Classification using different feature sets

(a) Highway environment with $N=2$

FeatureSets	vehicle recall (%)	vehicle precision (%)	vehicle F-measure (%)	total accuracy (%)
Pose-Variant	95.87	97.48	96.67	97.61
Pose-Invariant	94.68	94.50	94.59	96.87
ESF	73.86	94.39	82.87	91.17
VFH	50.42	87.80	64.06	83.63
USR	82.45	79.48	80.93	88.77

(b) Campus environment with $N=6$

FeatureSets	vehicle recall (%)	vehicle precision (%)	vehicle F-measure (%)	total accuracy (%)
Pose-Variant	86.21	90.91	88.50	98.32
Pose-Invariant	70.96	89.35	79.10	96.70
ESF	50.37	77.40	61.02	94.33
VFH	24.63	82.72	37.96	92.90
USR	37.50	70.34	48.92	93.10

the ST data accumulated from the LIDAR. The better performance of pose-variant over pose-invariant features indicates the usefulness of the scenario information as discussed in Section 7.2.3.

Computational Cost: On our experimental platform (computer equipped with a Core i7-4770 processor), the computational time required to process one new scan is $5 \sim 10$ ms for the campus environment with $N = 6$. In the highway environment with $N = 2$, the processing time is only $1 \sim 4$ ms. In short, computational costs are low, making our method suitable for real-time applications.

Summary: From the proceeding discussion, three major insights can be derived from our experimental results:

1. Accumulation in the temporal domain helps to prevent over-segmentation of sensor data in the cluttered environment (Figure 7.6).
2. The spatial-temporal features enable more accurate classification compared to using only spatial features from a single measurement (Figure 7.7 and Figure 7.9).
3. Increased size of the accumulation time window improves recognition accuracy in the cluttered environment up to a maximum time window (Figure 7.7).

7.5 Summary

In this chapter, we propose and investigate a novel spatial-temporal approach for moving object recognition with a single 2D LIDAR. By using crafted spatial-temporal features, we obtain promising classification results in two different experimental settings. Our results suggest that our approach is particularly applicable in cluttered environments, where temporal windowing prevents over-segmentation of the observations and the accumulation of sensor information makes moving object recognition more accurate. As future work, we plan to investigate the performance of our approach on other classes of moving objects (e.g., pedestrians and motorcycles).

The contributions of our work include two parts. Firstly, we develop a spatial-temporal approach for moving object recognition using a 2D planar LIDAR, which achieves equivalent/better performance than the state-of-art algorithm [154] (although different databases are used, their experiment is carried out in a similar campus environment, and our algorithm achieves equivalent or better performance both qualitatively and quantitatively). Secondly, we design two sets of spatial-temporal feature specifically for the accumulated 2D laser scans, which outperform the existing 3D feature sets and achieve good recognition results.

Chapter 8

From Human Activity Learning to Semantic Mapping

For vehicle autonomous navigation in the urban environment, object recognition functions are developed to detect and identify different objects of interests, as discussed in the previous chapters. The detection is performed online in the local neighborhood of the own vehicle, to help it handle various momentary conditions reactively. While the detection is generally short-term and object-oriented, some long-term understanding of the environment can be gained from these temporal detections. Environment understanding targets acquiring long-term knowledge about the environment. The knowledge, also called the model, serves as the prior information for robot autonomous navigation, which can be used to help vehicle localization, path planning, and all the other purposes. The knowledge of an environment can be about any dimension of its properties, such as its spatial layout, its temperature, the types of objects placed in it, etc. In our research, three dimensions of knowledge important for autonomous navigation are identified: the **metric dimension** about its geometric layout, the **semantic dimension** about the semantic meanings of different places, and the **activity dimension** about the activity patterns of human agents living in it.

In traditional robotics studies, enormous efforts have been spent on the problem of metric mapping to derive a metric model of the environment. The output of metric mapping is a metric model of the environment, which captures its geometric layout and is mainly used for the localization purpose. Nowadays researchers are trying to augment traditional metric maps with more high-level knowledge, such as semantic information

and activity information, to help the robot to really understand its environment. This chapter introduces our research on semantic mapping and activity learning.

8.1 Introduction

Semantic mapping has become a popular research topic in recent years. By augmenting traditional metric/topological maps with higher-level semantic knowledge, researchers aim to help robots to really “understand” their environments. A semantic map can not only facilitate human-robot interaction, but also help a robot perform advanced reasoning and planning. In the past few years, various methods have been proposed for semantic mapping. Depending on the sources of semantic information, these methods can be roughly classified into three categories: appearance-based approach, object-based approach, and activity-based approach, as discussed in Section 2.4.

In our research, we present a semantic mapping method based on pedestrian activity patterns in the urban road environment. While an environment serves as the space for different agents to conduct different activities, it can be divided into different functional areas, with each area corresponding to certain types of activities. For this reason, we can infer the semantic meaning of an area from its associated activity information. The activity information of a place should be another important dimension of information, together with the metric information and the semantic information. The metric dimension of a place usually describes some geometric shapes or occupancy information, the semantic dimension denotes its meaning, and the activity dimension describes the agent behaviors in it. Our philosophy is that these three dimensions are highly correlated, and can be inferred from each other.

In our specific application, we want to recognize different functional areas for pedestrians in the urban road environment, i.e., “pedestrian path”, “entrance/exit”, “crossing” and “sidewalk”. By observing pedestrian activity over time, the semantic properties of a place can be inferred from the learned motion patterns in it. Without loss of generality, we focus on motion patterns as the key features of pedestrian activity representation. A pedestrian activity model of the environment is first learned, and then 2D grid semantic mapping is performed by classifying the semantic properties of each cell using the learned activity model. Our proposed method is tested through experiments, and has

shown good performance. In general, our method can be extended beyond pedestrian activities to vehicles, cyclists, or other agents in the outdoor environment.

The contribution of this chapter is clear: to our knowledge, it is the first time to propose the idea of semantic mapping by learning agents' spatial activity models, especially with a mobile platform. The remainder of this chapter is organized as follows. Section 8.2 gives a brief overview of our system. Section 8.3 describes pedestrian activity learning. In Section 8.4, we introduce our algorithm for semantic mapping from pedestrian activity. Experimental results and analysis are presented in Section 8.5. Finally, Section 8.6 concludes the chapter.

8.2 System Overview

8.2.1 Multi-dimensional Grid Map

In the field of metric mapping, the Occupancy Grid Map (OGM) is one of the most popular representations [16]. It represents the environment by an evenly spaced grid, with each cell corresponding to a variable of occupancy to be estimated. In this work, we extend the idea of OGM to a multi-dimensional grid map, where each cell has multiple dimensions of information. The multi-dimensional grid map can be formulated as follows: $M = \{m_{ij} | 0 \leq i \leq w - 1, 0 \leq j \leq h - 1\}$, $m_{ij} = (\mathfrak{M}_{ij}, \mathfrak{S}_{ij}, \mathfrak{A}_{ij})^T$. M denotes the map, m_{ij} the grid cell indexed by i and j , which is composed of multiple dimensions of information: metric information \mathfrak{M}_{ij} , semantic information \mathfrak{S}_{ij} , and activity information \mathfrak{A}_{ij} . The width and height of the map are denoted by w and h respectively. These different dimensions of information are correlated, and can be inferred from each other: knowing the metric property of a place will help to infer its semantic meaning, and vice versa; the semantic meaning of a place may help the robot to infer its normal agent activity, and vice versa; etc. In our application, we want to infer the semantic dimension of information from the activity dimension, as shown in Figure 8.1.

8.2.2 System Framework

We want to realize semantic mapping from learning pedestrian activity in the urban road environment, with a mobile platform of an autonomous vehicle. The system framework

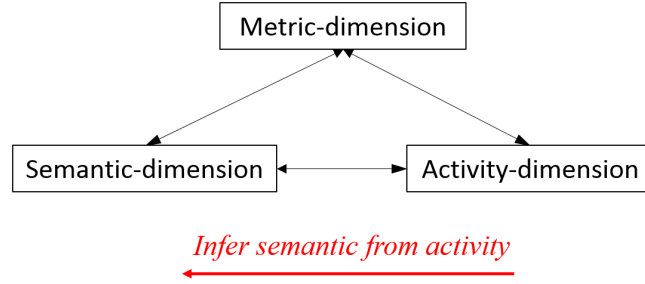


Figure 8.1: Correlated multiple dimensions of information

is illustrated by Figure 8.2. Firstly, pedestrians are detected and tracked using on-board sensors, and the collected tracks are then transformed into the global map frame using the vehicle localization function. Secondly, track classification and clustering is performed. Thirdly, activity information from moving tracks is registered into the grid map, and then a pedestrian activity model is learned. Finally, the semantic information \mathfrak{S}_{ij} is inferred from the learned activity pattern \mathfrak{A}_{ij} , together with prior road network information.

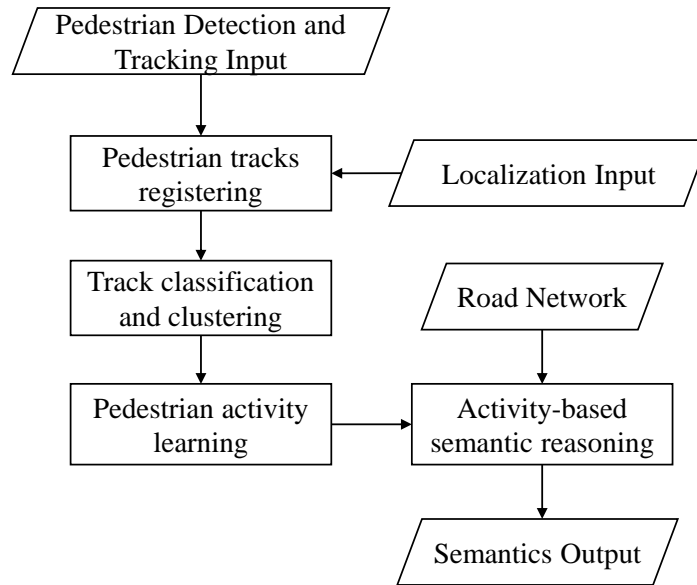


Figure 8.2: Flowchart of semantic mapping from activity learning

8.3 Pedestrian Activity Learning

This section presents our method of pedestrian activity learning from a mobile platform. We will learn the activity information of each cell \mathfrak{A}_{ij} from collected pedestrian tracks.

Activity model learning is not a new topic in the computer vision community, where researchers have proposed various methods to learn pedestrian motion patterns. Some representative work can be found in [118, 119]. However, most of the algorithms use a stationary camera and assume the observability of complete trajectories, which is not a valid assumption for applications using mobile robots. A. Lookingbill et al. in [120] use a helicopter to identify moving objects on the ground and learn their motion patterns. This work shows interesting results and enlightens us about representing motion patterns in the form of grid map. However, it only estimates the motion patterns of cells where moving objects are observed, and it also neglects the relationship between neighboring places. In our work, we use Gaussian Processes to learn the activity model of the entire environment, which is able to overcome the above problems.

In this section, we will first discuss the acquisition of pedestrian tracks, then introduce the classification and clustering of tracks, and finally present the GP-based motion model learning.

8.3.1 Pedestrian Detection and Tracking

Pedestrian detection and tracking is one fundamental function for pedestrian activity learning, which is performed using onboard sensors. A laser range finder is used for pedestrian hypothesis generation and tracking, and a camera is used for hypothesis verification. For more details, please refer to our previous work in [155]. The output tracks are sequences of pedestrian positions with time stamps, from which moving speed and direction can also be calculated. While pedestrians are initially detected in the local coordinates of the vehicle, we transform the track information into the global frame of the map, using the vehicle localization function.

8.3.2 Track Classification and Clustering

Before using the collected tracks for pedestrian activity learning, track classification and clustering should be applied. Due to the noise in the pedestrian detection and tracking, plus the noise incurred by localization error during track transformation, the motion of some tracks may be very unstable, or they are simply not tracks for pedestrians. In some other cases, static tracks may appear when pedestrians stand still for long time in some places. These tracks are not useful for pedestrian activity (dynamic) learning, and

should be filtered out. A classification process is used to classify the tracks into three types, “moving”, “static”, and “noisy”. The classification is based on several features of the track, such as track length, moving speed, etc. Only “moving” tracks will be used for the activity learning purpose.

Track clustering is performed to cluster heterogeneous tracks into different homogeneous groups. In related research from the computer vision community, pedestrian tracks are usually carefully clustered into multiple groups of high similarity. In our work, however, the mobile platform works in a fairly large area and may collect pedestrians from many heterogeneous motion types. Performing careful clustering and learning the activity model for each of these types are computationally expensive or even infeasible. On the other hand, since our interest is the activity patterns at individual places, rather than those of the complete pedestrian trajectory spanning in the temporal domain, there is hence no need to perform such clustering and learning.

In fact, from a microscopic point of view, for an individual place in the urban road environment, there are usually only two dominant motion patterns of pedestrians, which have similar speed but opposite directions. We denote this assumption as the “bidirectional property” of pedestrian activity. While this assumption appears arbitrary at first glance, it generally holds true for the urban road environment, where pedestrians walk either along or across the road links. This “bidirectional property” simplifies our clustering problem: we cluster the moving tracks scattered over the map into two groups, and only need to guarantee that the activity of each group is consistent at the microscopic cell level.

Our clustering algorithm can be formulated as follows. The set of pedestrian tracks is denoted as $S = \{s_1, \dots, s_m\}$. One track s is a set of position-speed-angle tuples: $s = \{t_1, \dots, t_n\}$, $t_i = \langle x_i, y_i, v_i, \theta_i \rangle$, where x_i, y_i are pedestrian positions, v_i the speed, and θ_i the moving direction. The input of the clustering is S , and the output is two clusters of tracks A and B. During the clustering process, each cluster will maintain a set of tuples as its characteristic quality, which is an assembly of the tuples from all its member tracks. The two tuple sets are denoted as α and β respectively.

The similarity between two tuples p, q is defined as:

$$sim_{p,q} = \frac{1 - 2|\theta_p - \theta_q|/\pi}{\|x_p - x_q, y_p - y_q\| + const.} \quad (8.1)$$

The similarity score between a track s and a cluster with tuple set γ is defined as:

$$SIM_{s,\gamma} = \sum_i^n (max_{p \in \gamma} sim_{t_i,p} + min_{q \in \gamma} sim_{t_i,q}) \quad (8.2)$$

During the clustering process, the longest track is first picked out as the seed track for cluster A, and its tuples form the tuple set α . Then the track having the minimum similarity value with α is selected as the seed track for cluster B, whose tuples then form the tuple set β . The track having the highest similarity score with either cluster A or B is assigned to cluster A or B accordingly, until all the tracks are clustered. The pseudo-code of the cluster algorithm can be found in Algorithm 2.

8.3.3 Activity Learning with Gaussian Process

After the classification and clustering process, we get two clusters of moving tracks. The tracks from the same cluster share similar cell-level motion patterns, which are of interest to us and need to be learned. We use the Gaussian Process (GP) method for this activity learning purpose. To briefly introduce GP, it is a collection of random variables, any finite number of which have (consistent) joint Gaussian distributions. GP can be

Algorithm 2: Pseudo-code for track clustering

Input: The set of pedestrian tracks $S = \{s_1, \dots, s_m\}$

Output: clusters of tracks A and B

```

1 A = B =  $\alpha$  =  $\beta$  =  $\emptyset$ ;
2 Find the longest track  $s_l$ ;
3 Add  $s_l$  to A; Add tuples into  $\alpha$ ; Erase  $s_l$  from  $S$ ;
4 Let  $s_k = \arg \min_{s_k \in S} SIM_{s_k, \alpha}$ ;
5 Add  $s_k$  to B; Add the tuples into  $\beta$ ; Erase  $s_k$  from  $S$ ;
6 while  $S \neq \emptyset$  do
7    $score\_A = \max_{s_p \in S} SIM_{s_p, \alpha}$ ;
8    $s_p = \arg \max_{s_p \in S} SIM_{s_p, \alpha}$ ;
9    $score\_B = \max_{s_q \in S} SIM_{s_q, \beta}$ ;
10   $s_q = \arg \max_{s_q \in S} SIM_{s_q, \beta}$ ;
11  if  $score\_A \geq score\_B$  then
12    | add  $s_p$  to A; add tuples into  $\alpha$ ; erase  $s_p$  from  $S$ ;
13  else
14    | add  $s_q$  to B; add tuples into  $\beta$ ; erase  $s_q$  from  $S$ ;
15  end
16 end
17 return A and B;

```

used to solve both regression and classification problems. Please refer to [155] for more details. We model our activity learning as Gaussian Process Regression (GPR). The set of position-speed-angle tuples for each cluster serves as the observation input, and the prediction output is the information on pedestrian speed v_{ij} and angle θ_{ij} : $\mathfrak{A}_{ij} = \{\bar{v}_{ij}, \sigma_{v_{ij}}^2, \bar{\theta}_{ij}, \sigma_{\theta_{ij}}^2\}^T$.

It should be mentioned that while the speed value can be estimated directly from GPR, it is not suitable to do so for the pedestrian moving angle. Unlike a linear variable distributed in $(-\infty, +\infty)$, the angle variable is a circular variable in $[0, 2\pi)$, whose mean and variance are ‘‘circular mean’’ and ‘‘circular variance’’ to be calculated in different ways. For a simple example, the difference between angle 1° and 359° is actually 2° , rather than 358° as calculated in the linear way. Based on direction statistics [156], we model the angle distribution as a Projected Normal Distribution, which can be calculated from the bivariate normal distribution of the speed vector $\vec{v} = (v_x, v_y)$. In the activity learning process, three separate GPRs will be trained, with one for the scalar speed v , and the other two for speed values in x and y directions v_x, v_y . By assuming the independence of v_x and v_y , we can synthesize the bivariate distribution of the speed vector, from which the distribution of the moving angle can be calculated.

Gaussian Process Regression Model

Let X be the 2-dimensional position vector in the map coordinate, $X \in \mathbb{R}^2$, $X = (x, y)$. Let Y be the output value, $Y \in \mathbb{R}$. Our Gaussian Process Regression model is as follows:

$$Y = F(X) + \xi, \quad F \sim GP(m, K), \quad \xi \sim N(0, \sigma_n^2) \quad (8.3)$$

$F(X)$ is a function distributed as a GP with mean function m and covariance function K . It can be calculated that the output function Y is also distributed as a GP:

$$Y \sim GP(m, K + \sigma_n^2 \sigma_{ii'}) \quad (8.4)$$

where $ii' = 1$ iff $i = i'$. Given a set of training data (X, Y) , the posterior distribution for a set of test points X^* is a Gaussian distribution:

$$Y^*|Y \sim N(m(X^*) + K^T(X, X^*)K^{-1}(X, X)(Y - m(X)), \quad (8.5)$$

$$K(X^*, X^*) - K^T(X, X^*)K^{-1}(X, X)K(X, X^*))$$

In our application, we want to get the posterior distribution for v, v_x, v_y at each test point X_{ij} , where X_{ij} is the position of m_{ij} in the map frame. For this purpose, three separate GPRs are trained, using the tuple set of each cluster as the training data. Zero mean function m and squared exponential covariance function K are used in our GPRs, where $m(X) = 0, K(X, X') = \sigma_y^2 \exp \frac{-(X-X')^2}{2l^2}$. The hyperparameters (σ_n, σ_y, l) are learned by maximizing the log-likelihood of the observation in the training data.

Projected Normal Distribution of Moving Angles

We use a Projected Normal Distribution (PND) to model the probabilistic density function of the pedestrian moving angle. Let \vec{x} be a random two-dimension vector which has a normal distribution $N_2(\mu, \Sigma)$, in which case the angle of \vec{x} is said to have a projected normal (or angular Gaussian) distribution $PN_2(\mu, \Sigma)$. The probabilistic density function of $PN_2(\mu, \Sigma)$ is as follows:

$$p(\theta; \mu, \Sigma) = \frac{\vartheta(\mu; 0, \Sigma) + |\Sigma|^{-\frac{1}{2}} D(\theta) \Phi(D(\theta)) \phi(|\Sigma|^{-\frac{1}{2}} (x^T \Sigma^{-1} x)^{-\frac{1}{2}} \mu \wedge x)}{x^T \Sigma^{-1} x} \quad (8.6)$$

where $\vartheta(\mu; 0, \Sigma)$ denotes the value of the probability density function for $N_2(0, \Sigma)$ at point μ , Φ and ϕ denote the probability density function and cumulative density function of $N(0, 1)$, $x = (\cos\theta, \sin\theta)^T$, $D(\theta) = \frac{\mu^T \Sigma^{-1} x}{(x^T \Sigma^{-1} x)^{-1/2}}$, and $\mu \wedge x = \mu_1 \sin\theta - \mu_2 \cos\theta$ with $\mu = (\mu_1, \mu_2)^T$.

In our application, pedestrian moving direction is actually distributed according to PND. To calculate the distribution, the normal distribution of the pedestrian speed vector \vec{v} is used. It is synthesized using the marginalized distribution of v_x and v_y by assuming their independence: $\vec{v} \sim N(\text{diag}(\mu_{v_x}, \mu_{v_y}), \text{diag}(\sigma_{v_x}^2, \sigma_{v_y}^2))$ With this bivariate normal distribution, the probabilistic density function of the moving angle can be calculated. In our semantic reasoning, the circular mean of the distribution $\bar{\theta}_{ij}$ is adopted as the

pedestrian moving angle, and the circular variance is used to represent the uncertainty of this moving angle $\sigma_{\theta_{ij}}^2$. For the detailed definition and calculation of circular mean and variance, please refer to [156].

Bidirectional Property of Pedestrian Activity

As discussed in Section 8.3.2, we classify the collected pedestrian tracks into 2 clusters, and learn their activity models independently. According to real experiments, two learned activity models are actually like a mirror-pair: the moving direction of one place is actually the opposite direction of the other. This leads us to the assumption that pedestrian activity at a place is often “bidirectional”, which allows us to learn the activity model of one track cluster, and infer the other via rotating its direction by 180° . In our application, we choose to learn the activity model of the first cluster. Track information from the second cluster is also utilized in the activity learning: the angle values in its activity tuples are increased by 180° , and the tuples are used together with the training data from the first cluster.

In the later section of semantic reasoning, we will use the right angle between pedestrian moving direction and road link direction as a feature to infer a place’s semantics. Since this angle difference calculated with either of the two activity models is the same, we will use the first model as the surrogate for both.

8.4 Activity-based Semantic Reasoning

This section introduces our method of activity-based semantic mapping. We want to perform two levels of semantic reasoning, one coarse-level to identify “pedestrian path” (shorthand as “PP”), and one refined-level reasoning to recognize three different types of functional areas from the path, which includes “entrance/exit” (EE), “crossing” (CR), and “sidewalk” (SW). It should be mentioned that these three types of areas are not necessarily mutually exclusive, considering the fact that the same area may serve different purposes at the same time. To capture the semantic properties at place m_{ij} in the map, a semantic vector of four binary variables is introduced: $\mathfrak{S}_{ij} = (p_{ij}, e_{ij}, c_{ij}, s_{ij})^T$, where

- p_{ij} , a binary variable for “path”, $\Lambda_p = \{\text{PP}, \text{non-PP}\}$, $p_{ij} \in \Lambda_p$;
- e_{ij} , for “entrance/exit”, $\Lambda_e = \{\text{EE}, \text{non-EE}\}$, $e_{ij} \in \Lambda_e$;

- c_{ij} , for “crossing”, $\Lambda_c = \{\text{CR}, \text{non-CR}\}$, $c_{ij} \in \Lambda_c$;
- s_{ij} , for “sidewalk”, $\Lambda_s = \{\text{SW}, \text{non-SW}\}$, $s_{ij} \in \Lambda_s$;

The input information of the semantic reasoning process is the activity information \mathfrak{A}_{ij} , and prior road network information.

8.4.1 Pedestrian Path Learning

Pedestrian Intensity

In the urban road environment, there are certain explicit or implicit paths that pedestrians can take. The more pedestrians that pass through a certain place, the higher likelihood for it to be part of pedestrian paths. In other words, the pedestrian number counted in one place can be a useful indicator to distinguish its semantic type. Based on this idea, we introduce a measurement “pedestrian intensity” as a feature for pedestrian path classification. The intensity at place m_{ij} is denoted as I_{ij} , which is a function of pedestrian count N_{ij} :

$$I_{ij} = I_{local_{ij}} \times I_{global_{ij}}; \quad (8.7)$$

$$I_{local_{ij}} = N_{i,j} / \max_{a,b} (N_{i+a,j+b}); \quad (8.8)$$

$$I_{global_{i,j}} = \frac{1}{1 + \exp(-N_{ij} + N_{exp})} - \frac{1}{1 + \exp(N_{exp})}; \quad (8.9)$$

where $a, b \in Z \cap [-l/2, l/2]$. The pedestrian intensity is the multiplication of two factors, the local factor and the global factor, denoted by $I_{local_{ij}}$ and $I_{global_{ij}}$. The local factor is used to normalize the pedestrian count with the maximum values in a $l \times l$ local window. This factor will help to mitigate the problem of data imbalance, which will arise when the observation periods for different areas are too different, leading to the imbalance that pedestrian tracks in some areas are intensively collected while other areas may be overlooked. This local factor has similar effects to the adaptive threshold in image processing, in which it can be used to recover image details when image brightness is unbalanced. The global factor is namely a logistic function of $N_{i,j}$, which increases quickly when $N_{i,j}$ is near N_{exp} , but changes slowly when far away. N_{exp} is a constant value chosen as the expected pedestrian count at a “path” place.

Classification using Markov Random Field (MRF)

Based on pedestrian intensity calculated from the previous step, we use a Markov Random Field (MRF) for path classification. MRF is a popular technique in image processing, which can capture the dependency between neighboring pixels and is widely used for image segmentation, restoration and other purposes. For more details please refer to [157].

We model our classification problem as a pairwise MRF: Given the intensity data $I = \{I_{i,j}\}$, we want to estimate the “path” semantics of the map $P = \{p_{ij}\}$. Let’s assume that $I_{i,j}|p_{ij} \sim N(\mu_{p_{ij}}, \sigma_{p_{ij}})$, where $\mu_{p_{ij}}$ and $\sigma_{p_{ij}}$ can be learned through training data. We get the energy function

$$U = \sum_{ij} \left(\log(\sqrt{2\pi}\sigma_{p_{ij}}) + \frac{(I_{ij} - \mu_{p_{ij}})^2}{2\sigma_{p_{ij}}^2} \right) + \sum_{i,j,h,k} \beta \delta(p_{ij}, p_{hk}) \quad (8.10)$$

where p_{ij} and p_{hk} are “path variables” of neighboring places m_{ij} and m_{hk} , and β is a weighting parameter, $\beta \geq 0$. By minimizing this energy function, the optimal classification for pedestrian paths can be found, denoted as $\hat{P} = \{\widehat{p}_{ij}\}$.

8.4.2 Refined Semantics Learning

After the coarse-level semantic learning for pedestrian paths, we want to perform refined semantic reasoning to learn the functional areas in the path, i.e. ”entrance/exit”, ”crossing”, and ”sidewalk”. We use Naive Bayes Classifiers (NBC) to learn the semantic variables e_{ij} , c_{ij} and s_{ij} . A Naive Bayes Classifier is a simple probabilistic classifier based on Bayes’ theorem assuming independence between features given the class variable. The probability model for a classifier is a conditional model:

$$p(C|F) = p(C|F_1, \dots, F_n) = \frac{1}{Z} p(C) \prod_{i=1}^n p(F_i|C) \quad (8.11)$$

where C is the class variable, F is the feature set $F = \{F_1, \dots, F_n\}$, z a normalizer, $p(C)$ the class prior, and $p(F_i|C)$ the feature model for F_i given class C , $F_i \in F$.

In our application, three different NBCs are built to classify the three types of functional areas separately, denoted as $p(e_{ij}|F)$, $p(c_{ij}|F)$ and $p(s_{ij}|F)$. We use the similar set of features F for the three NBCs, with different feature models. The set of features used

here includes “path property” $F_{pp_{ij}}$, “moving direction” $F_{d_{ij}}$, “direction variance” $F_{dv_{ij}}$, and “position” $F_{p_{ij}}$. $F_{ij} = \{F_{pp_{ij}}, F_{d_{ij}}, F_{dv_{ij}}, F_{p_{ij}}\}$.

- $F_{pp_{ij}}$ is a binary feature, which is actually the classification result \widehat{p}_{ij} from the coarse-level “path” classification. The feature model $p(F_{pp_{ij}}|C)$ is designed to carry the idea that if a place is not a pedestrian path, it is not likely to be some functional area.
- $F_{d_{ij}}$ is about the angle of the pedestrian moving direction. $\bar{\theta}_{ij}$ in \mathfrak{A}_{ij} is chosen as its value. This feature carries the typical motion information at each cell, which is highly related to its semantic meaning.
- $F_{dv_{ij}}$ is about the uncertainty of the learned pedestrian moving angle. $\sigma_{\theta_{ij}}^2$ is chosen as its value. The bigger $F_{dv_{ij}}$ is, the more unreliable the calculated moving direction $F_{d_{ij}}$.
- $F_{p_{ij}}$ is about a place’s position relative to the road network. This feature is introduced with the idea that the functional semantics of a certain place are actually related to its position on the road.

In the rest of this subsection, we will first introduce the prior road information used in the semantic classification process, and then present the classification for each type of semantics. The feature models in different NBCs will be discussed.

Prior Road Information

In our previous work [10], we are able to get two kinds of maps for the road network, one binary grid map and one topo-metric map. The binary grid map denotes the binary status of each place cell, “road” or “non-road”. The topo-metric map is a compact representation for the road network, in which road links are represented by fitted splines. We use above two types of road maps to get the required position information in the semantic reasoning process. For example, based on the binary grid map, road boundary information can be retrieved; based on the topo-metric map, the angle of a road link can be calculated from its spline representation; etc.

Entrance/Exits $p(e_{ij}|F)$

For urban road environments, pedestrian entrance/exits are where pedestrians enter onto and depart from the road. Due to the bidirectional property of pedestrian motion, people usually use the same pathway for entrance as well as exit into a spatial region. The knowledge of pedestrian entrances and exits in a road network is of vital importance and can help an autonomous vehicle's safe navigation. We use Naive Bayes Classifiers to recognize such areas, based on the feature set F_{ij} . The feature models are built as below. (It should be mentioned that the above feature model is just a "simplistic abstract" model, which can have different variants in real applications.)

i) $p(F_{pp_{ij}}|e_{ij})$: The entrance/exits (EE) are functional areas of pedestrians, which should only appear on a pedestrian path. If an area is "EE", it should be "PP". The infinitesimal ϵ is to avoid degenerate cases. If an area is "non-EE", its possibility to be a "PP" is denoted as k_{ee} , which is approximated by the ratio of extracted "PP" area over the road surface region. It should be mentioned that the same feature models are chosen for the other two semantic properties c_{ij} and s_{ij} , except that different parameters k_{cr} and k_{sw} are used for k_{ee} .

$$\begin{aligned} p(F_{pp_{ij}} = \text{PP} \quad |e_{ij} = \text{EE}) &= 1.0 - \epsilon; \\ p(F_{pp_{ij}} = \text{non-PP} \quad |e_{ij} = \text{EE}) &= \epsilon; \\ p(F_{pp_{ij}} = \text{PP} \quad |e_{ij} = \text{non-EE}) &= k_{ee}; \\ p(F_{pp_{ij}} = \text{non-PP} \quad |e_{ij} = \text{non-EE}) &= 1.0 - k_{ee}; \end{aligned}$$

ii) $p(F_{d_{ij}}|e_{ij})$: When a pedestrian enters or leaves a road link, its moving direction is usually perpendicular to the road direction. This basic idea is reflected in the feature model, where $F_{d_{ij}}$ is the pedestrian moving direction, rd_{ij} is the direction of the nearest road link calculated from its spline representation, and $\Delta(,)$ is the function to find the right angle between these two directions.

$$\begin{aligned} p(F_{d_{ij}}|e_{ij} = \text{EE}) &= \frac{4}{\pi} \Delta(F_{d_{ij}}, rd_{ij}); \\ p(F_{d_{ij}}|e_{ij} = \text{non-EE}) &= \frac{2}{\pi}; \end{aligned}$$

iii) $p(F_{dv_{ij}}|e_{ij})$: The feature of angle variance is used to carry the uncertainty of the moving direction estimation. This feature model is the same for the other two NBCs.

$$p(F_{dv_{ij}}|e_{ij} = \text{EE}) = \frac{2(\max_{i,j} F_{dv_{ij}} - F_{dv_{ij}})}{(\max_{i,j} F_{dv_{ij}} - \min_{i,j} F_{dv_{ij}})^2}$$

$$p(F_{dv_{ij}}|e_{ij} = \text{non-EE}) = \frac{1.0}{\max_{i,j} F_{dv_{ij}} - \min_{i,j} F_{dv_{ij}}}$$

iv) $p(F_{p_{ij}}|e_{ij})$: Pedestrian entrances/exits should appear nearby the road boundary. $F_{p_{ij}}$ denotes a place's distance to the boundary of the road, and EE_r is a fixed parameter to control the probability. For an area that is “non-EE”, the probability density function of $F_{p_{ij}}$ is assumed to be a uniform distribution over $[0, \frac{\text{road_width}_{ij}}{2.0}]$.

$$p(F_{p_{ij}}|e_{ij} = \text{EE}) = 1.0 - \frac{F_{p_{ij}}}{\text{EE}_r}$$

$$p(F_{p_{ij}}|e_{ij} = \text{non-EE}) = \frac{2.0}{\text{road_width}_{ij}}$$

With the Naive Bayes Classifier, we can get the “EE” probability of a place. The place with $p(e_{ij} = \text{EE}|F) > 0.5$ is classified as an EE cell. However, these results are in the format of individual cells, and we want to further cluster them into individual EE objects. A Gaussian Mixture Model (GMM) is used for the clustering purpose, with which EE cells of places are clustered as EE objects. Each EE object corresponds to a 2D position in the map. The Bayes Information Criterion (BIC) is used to select the best cluster number. After the clustering, we get a set of EE objects $\xi\xi = \{\text{EE}_1, \dots, \text{EE}_n\}$.

Crossing $p(c_{ij}|F)$

A pedestrian crossing is where pedestrians move across the road. A “crossing” place should be part of the pedestrian “path”. From the activity view, pedestrian moving direction should be perpendicular to the road direction. From the position view, the sum of the distances to its two nearest entrance/exits should be around the road width. With these ideas, the feature models can be built. With the NBC, we are able to learn a place's semantic property of “crossing”. While we can get the classification results directly from NBC, the results may not be smooth in neighbouring areas. In our application, we treat

a place’s probability of “crossing” as a feature, and input it into our MRF framework, to generate better classification results.

Sidewalk $p(s_{ij}|F)$

A sidewalk is a place where pedestrians walk alongside the road. It should appear near the road boundary, and pedestrian moving direction should be parallel to the road direction. Given these ideas, the feature models of $p(s_{ij}|F)$ can be built. To generate smooth classification results, a MRF is used to generate more homogeneous results for s_{ij} , as for c_{ij} discussed previously.

8.5 Experiments

8.5.1 Experiment Setup

Our test bed is a Yamaha G22E golf cart with autonomous driving ability, as shown by Figure 4.4. The pedestrian detection and tracking are performed using a 4-layer LIDAR (SICK LD-MRS400001) mounted at waist height, and a simple webcam above it. Vehicle localization is performed using a tilted-down LIDAR (SICK LMS-291) at the upper front, together with the vehicle’s odometry system. Our experiment environment is the Engineering Campus of the National University of Singapore, where pedestrian activities in two typical areas (“Area MCD” and “Area CR”) are observed and collected, as shown in Fig 8.3(a).

A multi-dimensional grid map is built to cover the Engineering Campus, whose size is 1099×973 cells, with its resolution 0.5 m/cell. For visualization purposes, we are showing the complete map, but highlight the activity learning and semantic mapping results of the two interesting regions. The size of “Area MCD” is 338×100 cells, and “Area CR” is 90×90 cells.

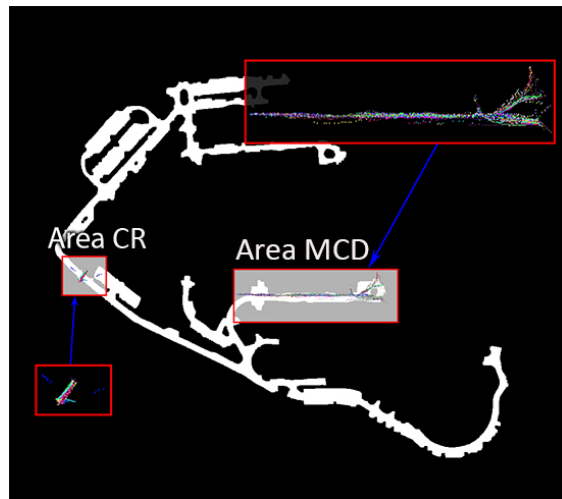
8.5.2 Experiment Results

Pedestrian Activity Learning

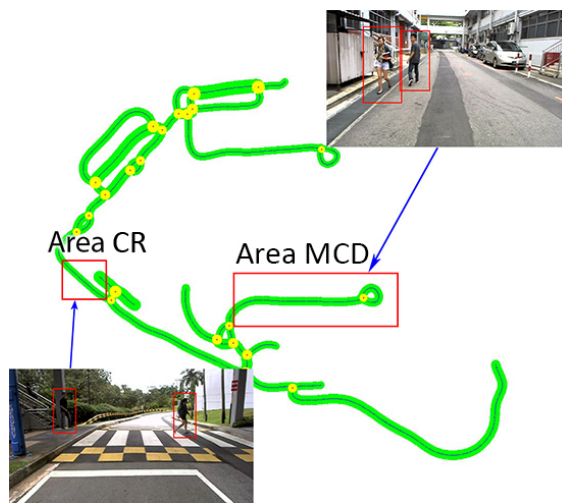
There are 306 tracks collected in our experiment, as shown in Figure 8.3. Figure 8.3(a) shows the satellite image of our experiment environment, where the two areas of interest



(a) Satellite image



(b) Binary road map



(c) Topo-metric graph

Figure 8.3: Experiment environment and road network information (zoom in when read)

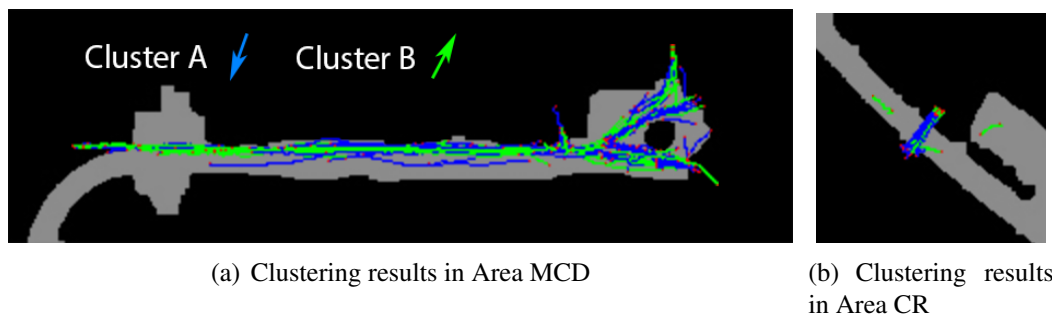


Figure 8.4: Track clustering results

are highlighted. Collected pedestrian tracks are also overlaid in the picture, drawn in different colors. Figure 8.3(b) shows a binary road image from our previous work [9], where white areas are road surface. Pedestrian tracks are overlaid on it. Figure 8.3(c) shows the topo-metric graph of the road network, with two sub-images showing the pedestrian detection results in the two areas.

The track classification and clustering results are summarized as follows: moving tracks number 205, with 100 in cluster A and 105 in cluster B; static tracks number 10; noisy tracks number 91. It can be seen that static tracks are only a small portion of the track set, meaning that pedestrians in the two surveyed areas are mostly in movement. The relatively large number of “noisy tracks” is due to our strict criteria of track classification, which help us to get reliable moving tracks. For the moving tracks, the similar sizes of the two clusters incidentally show the “bidirectional property” of pedestrian activity. Figure 8.4 visualizes the results of moving track clustering, where cluster A and cluster B are colored in blue and green respectively, and red dots are their end points. Tracks in cluster A generally move from right to left, up to down, where tracks in cluster B takes the opposite direction. The clustering results are checked manually and no errors are found. (An error here denotes the case where a pedestrian track moves in one direction is falsely grouped into the opposite cluster.)

Given the results from the track classification and clustering, we try to learn the activity model using Gaussian Processes. As discussed in Section 8.3.3, we only need to learn the activity model in one direction. In this experiment, we learn the activity model in the direction of cluster A. Figure 8.5 illustrates the pedestrian moving direction $\bar{\theta}_{ij}$ of the learned activity model. The direction values are shown by red arrows, which are overlaid onto the satellite image for visualization. We can have an overview of the pedestrian motion flow in the environment from this figure.

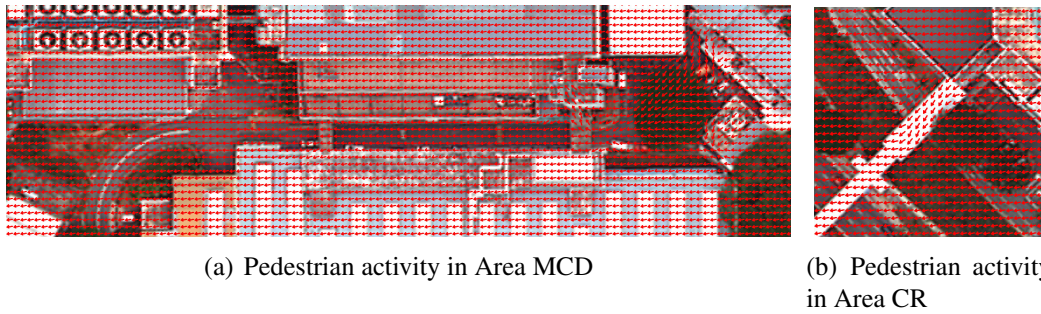


Figure 8.5: Moving direction of activity model (zoom in when read)

Activity-based Semantic Mapping

Together with the road network information, semantic mapping can be performed. Table 8.1 shows the mapping results for semantic properties of the four types.

For the “path” property, it can be seen that our defined feature of “pedestrian intensity” is able to boost the path trunk which most people take and depress any erratic tracks. The classification results from the MRF show a complete path and no false positives. For the “entrance/exit” property, the output probability of NBC is shown by a grayscale image, which is overlaid on the satellite image for visualization. The classification results that we get from the NBC are individual “EE cells”. We use the GMM technique to cluster these cells, and recognize the “EE objects”. The best cluster number is selected automatically with BIC, and finally we recognize the 7 entrances/exits in the two areas. This result is a perfect result according to our ground truth. For the “crossing” property, we are able to recognize the important crossing area in Area CR. However, some cells in Area MCD (no crossing exists) are misclassified as “crossing”. The accuracy for the classification result is 80.3%. This number can be further improved by filtering out those small pieces of areas according to their size. For the “sidewalk” property, we recognize a long sidewalk in Area MCD, which has several disconnected pieces at the right end. According to our definition of pedestrian sidewalk, these disconnected pieces do have a “sidewalk” property. They can be filtered out according to their sizes if our purpose is to find individual “sidewalk objects” rather than “sidewalk cells”. In summary, our activity-based semantic mapping provides promising results. The four types of semantic properties are mapped well in our two survey areas.



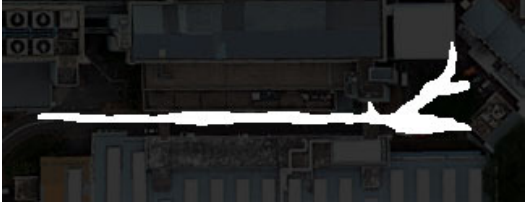




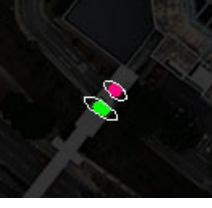

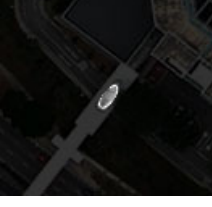

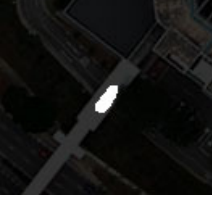

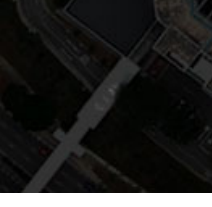
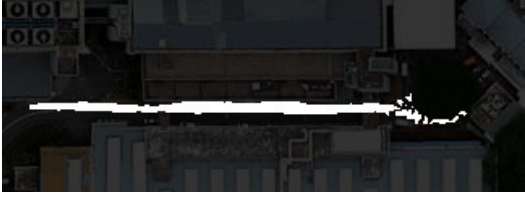
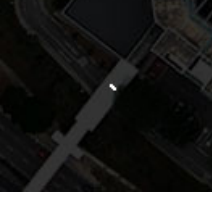
	Area MCD	Area CR
Pedestrian intensity		
Path classification		
EE probabiliy		
EE objects		
CR probability		
CR classification		
SW probability		
SW classification		

Table 8.1: Mapping results for semantic properties of the four types

8.6 Summary

In this chapter, we propose a novel semantic mapping method based on pedestrian activity in the urban road environment. Pedestrians are detected and tracked using an autonomous vehicle, and the collected track information is used to learn the pedestrian activity model in the environment. Based on the learned pedestrian activity patterns and prior road network information, semantic mapping is performed. Our work is tested through real experiments, and shows promising results.

The contribution of our work is that we propose the novel idea of semantic mapping via pedestrian activity learning. Unlike existing approaches which usually solve the environment understanding problem of different dimensions independently, we utilize the correlations between them and demonstrate the feasibility of learning knowledge from each other.

To detect and track pedestrians using a mobile platform is not as convenient as using a surveillance camera. In this work, only 306 tracks are collected to test our method. In the future work, we will test our method with more experiments in different road scenarios. Besides pedestrians, there are other equally important agents moving on urban roads, such as vehicles and motorbikes. In our future work, we will extend our method to other types of agents, and learn richer semantic information from their behaviors.

Chapter 9

Conclusions and Future Work

9.1 Conclusions

This thesis focuses on developing the perception functions for vehicle autonomous driving in the urban road environment. Fundamental perception requirements are identified through literature reviews. We demonstrate that with the minimal sensing configuration, our algorithms are able to achieve equivalent or better performance compared to the existing work.

9.1.1 A Brief Review

We first study the history and current status of autonomous vehicle technology, and summarize the important perception requirements for autonomous navigation in the urban road environment. Three fundamental perception tasks are studied in detail, including localization, object recognition, and environment understanding. Our research around these topics are the main body of this thesis.

To address the problem of vehicle localization, we manage to utilize the typical features in the urban road environment for pose estimation, with only a tilted-down 2D LIDAR and the odometry system. In the first stage of our research, curb-intersection features are extracted to localize the vehicle. Compared to existing approaches that purely use curb features for vehicle localization [43] [44] [45] [46], we introduce an “intersection feature” as a complement, and utilize the combined curb-intersection feature for better localization. Our algorithm achieves accurate estimation results in both the lateral and the longitudinal directions. However, the curb-intersection-based algorithm

only applies to roads where curbs exist, and may not be general enough for all the urban road scenarios. For this reason, in the second stage of the research, we consider incorporating other urban features for localization. Since the urban environment is composed of artificial objects or structures which usually have vertical surfaces, we try to utilize these vertical surfaces as the localization features. Compared to the curb-intersection algorithm, the “vertical surface” algorithm is applicable to general urban road scenarios (with/without curbs), and has better localization accuracy. Our method achieves equivalent performance to the state-of-art algorithm using a 3D LIDAR [15], however, with the much reduced sensing ability of a 2D laser sensor.

Problems of object recognition are also studied in this thesis. While object recognition is a broad research topic, our attention is focused on two specific tasks that are more relevant to vehicle autonomous driving, i.e., road detection and moving object recognition. For the task of road detection, we investigate two categories of research, i.e., road marking detection using vision, and road surface-boundary detection using LIDAR. For vision-based marking detection, we propose a general framework for the detection and analysis of various types of markings. For LIDAR-based surface-boundary detection, we introduce the idea of a rolling window and solve the problem in a 3D manner. As for the task of moving object recognition, we propose a spatial-temporal approach to solve it, with only a 2D planar LIDAR. Avoiding using more elaborate and costly sensors like a Velodyne, we show that it is possible to obtain highly accurate object classification via temporal accumulation. Our algorithm is tested in both campus and highway scenarios, and shows good accuracy.

Besides the object recognition functions developed for the short-term object-oriented detection purpose, to endow the robot with higher-level intelligence, we are also interested in acquiring some long-term environment-oriented understanding. While the understanding of an environment can concern anything about its properties, in our research, we concentrate on the semantic and activity dimensions. Unlike existing research approaching different dimensions of understanding independently, we argue that these dimensions are highly correlated and can be learned from each other. We implement this idea to infer semantic understanding from learned activity knowledge, and achieve promising results.

9.1.2 Insights of Minimal Sensing

We summarize three key insights of “perception under minimal sensing” as follows, which can be treated as the essence of our research:

1. **Make use of prior knowledge.** The prior knowledge of an environment is of vital importance for a vehicle to navigate in it. This prior knowledge can be a metric map, which assists vehicles in performing map-aided localization: Chapter 3 utilizes a “curb map” as the prior, and Chapter 4 makes use of a “vertical surface map” generated beforehand. The prior knowledge can also be semantic or activity related, as we learned in Chapter 8. Although no concrete implementations are shown in this thesis that utilize the semantic or activity knowledge, it is well acknowledged that such information is quite beneficial for vehicle perception and navigation.
2. **Make use of the characteristics of the environment.** Due to limited sensing ability, the characteristics of the environment have to be well studied and utilized. For vehicle localization, we avoid using expensive GPS/INS units, and make use of the typical features in the urban road environment for pose estimation, i.e. curb-intersection features (Chapter 3) and vertical surface features (Chapter 4). Since the urban roads are usually flat with only large curvatures at the boundaries, we utilize this observation for road surface-boundary detection (Chapter 6). In addition, well painted roads make it possible for us to extract and recognize the markings on them (Chapter 5).
3. **Make use of the temporal relationship between adjacent measurements.** Through the whole thesis, we rely on the two 2D LIDARs as the major exteroceptive sensors, and realize most of the perception functions. While 2D LIDARs only provide a series of range values whose information is very sparse, we manage to overcome this difficulty by referring to the temporal relationships between adjacent measurements. The idea of a 3D rolling window captures the temporal relationships between laser scans from a tilted-down LIDAR, and allows us to accumulate 3D data and reconstruct the static environment as the vehicle moves. We rely on the accumulated 3D data for not only vehicle localization (Chapter 4), but also road surface-boundary detection (Chapter 6). Chapter 7 introduces our spatial-temporal

approach for moving object detection, where we utilize the temporal patterns of consecutive scans from a planar LIDAR for object classification.

9.1.3 Contributions

In summary, this thesis introduces our research into the perception functions of vehicle autonomous driving in the urban road environment. Table 9.1 summarizes the sensors usage of different perception functions studied in this thesis. We demonstrate that with the minimal sensing configuration, our algorithms are able to achieve equivalent or better performance compared to the existing work. The contributions can be summarized as follows:

1. We develop a curb-intersection feature based Monte Carlo Localization algorithm, which achieves accurate estimation results in both the lateral and the longitudinal directions of urban roads. Compared to existing approaches that purely use curb features, we introduce an “intersection feature” as a complement, and utilize the combined curb-intersection feature for the localization purpose. The contributions of our algorithm also include the way we represent and utilize the curb-intersection features. The idea of “synthetic LIDAR” enables us to encode the features into the format of laser scans, and use the standard measurement models of laser sensors for accurate and robust localization (Chapter 3).
2. We propose a vertical-surface feature based localization algorithm, which achieve equivalent performances to the state-of-the-art algorithm using a 3D LIDAR, however with the much reduced sensing ability of a 2D laser sensor. Similar to the curb-intersection algorithm, the contributions of our algorithm also include the idea of “synthetic LIDAR”, enabling us to use the standard measurement model of laser sensors for localization on the projected 2D plane, both efficiently and precisely (Chapter 4).
3. We design a general framework for road marking detection and analysis using vision; unlike existing approaches which are usually case-specific, the proposed method is able to detect various types of markings under a uniform framework (Chapter 5).

Table 9.1: Sensor usage of different perception functions

Perception Functions		Chapter	Required Sensors			
			Odometry	Tilted LIDAR	Planar LIDAR	Webcam
Localization	Curb-intersection Feature based	3	√	√	×	×
	Synthetic LIDAR based	4	√	√	×	×
Object Recognition	Road marking detection	5	×	×	×	√
	Road surface-boundary detection	6	√	√	×	×
	Moving object recognition	7	√	×	√	×
Environment Understanding		8	√	√	√	√

4. We introduce a cascaded approach for road surface-boundary detection using accumulated 3D data; compared to existing 2D approaches, our algorithm does not have strong assumptions on the sensing scenario, and is able to handle temporal noise; the contributions here also include the study for the probabilistic characteristics of the accumulated 3D data (Chapter 6).
5. We develop a spatial-temporal approach for moving object recognition using a 2D planar LIDAR, which achieves equivalent/better performance than the state-of-the-art algorithm; the contributions include not only the spatial-temporal method itself, but also the designed spatial-temporal features of 2D laser scans (Chapter 7).
6. We study the different dimensions of environment understanding, and propose the novel idea of semantic mapping through pedestrian activity learning; while existing approaches usually solve the environment understanding problem at different dimensions individually, we study the correlations between them and demonstrate the feasibility of learning knowledge from each other (Chapter 8).
7. We summarize three key insights of “perception under minimal sensing”, which can be treated as the essence of our research (Chapter 9).

9.2 System Integration and Practical Issues

This section introduces the integration of the developed perception functions in our autonomous vehicle system, and discusses some practical issues of the current sensor configuration.

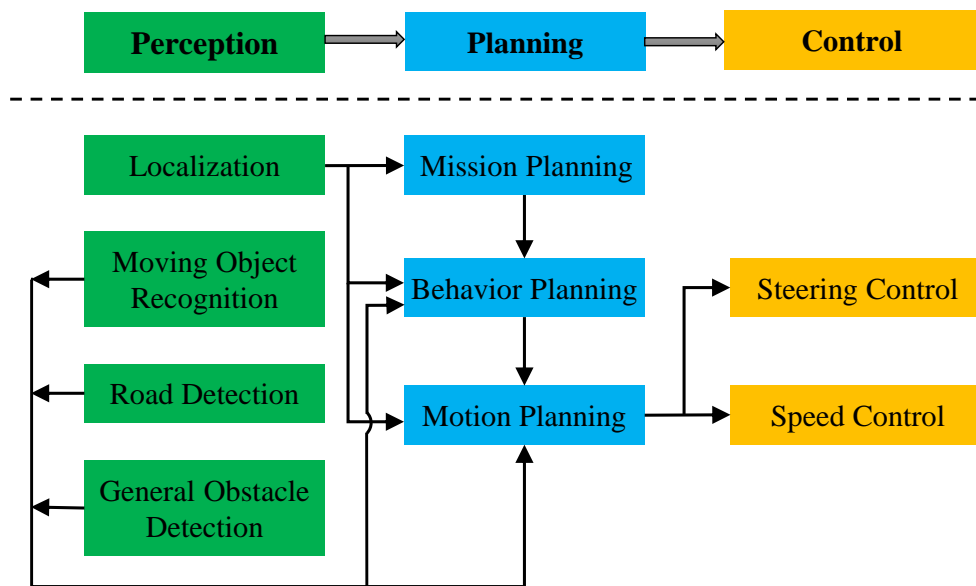


Figure 9.1: Software system of our autonomous vehicle

Figure 9.1 illustrates the software system of our autonomous vehicle, which is consisted of three modules: perception, planning and control. The perception module estimates the states of the surrounding environment as well as the states of the vehicle itself, and provides the information to the planning module. We adopts a three-layer architecture for the planning module [158], which includes the mission-planning layer for global path selection, the behavior-planning layer to choose appropriate driving mode, and the motion-planning layer for local path generation and obstacle avoidance. The control module takes care of the low-level speed and steering control. From the figure, it can be illustrated that the localization function plays a vital role for the planning module, depending on which the vehicle carries out all the three layers of planning. The outputs of moving object detection and road detection are used for motion planning, where the vehicle determines its trajectory as well as speed to drive on-road and avoid other moving objects. For safety considerations, we also implement a “general object detection” function in the perception module, which takes the input from the planar LIDAR, and treats every laser point as an obstacle. This guarantees the safety of the vehicle in the presence of general obstacles, and in case of the failures of other perception functions.

In our current prototype, we use on two LIDARs for different purposes: a tilted-down LIDAR for localization and road detection (chapter 3, chapter 4, and chapter 6),

and a planar LIDAR for obstacle detection (chapter 7). Generally the current sensor configuration achieves satisfactory performance; however, there are two practical issues to be considered in real applications.

The first issue is related to the sensor coverage. In the current sensor configuration, obstacle recognition is handled by the planar LIDAR, which assumes that obstacles appear at the height of the planar LIDAR. While this assumption generally holds in the real road traffic, it may fail in some extreme cases such as short kids or dogs. Although the tilted-down LIDAR may be used for the detection of these low obstacles, it is only able to detect them from a particular distance (depending on its mounting height and tilted-down angle). To tackle this limitation but avoid using 3D LIDAR, we are exploring the idea of active sensing: the planar LIDAR will be mounted on a pan-tilt mechanism to achieve complete sensor coverage.

The second issue is related to the sensory modality. Currently our perception algorithms rely heavily on the LIDAR sensors. Although LIDARs are well-known for their robustness and accuracy, they also have their own limitations. One of the biggest challenges with the LIDAR is the black glossy vehicles, whose surfaces have low rate of diffuse reflection, leading to missing data in the laser scans. For this reason, our planar LIDAR has difficulty in recognizing such vehicles from a distance. To tackle this limitation, other sensory modalities have to be employed. Currently we are working on fusing the vision-based vehicle detection technology with our LIDAR-based method.

9.3 Future Work

This section introduces the possible topics in our future work. Besides the planned works discussed at the end of each chapter, our future research will be performed in the following four areas:

1. **Extend the current algorithms from a single sensory modality to multiple sensory modalities.** While most of our existing work is carried out using a single sensory modality, to have redundancy as well as to make the algorithms more robust, we plan to utilize multiple sensory modalities for each perception function. For example, vision-based localization will be developed as a complement and fused to the current LIDAR-based algorithm.

2. **Extend the research from a single vehicle to multiple vehicles.** While this thesis only discusses the perception issues with a single robot, some interesting perception problems may appear when multiple autonomous vehicles work together, such as cooperative localization, cooperative SLAM, etc. The problems of cooperative perception will be studied to realize better perception.
3. **Realize Metric-Semantic-Activity SLAM in the dynamic environment.** In the research into environment understanding, we argue that the three dimensions of environment understanding are correlated and can be learned from each other. We want to apply this idea to realize metric-semantic-activity SLAM in the dynamic environment. While traditional metric SLAM research treats the environment as static, the assumption does not hold in the crowded and dynamic urban environment. By incorporating semantic and activity information in the SLAM process, a better and more meaningful model of the environment can be built.
4. **Explore active sensing.** In the current setup, all of our algorithms can be treated as “passive” sensing, in the sense that the sensors are fixed, and cannot change their positions and orientations actively to achieve better perception performance. The idea of active sensing will be explored in our future research. Enabling us to dynamically allocate the limited sensing resources to more urgent perception tasks, it fits perfectly into our idea of “perception under minimal sensing”.

Bibliography

- [1] M. Buehler, K. Iagnemma, and S. Singh, “The 2005 darpa grand challenge,” *Springer Tracts in Advanced Robotics*, vol. 36, no. 5, pp. 1–43, 2007.
- [2] ———, *The DARPA urban challenge: autonomous vehicles in city traffic*. Springer, 2009, vol. 56.
- [3] J. Markoff, “Google cars drive themselves, in traffic,” *The New York Times*, vol. 10, p. A1, 2010.
- [4] Z. J. Chong, B. Qin, T. Bandyopadhyay, T. Wongpiromsarn, E. S. Rankin, M. H. Ang Jr., E. Frazzoli, D. Rus, D. Hsu, and K. H. Low, “Autonomous Personal Vehicle for the First- and Last-Mile Transportation Services,” in *IEEE International Conference on Robotics, Automation and Mechatronics (RAM)*, 2011.
- [5] Z. J. Chong, B. Qin, T. Bandyopadhyay, T. Wongpiromsarn, B. Rebsamen, P. Dai, E. S. Rankin, and M. H. Ang Jr., “Autonomy for mobility on demand,” in *IEEE International Conference on Intelligent Autonomous Systems*, 2012.
- [6] B. Qin, Z. J. Chong, T. Bandyopadhyay, M. H. Ang Jr., E. Frazzoli, and D. Rus, “Curb-Intersection Feature Based Monte Carlo Localization on Urban Roads,” in *IEEE International Conference on Robotics and Automation*, 2012.
- [7] Z. J. Chong, B. Qin, T. Bandyopadhyay, T. Wongpiromsarn, B. Rebasmen, P. Dai, S. Kim, M. H. Ang Jr., D. Hsu, D. Rus, and E. Frazzoli, “Autonomy for Mobility on Demand,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.
- [8] Z. J. Chong, B. Qin, T. Bandyopadhyay, M. Ang, E. Frazzoli, and D. Rus, “Synthetic 2d LIDAR for precise vehicle localization in 3d urban environment,” in

- IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany, May 2013.
- [9] B. Qin, Z. J. Chong, T. Bandyopadhyay, M. H. Ang Jr., E. Frazzoli, and D. Rus, “Road Detection and Mapping using 3D Rolling Window,” in *Intelligent Vehicles Symposium (IV), 2013 IEEE*. IEEE.
- [10] B. Qin, Z. J. Chong, T. Bandyopadhyay, and M. H. Ang Jr., “Metric Mapping and Topo-metric Graph Learning of Urban Road Network,” in *IEEE International Conference on Robotics, Automation and Mechatronics (RAM)*, 2013.
- [11] B. Qin, Z. J. Chong, T. Bandyopadhyay, M. H. Ang Jr., E. Frazzoli, and D. Rus, “Learning Pedestrian Activities for Semantic Mapping,” in *IEEE International Conference on Robotics and Automation*, 2014.
- [12] B. Qin, Z. J. Chong, S. H. Soh, T. Bandyopadhyay, M. H. Ang Jr., E. Frazzoli, and D. Rus, “Learning Pedestrian Activities for Semantic Mapping,” in *IEEE International Conference on Robotics and Automation*, 2014.
- [13] Smart driverless golf cart provides a glimpse into a future of autonomous vehicles. <http://newsoffice.mit.edu/2013/smart-driverless-golf-cart-provides-a-glimpse-into-a-future-of-autonomous-vehicles>.
- [14] Singapore-made driverless car to ply nus roads. <http://www.straitstimes.com/breaking-news/singapore/story/singapore-made-driverless-car-ply-nus-roads-20140129>.
- [15] J. Levinson and S. Thrun, “Robust vehicle localization in urban environments using probabilistic maps,” in *ICRA’10*, 2010, pp. 4372–4378.
- [16] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT Press, 2005.
- [17] B. Qin, Z. J. Chong, T. Bandyopadhyay, M. H. Ang Jr., E. Frazzoli, and D. Rus, “A general framework for road marking detection and analysis,” in *Intelligent Transportation Systems Conference, 2013. ITSC’13. IEEE*.
- [18] Autonomous car. http://en.wikipedia.org/wiki/Autonomous_car.

-
- [19] U. Ozguner, T. Acarman, and K. A. Redmill, *Autonomous ground vehicles*. Artech House, 2011.
- [20] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann *et al.*, “Stanley: The robot that won the darpa grand challenge,” *The 2005 DARPA Grand Challenge*, pp. 1–43, 2007.
- [21] J. Markoff, “Google Cars Drive Themselves, in Traffic,” *The New York Times*, October 9, 2010.
- [22] A. Broggi, M. Buzzoni, S. Debattisti, P. Grisleri, M. C. Laghi, P. Medici, and P. Versari, “Extensive tests of autonomous driving technologies,” 2013.
- [23] J. Wei, J. M. Snider, J. Kim, J. M. Dolan, R. Rajkumar, and B. Litkouhi, “Towards a viable autonomous driving research platform,” in *Intelligent Vehicles Symposium (IV), 2013 IEEE*. IEEE, 2013, pp. 763–770.
- [24] Robotcar. <http://mrg.robots.ox.ac.uk/robotcar/>.
- [25] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt *et al.*, “Towards fully autonomous driving: Systems and algorithms,” in *Intelligent Vehicles Symposium (IV), 2011 IEEE*. IEEE, 2011, pp. 163–168.
- [26] M. J. Milford and G. F. Wyeth, “Mapping a suburb with a single camera using a biologically inspired slam system,” *Robotics, IEEE Transactions on*, vol. 24, no. 5, pp. 1038–1053, 2008.
- [27] M. Cummins and P. Newman, “Highly scalable appearance-only slam-fab-map 2.0,” in *Robotics: Science and Systems*, vol. 1. Seattle, USA, 2009, pp. 12–18.
- [28] C. McManus, W. Churchill, W. Maddern, A. Stewart, and P. Newman, “Shady Dealings: Robust, Long- Term Visual Localisation using Illumination Invariance,” in *IEEE International Conference on Robotics and Automation*, 2014.
- [29] H. Cho, P. E. Rybski, A. Bar-Hillel, and W. Zhang, “Real-time pedestrian detection with deformable part models,” in *Intelligent Vehicles Symposium (IV), 2012 IEEE*. IEEE, 2012, pp. 1035–1042.

- [30] A. Cappalunga, S. Cattani, A. Broggi, M. S. McDaniel, and S. Dutta, “Real time 3d terrain elevation mapping using ants optimization algorithm and stereo vision,” in *Intelligent Vehicles Symposium (IV), 2010 IEEE*. IEEE, 2010, pp. 902–909.
- [31] S. Sengupta, P. Sturgess, P. H. Torr *et al.*, “Automatic dense visual semantic mapping from street-level imagery,” in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*.
- [32] S. Sengupta, E. Greveson, A. Shahrokni, and P. H. Torr, “Urban 3d semantic modelling using stereo vision,” in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 580–585.
- [33] A. Mohamed and K. Schwarz, “Adaptive Kalman filtering for INS/GPS,” *Journal of Geodesy*, vol. 73, no. 4, pp. 193–203, 1999.
- [34] H. H. Qi and J. B. Moore, “Direct Kalman filtering approach for GPS/INS integration,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 38, no. 2, pp. 687–693, 2002.
- [35] H. Carvalho, P. DelMoral, A. Monin, and G. Salut, “Optimal nonlinear filtering in GPS/INS integration,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 33, no. 3, pp. 835–850, 1997.
- [36] I. Skog and P. Handel, “In-car positioning and navigation technologiesa survey,” *Intelligent Transportation Systems, IEEE Transactions on*, vol. 10, no. 1, pp. 4–21, 2009.
- [37] Inertial navigation system. http://en.wikipedia.org/wiki/Inertial_navigation_system.
- [38] C.-W. Tan and S. Park, “Design of accelerometer-based inertial navigation systems,” *Instrumentation and Measurement, IEEE Transactions on*, vol. 54, no. 6, pp. 2520–2530, 2005.
- [39] Y. Yi and D. Grejner-Brzezinska, “Tightly-coupled gps/ins integration using unscented kalman filter and particle filter,” in *Proceedings of the 19th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2006)*, 2001, pp. 2182–2191.

- [40] M. E. El Najjar and P. Bonnifait, "A road-matching method for precise vehicle localization using belief theory and Kalman filtering," *Autonomous Robots*, vol. 19, no. 2, pp. 173–191, 2005.
- [41] J. Guivant and R. Katz, "Global urban localization based on road maps," *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vols 1-9*, pp. 1085–1090, 2007.
- [42] C. Fouque, P. Bonnifait, and D. Betaille, "Enhancement of global vehicle localization using navigable road maps and dead-reckoning," *2008 IEEE/ION Position, Location and Navigation Symposium, Vols 1-3*, pp. 999–1004, 2008.
- [43] M. Jabbour and P. Bonnifait, "Global localization robust to gps outages using a vertical lidar," in *Control, Automation, Robotics and Vision, 2006. ICARCV'06. 9th International Conference on*. IEEE, 2006, pp. 1–6.
- [44] P. Bonnifait, M. Jabbour, and V. Cherfaoui, "Autonomous navigation in urban areas using gis-managed information," *International Journal of Vehicle Autonomous Systems*, vol. 6, no. 1, pp. 83–103, 2008.
- [45] N. Suganuma and T. Uozumi, "Precise position estimation of autonomous vehicle based on map-matching," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*. IEEE, pp. 296–301.
- [46] Y. Morales, T. Tsubouchi, and S. Yuta, "Vehicle 3d localization in mountainous woodland environments," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. IEEE, 2009, pp. 3588–3594.
- [47] M. Hentschel, O. Wulf, and B. Wagner, "A GPS and Laser-based Localization for Urban and Non-Urban Outdoor Environments," *2008 IEEE/RSJ International Conference on Robots and Intelligent Systems, Vols 1-3, Conference Proceedings*, pp. 149–154, 2008.
- [48] O. Wulf, K. O. Arras, H. I. Christensen, and B. Wagner, "2d mapping of cluttered indoor environments by means of 3d perception," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 4. IEEE, 2004, pp. 4204–4209.

- [49] M. Hentschel and B. Wagner, “Autonomous robot navigation based on open-streetmap geodata,” in *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*. IEEE, 2010, pp. 1645–1650.
- [50] I. Miller, M. Campbell, and D. Huttenlocher, “Map-aided localization in sparse global positioning system environments using vision and particle filtering,” *Journal of Field Robotics*, vol. 28, no. 5, pp. 619–643, 2011.
- [51] J. Levinson, M. Montemerlo, and S. Thrun, “Map-based precision vehicle localization in urban environments,” in *Proceedings of Robotics: Science and Systems*, Atlanta, GA, USA, June 2007.
- [52] I. Baldwin and P. Newman, “Road vehicle localization with 2d push-broom lidar and 3d priors,” in *Proc. IEEE International Conference on Robotics and Automation (ICRA2012)*, Minnesota, USA, May 2012.
- [53] H. Durrant-Whyte and T. Bailey, “Simultaneous localization and mapping: part i,” *Robotics & Automation Magazine, IEEE*, vol. 13, no. 2, pp. 99–110, 2006.
- [54] T. Bailey and H. Durrant-Whyte, “Simultaneous localization and mapping (slam): Part ii,” *IEEE Robotics & Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006.
- [55] J. Neira and J. D. Tardós, “Data association in stochastic mapping using the joint compatibility test,” *Robotics and Automation, IEEE Transactions on*, vol. 17, no. 6, pp. 890–897, 2001.
- [56] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit *et al.*, “Fastslam: A factored solution to the simultaneous localization and mapping problem,” in *Proceedings of the National conference on Artificial Intelligence*. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2002, pp. 593–598.
- [57] M. Montemerlo and S. Thrun, “Fastslam 2.0,” *FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics*, pp. 63–90, 2007.
- [58] G. Grisetti, C. Stachniss, and W. Burgard, “Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling,” in

- Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on.* IEEE, 2005, pp. 2432–2437.
- [59] ———, “Improved techniques for grid mapping with rao-blackwellized particle filters,” *Robotics, IEEE Transactions on*, vol. 23, no. 1, pp. 34–46, 2007.
- [60] S. Thrun and M. Montemerlo, “The graph slam algorithm with applications to large-scale mapping of urban structures,” *The International Journal of Robotics Research*, vol. 25, no. 5-6, pp. 403–429, 2006.
- [61] I. Ulrich and I. Nourbakhsh, “Appearance-based place recognition for topological localization,” in *Robotics and Automation, 2000. Proceedings. ICRA’00. IEEE International Conference on*, vol. 2. Ieee, 2000, pp. 1023–1029.
- [62] P. Newman, D. Cole, and K. Ho, “Outdoor slam using visual appearance and laser ranging,” in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on.* IEEE, 2006, pp. 1180–1187.
- [63] M. Saedan, C. W. Lim, and V. Ang, “Appearance-based slam with map loop closing using an omnidirectional camera,” in *Advanced intelligent mechatronics, 2007 IEEE/ASME international conference on.* IEEE, 2007, pp. 1–6.
- [64] M. Cummins and P. Newman, “Fab-map: Probabilistic localization and mapping in the space of appearance,” *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 647–665, 2008.
- [65] ———, “Appearance-only slam at large scale with fab-map 2.0,” *The International Journal of Robotics Research*, vol. 30, no. 9, pp. 1100–1123, 2011.
- [66] R. Kümmerle, R. Triebel, P. Pfaff, and W. Burgard, “Monte carlo localization in outdoor terrains using multilevel surface maps,” *Journal of Field Robotics*, vol. 25, no. 6-7, pp. 346–359, 2008.
- [67] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann, “6d slam3d mapping outdoor environments,” *Journal of Field Robotics*, vol. 24, no. 8-9, pp. 699–722, 2007.

- [68] M. Aly, "Real time detection of lane markers in urban streets," in *Intelligent Vehicles Symposium, 2008 IEEE*. IEEE, 2008, pp. 7–12.
- [69] Z. Sun, G. Bebis, and R. Miller, "On-road vehicle detection: A review," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 5, pp. 694–711, 2006.
- [70] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, no. 9, pp. 1627–1645, 2010.
- [71] H. T. Niknejad, A. Takeuchi, S. Mita, and D. McAllester, "On-road multivehicle tracking using deformable object model and particle filter with improved likelihood estimation," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 13, no. 2, pp. 748–758, 2012.
- [72] O. M. Mozos, R. Kurazume, and T. Hasegawa, "Multi-part people detection using 2d range data," *International Journal of Social Robotics*, vol. 2, no. 1, pp. 31–40, 2010.
- [73] H. Zhao and R. Shibasaki, "A novel system for tracking pedestrians using multiple single-row laser-range scanners," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 35, no. 2, pp. 283–291, 2005.
- [74] A. Petrovskaya and S. Thrun, "Model based vehicle detection and tracking for autonomous urban driving," *Autonomous Robots*, vol. 26, no. 2-3, pp. 123–139, 2009.
- [75] T.-D. Vu and O. Aycard, "Laser-based detection and tracking moving objects using data-driven markov chain monte carlo," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 3800–3806.
- [76] F. Nashashibi and A. Bargeton, "Laser-based vehicles tracking and classification using occlusion reasoning and confidence estimation," in *Intelligent Vehicles Symposium, 2008 IEEE*. IEEE, 2008, pp. 847–852.

- [77] D. Z. Wang, I. Posner, and P. Newman, “What could move? finding cars, pedestrians and bicyclists in 3d laser data,” in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 4038–4044.
- [78] A. Teichman, J. Levinson, and S. Thrun, “Towards 3d object recognition via classification of arbitrary object tracks,” in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 4034–4041.
- [79] G. Reina, J. Underwood, G. Brooker, and H. Durrant-Whyte, “Radar-based perception for autonomous outdoor vehicles,” *Journal of Field Robotics*, vol. 28, no. 6, pp. 894–913, 2011.
- [80] K. Kaliyaperumal, S. Lakshmanan, and K. Kluge, “An algorithm for detecting roads and obstacles in radar images,” *Vehicular Technology, IEEE Transactions on*, vol. 50, no. 1, pp. 170–182, 2001.
- [81] G. Gate and F. Nashashibi, “Fast algorithm for pedestrian and group of pedestrians detection using a laser scanner,” in *Intelligent Vehicles Symposium, 2009 IEEE*. IEEE, 2009, pp. 1322–1327.
- [82] C. Premebida, G. Monteiro, U. Nunes, and P. Peixoto, “A lidar and vision-based approach for pedestrian and vehicle detection and tracking,” in *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*. IEEE, 2007, pp. 1044–1049.
- [83] L. Bombini, P. Cerri, P. Medici, and G. Alessandretti, “Radar-vision fusion for vehicle detection,” in *Procs. International Workshop on Intelligent Transportation*, 2006, pp. 65–70.
- [84] W. Zhang, “LIDAR-based road and road-edge detection,” in *Intelligent Vehicles Symposium (IV), 2010 IEEE*, pp. 845–848.
- [85] R. de Charette and F. Nashashibi, “Real time visual traffic lights recognition based on spot light detection and adaptive traffic lights templates,” in *Intelligent Vehicles Symposium, 2009 IEEE*. IEEE, 2009, pp. 358–363.

- [86] Y. Shen, U. Ozguner, K. Redmill, and J. Liu, "A robust video based traffic light detection algorithm for intelligent vehicles," in *Intelligent Vehicles Symposium, 2009 IEEE*. IEEE, 2009, pp. 521–526.
- [87] C. Bahlmann, Y. Zhu, V. Ramesh, M. Pellkofer, and T. Koehler, "A system for traffic sign detection, tracking, and recognition using color, shape, and motion information," in *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE*. IEEE, 2005, pp. 255–260.
- [88] G. Y. Song, K. Y. Lee, and J. W. Lee, "Vehicle detection by edge-based candidate generation and appearance-based classification," in *Intelligent Vehicles Symposium, 2008 IEEE*. IEEE, 2008, pp. 428–433.
- [89] D. Held, J. Levinson, and S. Thrun, "A probabilistic framework for car detection in images using context and scale," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 1628–1634.
- [90] A. Borkar, M. Hayes, M. Smith, and S. Pankanti, "A layered approach to robust lane detection at night," in *Computational Intelligence in Vehicles and Vehicular Systems, 2009. CIVVS'09. IEEE Workshop on*. IEEE, 2009, pp. 51–57.
- [91] A. Borkar, M. Hayes, and M. Smith, "Robust lane detection and tracking with ransac and kalman filter," in *Image Processing (ICIP), 2009 16th IEEE International Conference on*. IEEE, 2009, pp. 3261–3264.
- [92] C. D'Cruz and J. Zou, "Lane detection for driver assistance and intelligent vehicle applications," in *Communications and Information Technologies, 2007. ISCIT'07. International Symposium on*. IEEE, 2007, pp. 1291–1296.
- [93] S. Vacek, C. Schimmel, and R. Dillmann, "Road-marking analysis for autonomous vehicle guidance," in *Proceedings of the 3rd European Conference on Mobile Robots, EMCR 2007, September 19-21, 2007, Freiburg, Germany, 2007*.
- [94] Y. Li, K. He, and P. Jia, "Road markers recognition based on shape information," in *Intelligent Vehicles Symposium, 2007 IEEE*, june 2007, pp. 117–122.

-
- [95] S. Se, “Zebra-crossing detection for the partially sighted,” in *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, vol. 2, 2000, pp. 211–217 vol.2.
- [96] S. Thrun, M. Montemerlo, and A. Aron, “Probabilistic terrain analysis for high-speed desert driving,” *Proc. Robotics Science and Systems, Philadelphia, PA, USA*, 2006.
- [97] H. Cramer and G. Wanielik, “Road border detection and tracking in non cooperative areas with a laser radar system,” in *Proceedings of German Radar Symposium*, 2002, pp. 24–29.
- [98] W. S. Wijesoma, K. R. S. Kodagoda, and A. P. Balasuriya, “Road-boundary detection and tracking using lidar sensing,” *IEEE Transactions on Robotics and Automation*, vol. 20, no. 3, pp. 456–464, 2004.
- [99] K. R. S. Kodagoda, W. S. Wijesoma, and A. P. Balasuriya, “CuTE: Curb tracking and estimation,” *IEEE Transactions on Control Systems Technology*, vol. 14, no. 5, pp. 951–957, 2006.
- [100] F. Oniga, S. Nedeveschi, and M. Meinecke, “Curb detection based on a multi-frame persistence map for urban driving scenarios,” in *Intelligent Transportation Systems, 2008. ITSC 2008. 11th International IEEE Conference on*. IEEE, 2008, pp. 67–72.
- [101] C. Guo, S. Mita, and D. McAllester, “Stereovision-based road boundary detection for intelligent vehicles in challenging scenarios,” in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. IEEE, 2009, pp. 1723–1728.
- [102] C. Mertz, L. E. Navarro-Serment, R. MacLachlan, P. Rybski, A. Steinfeld, A. Suppé, C. Urmson, N. Vandapel, M. Hebert, C. Thorpe *et al.*, “Moving object detection with laser scanners,” *Journal of Field Robotics*, vol. 30, no. 1, pp. 17–43, 2013.

-
- [103] T. Miyasaka, Y. Ohama, and Y. Ninomiya, “Ego-motion estimation and moving object tracking using multi-layer lidar,” in *Intelligent Vehicles Symposium, 2009 IEEE*. IEEE, 2009, pp. 151–156.
- [104] T. D. Vu, J. Burlet, and O. Aycard, “Grid-based localization and local mapping with moving object detection and tracking,” *Information Fusion*, 2011.
- [105] C. C. Wang, C. Thorpe, and S. Thrun, “Online simultaneous localization and mapping with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas,” in *Robotics and Automation, 2003. Proceedings. ICRA’03. IEEE International Conference on*. IEEE.
- [106] O. Mozos, C. Stachniss, and W. Burgard, “Supervised learning of places from range data using adaboost,” in *Robotics and Automation, 2005. ICRA. Proceedings of the IEEE International Conference on*.
- [107] O. Martinez Mozos, A. Rottmann, R. Triebel, P. Jensfelt, W. Burgard *et al.*, “Semantic labeling of places using information extracted from laser and vision sensor data,” 2006.
- [108] A. Pronobis, O. M. Mozos, B. Caputo, and P. Jensfelt, “Multi-modal semantic place classification,” *The International Journal of Robotics Research*, vol. 29, no. 2-3, pp. 298–320, 2010.
- [109] I. Posner, D. Schroeter, and P. Newman, “Online generation of scene descriptions in urban environments,” *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 901–914, 2008.
- [110] A. Nüchter and J. Hertzberg, “Towards semantic maps for mobile robots,” *Robotics and Autonomous Systems*, 2008.
- [111] C. Galindo, A. Saffiotti, S. Coradeschi, P. Buschka, J.-A. Fernandez-Madriral, and J. González, “Multi-hierarchical semantic maps for mobile robotics,” in *Intelligent Robots and Systems, IEEE/RSJ International Conference on (IROS)*. IEEE, 2005.

- [112] S. Vasudevan and R. Siegwart, “Bayesian space conceptualization and place classification for semantic maps in mobile robotics,” *Robotics and Autonomous Systems*, vol. 56, no. 6, pp. 522–537, 2008.
- [113] D. F. Wolf and G. S. Sukhatme, “Semantic mapping using mobile robots,” *Robotics, IEEE Transactions on*, vol. 24, no. 2, pp. 245–258, 2008.
- [114] D. Xie, S. Todorovic, and S.-C. Zhu, “Inferring ”dark matter” and ”dark energy” from videos,” in *The IEEE International Conference on Computer Vision (ICCV)*, 2013.
- [115] G. Li, C. Zhu, J. Du, Q. Cheng, W. Sheng, and H. Chen, “Robot semantic mapping through wearable sensor-based human activity recognition,” in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 5228–5233.
- [116] N. Johnson and D. Hogg, “Learning the distribution of object trajectories for event recognition,” *Image and Vision Computing*, vol. 14, no. 8, pp. 609–615, 1996.
- [117] E. Swears, A. Hoogs, and A. A. Perera, “Learning motion patterns in surveillance video using hmm clustering,” in *Motion and video Computing, 2008. WMVC 2008. IEEE Workshop on*. IEEE, 2008, pp. 1–8.
- [118] D. Ellis, E. Sommerlade, and I. Reid, “Modelling pedestrian trajectory patterns with gaussian processes,” in *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 1229–1234.
- [119] X. Wang, K. T. Ma, G.-W. Ng, and W. E. L. Grimson, “Trajectory analysis and semantic region modeling using nonparametric hierarchical bayesian models,” *International journal of computer vision*, 2011.
- [120] A. Lookingbill, D. Lieb, D. Stavens, and S. Thrun, “Learning activity-based ground models from a moving helicopter platform,” in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, 2005, pp. 3948–3953.

- [121] M. Bennewitz, W. Burgard, and S. Thrun, "Learning motion patterns of persons for mobile service robots," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 4. IEEE, 2002, pp. 3601–3606.
- [122] S. Sehestedt, S. Kodagoda, and G. Dissanayake, "Models of motion patterns for mobile robotic systems," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 4127–4132.
- [123] S. Thrun, W. Burgard, and D. Fox, "A real-time algorithm for mobile robot mapping with applications to multi-robot and 3d mapping," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 1. IEEE, 2000, pp. 321–328.
- [124] A. Harrison and P. Newman, "High quality 3d laser ranging under general vehicle motion," in *Proc. IEEE International Conference on Robotics and Automation (ICRA'08)*, Pasadena, California, April 2008.
- [125] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation," in *Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. IEEE, November 2011.
- [126] J. Weingarten and G. Gruener, "A fast and robust 3d feature extraction algorithm for structured environment reconstruction," in *Reconstruction, 11th International Conference on Advanced Robotics (ICAR, 2003)*.
- [127] R. Kummerle, R. Triebel, P. Pfaff, and W. Burgard, "Monte carlo localization in outdoor terrains using multilevel surface maps," *Journal of Field Robotics*, vol. 25, no. 6-7, pp. 346–359, 2008.
- [128] K. Klasing, D. Althoff, D. Wollherr, and M. Buss, "Comparison of surface normal estimation methods for range sensing applications," in *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, may 2009, pp. 3206–3211.
- [129] J. Berkmann and T. Caelli, "Computation of surface geometry and segmentation using covariance techniques," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 16, no. 11, pp. 1114–1116, nov 1994.

- [130] C. M. Shakarji, “Least-squares fitting algorithms of the nist algorithm testing system,” in *Journal of Research of the National Institute of Standards and Technology*, 1998, pp. 633–641.
- [131] R. Rusu, “Semantic 3d object maps for everyday manipulation in human living environments,” *KI-Künstliche Intelligenz*, vol. 24, no. 4, pp. 345–348, 2010.
- [132] M. Muja and D. G. Lowe, “Fast approximate nearest neighbors with automatic algorithm configuration,” in *International Conference on Computer Vision Theory and Application VISSAPP’09*. INSTICC Press, 2009, pp. 331–340.
- [133] R. B. Rusu and S. Cousins, “3D is here: Point Cloud Library (PCL),” in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.
- [134] G. Tipaldi and K. Arras, “Flirt - interest regions for 2d range data,” in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, may 2010, pp. 3616 –3622.
- [135] M. Kaess, A. Ranganathan, and F. Dellaert, “iSAM: Incremental smoothing and mapping,” *IEEE Trans. on Robotics (TRO)*, vol. 24, no. 6, pp. 1365–1378, Dec. 2008.
- [136] H. A. Mallot, H. H. Bülthoff, J. Little, and S. Bohrer, “Inverse perspective mapping simplifies optical flow computation and obstacle detection,” *Biological cybernetics*, vol. 64, no. 3, pp. 177–185, 1991.
- [137] N. Nourani-Vatani and J. Roberts, “Automatic camera exposure control,” in *Proc. of the 2007 Australasian Conference on Robotics and Automation, Brisbane, Australia (December 2007)*, 2007.
- [138] C.-C. Chang and C.-J. Lin, “LIBSVM: A library for support vector machines,” *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011.
- [139] T. Zhang and C. Y. Suen, “A fast parallel algorithm for thinning digital patterns,” *Communications of the ACM*, vol. 27, no. 3, pp. 236–239, 1984.

- [140] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.
- [141] R. Smith, “An overview of the tesseract ocr engine,” in *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, vol. 2. IEEE, 2007, pp. 629–633.
- [142] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, “Visual categorization with bags of keypoints,” in *Workshop on statistical learning in computer vision, ECCV*, vol. 1, 2004, p. 22.
- [143] G. Bradski, “The OpenCV Library,” *Dr. Dobb’s Journal of Software Tools*, 2000.
- [144] W. Burgard, “Techniques for 3d mapping theory,” in *The 4th Summer School in Simultaneous Localisation and Mapping (SLAM)*, Sydney, January 2009.
- [145] R. Rusu, Z. Marton, N. Blodow, M. Dolha, and M. Beetz, “Functional object mapping of kitchen environments,” in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE, 2008, pp. 3525–3532.
- [146] P. F. Felzenszwalb and D. P. Huttenlocher, “Efficient graph-based image segmentation,” *International Journal of Computer Vision*, vol. 59, no. 2, pp. 167–181, 2004.
- [147] D. H. Douglas and T. K. Peucker, “Algorithms for the reduction of the number of points required to represent a digitized line or its caricature,” *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 10, no. 2, pp. 112–122, 1973.
- [148] M.-K. Hu, “Visual pattern recognition by moment invariants,” *Information Theory, IRE Transactions on*, vol. 8, no. 2, pp. 179–187, 1962.
- [149] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, “ROS: an open-source Robot Operating System,” in *ICRA Workshop on Open Source Software*, 2009.
- [150] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT Press, 2012.

-
- [151] W. Wohlkinger and M. Vincze, “Ensemble of shape functions for 3d object classification,” in *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2987–2992.
- [152] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, “Fast 3d recognition and pose using the viewpoint feature histogram,” in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 2155–2162.
- [153] P. J. Ballester and W. G. Richards, “Ultrafast shape recognition for similarity search in molecular databases,” *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, vol. 463, no. 2081, pp. 1307–1321, 2007.
- [154] D. Z. Wang, I. Posner, and P. Newman, “Model-free dynamic object detection and tracking with 2d lidar,” *The International Journal of Robotics Research (IJRR)*, 2015.
- [155] C. E. Rasmussen, “Gaussian processes for machine learning,” 2006.
- [156] K. V. Mardia and P. E. Jupp, *Directional statistics*. Wiley.com, 2009, vol. 494.
- [157] R. Kindermann, J. L. Snell *et al.*, *Markov random fields and their applications*.
- [158] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. N. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. M. Howard, S. Kolski, A. Kelly, M. Likhachev, M. McNaughton, N. Miller, K. Peterson, B. Pilnick, R. Rajkumar, P. Rybski, B. Salesky, Y.-W. Seo, S. Singh, J. Snider, A. Stentz, W. R. Whittaker, Z. Wolkowicki, J. Ziglar, H. Bae, T. Brown, D. Demitrish, B. Litkouhi, J. Nickolaou, V. Sadekar, W. Zhang, J. Struble, M. Taylor, M. Darms, and D. Ferguson, “Autonomous driving in urban environments: Boss and the Urban Challenge,” *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

Appendices

Appendix A

Author's Publications

1. S. Kim, **B. Qin**, Z. J. Chong, X. Shen, W. Liu, M. H. Ang Jr., E. Frazzoli, and D. Rus, “Multi-vehicle Cooperative Driving using Cooperative Perception: Design and Experimental Validation”, IEEE Transactions on Intelligent Transportation Systems, to appear.
2. **B. Qin**, Z. J. Chong, S. H. Soh, T. Bandyopadhyay, M. H. Ang Jr., E. Frazzoli, and D. Rus, “A Spatial-Temporal Approach for Moving Object Recognition with 2D LIDAR”, in International Symposium on Experimental Robotics (ISER), 2014.
3. **B. Qin**, Z. J. Chong, T. Bandyopadhyay, M. H. Ang Jr., E. Frazzoli, and D. Rus, “Learning Pedestrian Activities for Semantic Mapping”, in IEEE International Conference on Robotics and Automation (ICRA), 2014.
4. **B. Qin**, X. Shen, W. Liu, Z. J. Chong, T. Bandyopadhyay, M. H. Ang Jr., E. Frazzoli, and D. Rus, “A General Framework for Road Marking Detection and Analysis”, in IEEE Conference on Intelligent Transportation Systems (ITSC) , 2013.
5. Z. J. Chong, **B. Qin**, T. Bandyopadhyay, M. H. Ang Jr., E. Frazzoli, and D. Rus, “Mapping with Synthetic 2D LIDAR in 3D Urban Environment”, in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013
6. S. Kim, Z. J. Chong, **B. Qin**, X. Shen, Z. Cheng, W. Liu, and M. H. Ang Jr., “Co-operative Perception for Autonomous Vehicle Control on the Road: Motivation and Experimental Results”, in IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS), 2013

-
7. **B. Qin**, Z. J. Chong, T. Bandyopadhyay, M. H. Ang Jr., E. Frazzoli, and D. Rus, “Road Detection and Mapping using 3D Rolling Window”, in IEEE Intelligent Vehicles Symposium (IV), 2013.
 8. Z. J. Chong, **B. Qin**, T. Bandyopadhyay, M. H. Ang Jr., E. Frazzoli, and D. Rus, “Synthetic 2D LIDAR for Precise Vehicle Localization in 3D Urban Environment”, in IEEE International Conference on Robotics and Automation (ICRA), 2013
 9. **B. Qin**, Z. J. Chong, T. Bandyopadhyay, M. H. Ang Jr., “Road Mapping and Topo-metric Graph Learning of Urban Road Network”, in IEEE International conferences on Cybernetics and Intelligent Systems, & Robotics, Automation and Mechatronics (CIS & RAM), 2013
 10. B. Rebsamen, T. Bandyopadhyay, T. Wongpiromsarn, S. Kim, Z. J. Chong, **B. Qin**, M. H. Ang Jr., E. Frazzoli and D. Rus, “Utilizing the Infrastructure to Assist Autonomous Vehicles in a Mobility on Demand Context”, IEEE TENCON, 2012
 11. Z. J. Chong, **B. Qin**, T. Bandyopadhyay, T. Wongpiromsarn, B. Rebsamen, P. Dai, E. S. Rankin, and M. H. Ang Jr., “Autonomy for mobility on demand”, Intelligent Autonomous Systems (IAS) 12 (2013): 671-682
 12. Z. J. Chong, **B. Qin**, T. Bandyopadhyay, T. Wongpiromsarn, B. Rebsamen, P. Dai, S. Kim, M. H. Ang Jr., D. Hsu, D. Rus, and E. Frazzoli, “Autonomy for Mobility on Demand”, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS’12), Video Session, 2012
 13. **B. Qin**, Z. J. Chong, T. Bandyopadhyay, M. H. Ang Jr., E. Frazzoli, and D. Rus, “Curb-Intersection Feature Based Monte Carlo Localization on Urban Roads”, in IEEE International Conference on Robotics and Automation, 2012
 14. Z. J. Chong, **B. Qin**, T. Bandyopadhyay, T. Wongpiromsarn, E. S. Rankin, M. H. Ang Jr., E. Frazzoli, D. Rus, D. Hsu, and K. H. Low, “Autonomous Personal Vehicle for the First- and Last-Mile Transportation Services”, in IEEE International Conference on Robotics, Automation and Mechatronics (CIS-RAM), 2011

Appendix B

Video Links

1. **Autonomous Rebalancers for Mobility on Demand**
<http://www.youtube.com/watch?v=G8y5X-JtXNE>
2. **Autonomy for Mobility on Demand**
http://www.youtube.com/watch?v=666awITIn_o
3. **The Future with Self-Driving Cars**
http://www.youtube.com/watch?v=_3oTYHYblFY
4. **Curb-Intersection Feature Based Monte Carlo Localization**
<http://www.youtube.com/watch?v=XoZK1OS2dTE>
5. **Autonomous Navigation Demo - 2011 July**
http://www.youtube.com/watch?v=_YvPh56p0dk
6. **Visual Difference Between Two Separate Autonomous Runs**
<http://https://www.youtube.com/watch?v=lOxgYmPrMEg#t=107>
7. **Synthetic 2D LIDAR for Precise Localization in the 3D Urban Environment**
<http://www.youtube.com/watch?v=o8Ue7-qEWUQ>
8. **A General Framework For Road Marking Detection and Analysis**
<http://www.youtube.com/watch?v=wkZb6-LXvJQ>
9. **Road Detection and Mapping using 3D Rolling Window**
<http://www.youtube.com/watch?v=fle1MhqL5I>

10. **Moving Object Recognition Using a Planar LIDAR - Highway Scenario**

<http://www.youtube.com/watch?v=T2xbWknsJig>

11. **Moving Object Recognition Using a Planar LIDAR - Campus Scenario**

<http://www.youtube.com/watch?v=Y6XnLUhRxIg>