

# Mapping in Urban Environment for Autonomous Vehicle

Chong Zhuang Jie  
B. Eng (NTU)

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.



---

Chong Zhuang Jie

# Summary

*Keywords:* Autonomous vehicle, mapping, scan matching, localization, activity mapping

Autonomous vehicle has been touted as the next generation automobile that can change the way how people commute from one place to another. There are many benefits of having an autonomous vehicle. In order to allow safe navigation, an autonomous vehicle must maintain knowledge of its surrounding environment at all time.

Using our own instrumented vehicles, mapping is done using a single tilted down LIDAR with odometry information derived from vehicle's Inertia Measurement Unit and wheel encoder. The LIDAR in a push-broom configuration is used to sweep through the environment. By augmenting the raw sensor measurements, we show that it is sufficient to perform navigation in an urban environment by using a map that consists of only curb and intersection information.

This virtual sensors is further generalized by the notion of a synthetic LIDAR. A synthetic LIDAR makes use of the structure from motion of a moving vehicle by maintaining a rolling window. The rolling window reflects the most recent observations of the environment by reconstructing the environment in 3D. This way, meaningful features that can describe an environment can be extracted.

The matching between 2 synthetic LIDAR is performed with an extended correlative scan matching. Usually, a match only consider the point coordinates of an observation. In the extended matching, other features such as intensity and its surface normal relevant to a point sampled from the environment are considered. This scan matching is robust to local maxima and we show how it can be accelerated through Graphic Processing Unit.

---

In order to maintain a consistent map of the environment, observations obtained from synthetic LIDAR are put together using pose graph representation. In a simple graph construction, features from a single rolling window is used and a node in a graph is added at a fixed interval with respect to the width of the rolling window. Optimization of the graph is performed as loop closure is detected between the major nodes, as given by the matching result. The addition of loop closures can be performed in a supervised manner or in a completely automated fashion. The maps produced by different vehicles at different time can be updated by merging the maps together. It is subsequently updated with 3D occupancy grid using Octree representation.

The map is used to perform localization. We performed experiments to evaluate localization performance using curb based and synthetic LIDAR sensor models. We also discussed how the topology of a map can be extracted and represented by a topo-metric graph that can describe a map in a compact representation.

To include non-static objects, we also show how activity mapping can be performed using a known map. In particular, we show how pedestrian’s activity mapping can be done. We also show how semantic information can be extracted using an activity map.

In conclusion, we show that we are able to perform mapping in different kinds of urban environment we have encountered so far. The extension to this work is to include more data analysis on the map to extract complete prior knowledge presented from the environment. This is with the goal that the map is where the knowledge is shared among all other autonomous vehicles, in order to perform a safe and timely navigation.



# Acknowledgment

I would like to thank everyone, my family and friends, who have shaped the course of my research, one way or the other.

I am deeply thankful to my advisors Prof Emilio Frazzoli and Prof Marcelo H. Ang Jr., for their thoughtful insights and guidance along the research works. I am very humble to be able to join the SMART's Future Urban Mobility autonomy group, and given the chance to gain experience on how to handle a huge robotic mobile platform, where it all started with a golf cart.

I thank my friends in the autonomy group. They are many that came and went, but everyone of you have made impacts, big and small to my research and the project. The group members, in particular Tirtha, Nok, Brice, Pei Long, Pratik, Jeong Hwan, Seong-Woo, Tawit, Liu Wei, Xiao Tong, James and Scott, thank you for making the journey a pleasant one. A special thanks to Baoxing, for all the discussions, exchange and sharing of idea since day one of the project.

I would like to dedicate this thesis to my parents and siblings. For the care and love at the same time remain supportive on my research. I am grateful on many occasions I have the chance to get some time off from my research, pausing for the moment before continue to embark on the research journey with blessing, help and guidance.

# Contents

<b>Declaration</b>	<b>i</b>
<b>Summary</b>	<b>ii</b>
<b>Acknowledgments</b>	<b>iv</b>
<b>Table of Contents</b>	<b>v</b>
<b>List of Tables</b>	<b>ix</b>
<b>List of Figures</b>	<b>x</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Autonomous Vehicle . . . . .	1
1.2 Mobility-on-Demand . . . . .	2
1.3 Navigation and Localization . . . . .	3
1.4 Scope of the Thesis . . . . .	6
1.5 Contributions . . . . .	7
1.6 Thesis Outline . . . . .	8
<b>2 Literature Review</b>	<b>9</b>
2.1 Mapping . . . . .	9
2.2 Localization . . . . .	12
2.3 Road Detection . . . . .	14
2.4 Semantic Mapping . . . . .	15

<b>3</b>	<b>Synthetic LIDAR</b>	<b>17</b>
3.1	Road Network . . . . .	17
3.1.1	Curb Points Detection . . . . .	18
3.1.2	Constructions of Virtual Sensors . . . . .	19
3.2	Synthetic LIDAR . . . . .	21
3.2.1	3D rolling window . . . . .	21
3.2.2	Feature Extraction . . . . .	23
3.2.3	Synthetic LIDAR construction . . . . .	25
3.2.4	Probabilistic Characteristic . . . . .	27
3.3	Summary . . . . .	30
<b>4</b>	<b>Mapping with Synthetic LIDAR</b>	<b>31</b>
4.1	Scan Matching with Synthetic LIDAR . . . . .	31
4.2	Mapping with Pose Graph . . . . .	40
4.2.1	Pose Graph Construction . . . . .	41
4.2.2	Automatic Loop Closures . . . . .	42
4.2.3	Overall Algorithm . . . . .	44
4.3	Map Updates . . . . .	45
4.4	Mapping Results . . . . .	48
4.5	Summary . . . . .	53
<b>5</b>	<b>Map application</b>	<b>55</b>
5.1	Localization . . . . .	55
5.2	Road Mapping . . . . .	64
5.2.1	Region growing Method . . . . .	64
5.2.2	Dealing with Noisy Data . . . . .	65
5.2.3	Road Boundary Retrieval . . . . .	68
5.2.4	Post-Filtering . . . . .	68
5.2.5	Road Detection for Point Cloud Segmentation . . . . .	71
5.2.6	Probabilistic Road Mapping . . . . .	71

5.3	Graph Learning of Urban Road Network . . . . .	72
5.3.1	Topo-metric Graph . . . . .	72
5.3.2	Road Skeletonization . . . . .	73
5.3.3	Topological Structure Learning . . . . .	74
5.3.4	Metric Property Learning . . . . .	75
5.4	Summary . . . . .	75
<b>6</b>	<b>Activity Mapping</b>	<b>78</b>
6.1	Gaussian Process for Pedestrian Modeling . . . . .	78
6.1.1	Pedestrian Detection and Tracking . . . . .	79
6.1.2	Track Classification and Clustering . . . . .	80
6.1.3	Activity Learning with Gaussian Process . . . . .	82
6.1.4	Bidirectional Property of Pedestrian Activity . . . . .	84
6.1.5	Activity-based Semantic Mapping . . . . .	85
6.1.6	Pedestrian Path Learning . . . . .	86
6.1.7	Refined Semantics Learning . . . . .	87
6.1.8	Experiment Results . . . . .	90
6.2	Motion Planning with POMDP . . . . .	93
6.3	Summary . . . . .	95
<b>7</b>	<b>Conclusion</b>	<b>97</b>
7.1	Contributions . . . . .	98
7.2	Limitation and Future Work . . . . .	99
	<b>Bibliography</b>	<b>101</b>
	<b>Appendices</b>	<b>115</b>
<b>A</b>	<b>Vehicle Platform</b>	<b>116</b>
A.1	Golf Carts . . . . .	116
A.2	SCOT . . . . .	118
A.3	Coordinate Systems . . . . .	119

A.4 Euler Angles . . . . .	119
A.5 System Architecture . . . . .	120
A.6 Odometry . . . . .	120
<b>B Author's Publications</b>	<b>122</b>

# List of Tables

5.1	Localization error at several marked points . . . . .	60
5.2	Classification accuracy for boundary candidates . . . . .	69
6.1	Mapping results for semantic properties of the four types . . . . .	92

# List of Figures

1.1	A sample RNDF map provided by DARPA . . . . .	4
3.1	Road model for curb detection . . . . .	18
3.2	A typical filter response from a single reading of LIDAR . . . . .	19
3.3	Curb feature classification . . . . .	19
3.4	Construction of virtual LIDAR . . . . .	20
3.5	Virtual sensors . . . . .	20
3.6	A 3D rolling window . . . . .	22
3.7	Environmental reconstruction through rolling window . . . . .	25
3.8	Construction of synthetic LIDAR using surface normal . . . . .	26
3.9	2 different synthetic LIDARs . . . . .	27
3.10	Coordinate frames in a 3D rolling window . . . . .	28
4.1	The mapping framework . . . . .	32
4.2	Multiple look-up table rasterization . . . . .	35
4.3	Comparative examples with extended correlative scan matcher . . .	36
4.4	A wrong matching resolved through scan matching verification . . .	37
4.5	A jump flooding example . . . . .	38
4.6	A JFA with a 500x500 image with step halving process . . . . .	39
4.7	Acceleration of Voronoi construction through GPU. . . . .	40
4.8	Comparison of ECSM matching computation time . . . . .	41
4.9	A Comparison of pose graph optimization with a false loop closure .	42
4.10	Visualization showing the mapping process. . . . .	44

4.11 Mapping of the NUS campus area . . . . .	46
4.12 A merged map . . . . .	47
4.13 Map updates in 2D or 3D . . . . .	48
5.1 A curb map . . . . .	58
5.2 An overview of position estimates . . . . .	59
5.3 Typical particle behaviors . . . . .	59
5.4 Position estimation variance . . . . .	60
5.5 Mapping of the NUS engineering area . . . . .	61
5.6 Synthetic LIDAR localization results . . . . .	61
5.7 Vehicle path estimated using synthetic LIDAR localization . . . . .	62
5.8 Autonomous navigation with synthetic LIDAR . . . . .	63
5.9 Curvature of a road surface . . . . .	65
5.10 Noisy scan rolling window filtering . . . . .	67
5.11 Boundary point adjustment . . . . .	68
5.12 Increased classification accuracy through SVM . . . . .	69
5.13 Results of the SVM classification of road boundary and surfaces . . . . .	70
5.14 Point cloud segmentation example . . . . .	70
5.15 An example of road skeleton map . . . . .	73
5.16 A topo-metric graph map of NUS engineering . . . . .	76
6.1 Pedestrian detection algorithm . . . . .	79
6.2 Snapshot of the onboard pedestrian detector . . . . .	80
6.3 Pedestrian detection module . . . . .	81
6.4 Track clustering results . . . . .	91
6.5 Moving direction of activity model . . . . .	91
6.6 Pedestrian avoidance scenario . . . . .	93
6.7 Pedestrian crossing experiment . . . . .	94
A.1 First instrumented golf cart - Rudolph . . . . .	117
A.2 Second generation golf cart - DJ . . . . .	117



A.3	SCOT . . . . .	118
A.4	System architecture for our autonomous vehicles . . . . .	120

# Chapter 1

## Introduction

### 1.1 Autonomous Vehicle

An early presentation of driverless car was envisioned by highway futurist Norman Melancton Bel Geddes back in 1939 during World's Fair sponsored by General Motors [1]. He depicted that the car of the future could have sensors detecting the electron-magnetic fields that controls both the speed and the path of travel. Not only that, the car is equipped with radios so that nearby vehicles able to keep a safe distance between them while traveling.

For many years, research on fully autonomous vehicle has been pursued internationally. In 2004, the first long distance competition for autonomous vehicle, the DARPA Grand Challenge [2] was held in the Mojave Desert. The competition reflected on how technology has evolved in a way that allows unmanned vehicle to navigate through a vast area while following a designated route. Since then, a series of Grand Challenges [3,4] were held. The DARPA Urban Challenge in 2007 took a step further. In the challenge, a driverless vehicle has to navigate through difficult terrain, involving merging of intersections and avoiding obstacles while obeying to traffic laws.

Self-driving car was made popular with the report of Google's autonomous car project by The New York Times [5]. This is followed by Nevada Legislature passing state law [6] to allow the use of autonomous vehicle on public road, clearing the way where many has seen as the main obstacles that could hinder the progress of the development for autonomous vehicle.

Since then, many of the traditional car manufacturer companies introduced their own version of self-driving car. Showing off their recent development, BMW announced the ConnectedDrive Connect (CDC) [7] by navigating the vehicle using a highly accurate pre-mapped highway in 2012. Audi, on the other hand, has its Audi TTS [8] driving the Pikes Peak autonomously. Meanwhile, Toyota, the giant in the automotive industry announced its Prius Automatic Vehicle Operation System (AVOS) at November 2011 during Tokyo Auto Show [9]. Nissan too announced their autonomous car effort by aiming to be the first to perform public road test in Tokyo using Nissan's LEAF [10]. As recent as 30 July 2014, UK announced that autonomous vehicle will be allowed to perform testing on the road starting from January 2015 [11].

## 1.2 Mobility-on-Demand

The Mobility-On-Demand (MoD) system has emerged as a viable alternative to the conventional use of private transportation [12, 13]. As cities today facing over-saturated traffic demands, the rethinking of a modern solution is born. To meet the ever increasing demand of urban mobility, new type of transportation that is able to supplement the current infrastructure in place is required. As such, the MoD system is introduced as the solution to mitigate the first and last mile problem with the emphasis of being economical and sustainable.

Part of the main design of MoD system is the use of light-weight, short ranged vehicles that is powered by clean energy. Most important of all, a vehicle's ownership deployed under MoD system is both private and public. It is public since the vehicles of such system is provided as part of the MoD service. On the other hand, it can be used as if this is your own private vehicle since the end user of a MoD system is able to pick up a vehicle, drive it to the destination and drop it off. The system is designed to have the comfort of owning a private automobile without the usual burden of owning a private car, which includes costly maintenance, insurance, refueling or the need to provide a parking space.

A MoD system can also supplement and stimulate the use of public transport by providing a convenient means as a first- and last- mile transportation. The

first- and last- mile problem occurs when a user having difficulties of getting from their starting location to a transport central, for example from home to a MRT station and back. This is especially crucial when the distance from train station to home is not within a walking distance. The feasibility of MoD system has been demonstrated in many cities with bike sharing system [14].

One of the key challenges of a MoD system is to always keep a balanced distribution of vehicles. If not managed properly, it is bound to be unbalanced as the demand of the vehicles can be affected by many external events which can be hard to anticipate. An optimal real-time rebalancing policy [15] has been proposed for load balancing in anticipation of the demand. However, transporting the vehicles for the return trips remained an open question. Hence, it is highly desirable to put autonomy in a vehicle. The process can then be handled in an efficient way, in which load rebalancing of a fleet of vehicles can be achieved automatically. Putting autonomy into vehicles can play many other important roles too. Rebalancing the MoD system aside, autonomy allows realization of pick up and drop off at any place. This way, the system can handle virtually any number of stations. This becomes very similar to dynamic one-to-one pick-up and delivery problems [16,17], albeit the system is self-sustained with minimal human assistance.

There are other compelling reasons for enabling autonomous capability in the MoD system: The removal of a human driver for example, could potentially bring improvement towards productivity as a typical American spends an average of 100 hours a year in traffic [18]. Introduction of autonomous car also bring benefits in many other areas too. This includes fewer crashes; alleviate the problem of scarcity of parking area and elimination of constraint on the occupant's state, for example, underage road users and disabled people.

## 1.3 Navigation and Localization

To allow safe and timely navigation of autonomous vehicle from one point to another, an autonomous vehicle must maintain the most updated state of its surrounding at all time, in order to predict and react to the future state by applying necessary control actions. Typically, dead reckoning is available as the basic nav-

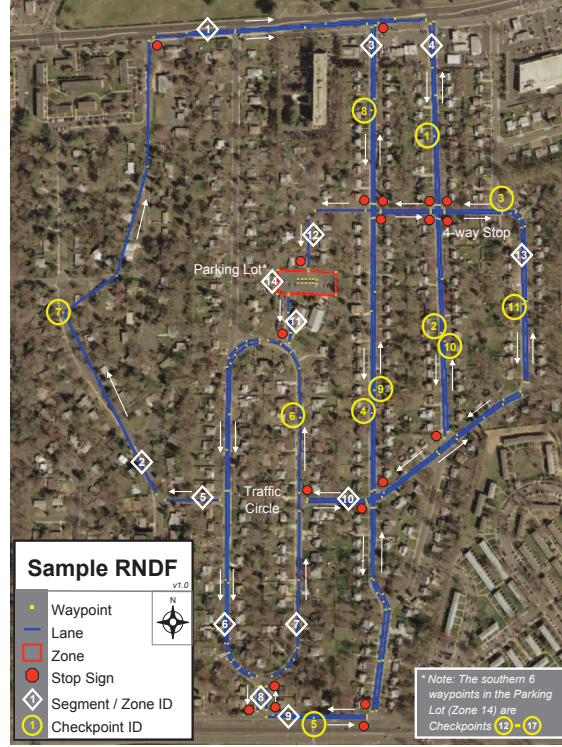


Figure 1.1: A sample RNDF map provided by DARPA

igation information that can be derived from wheel encoders, visual odometry, inertial measurement unit etc. This information alone is sufficient to allow robot to move from one place to another. However, it does not guarantee a collision free traversal, a basic criteria for safe navigation. In the DARPA Urban Grand Challenge (DUGC), a multitude of sensors are used to allow safe maneuver of a vehicle in real traffic environment. To sense the world, all of the finalists have vehicles equipped with a combination of LIDAR, radar and vision camera to observe the environment.

In DUGC, a Route Network Definition File (RNDF) is provided by the organizer, where topology of the road network and parking zones is encoded as GPS waypoints. The RNDF effectively serves as the map as every autonomous vehicle is allowed to travel within the region defined in the map. Figure 1.1 shows an example of the RNDF. In RNDF, the road is generally consists of segments of lanes. Each lane is made up of multiple way points given as GPS coordinates. The lane information can also optionally includes checkpoint, boundary information, stop sign and lane width.

The winning entry of the DUGC, Boss by Tartan Racing team employed 2

separated behaviors to navigate through the urban environment: on-road driving and unstructured driving [19, 20]. During on-road driving, the vehicle is given a desired lane to follow with a stop-line at the end of the lane as the navigation goal. For the unstructured driving (parking maneuver, making U-turn etc.), the goal is represented as the desired pose of the vehicle in the world. In this case, a more capable, 4D lattice planner that searches over vehicle position  $(x, y)$ , orientation  $\theta$ , and velocity  $v$  will invoke to generate a global path to the desired pose.

The other finisher of DARPA Urban Challenge, Tarlos by MIT uses a more general approach to navigate throughout the urban environment. Based on Rapidly-exploring Random Trees (RRT) planning algorithm [21], Tarlos has shown to be able to handle numerous traffic and intersection scenarios that were never tested before. To always guarantee a dynamically feasible trajectory, forward simulation of a closed-loop system which consists of vehicle model and controller is performed. By sampling control inputs to the vehicle model, forward simulation is done to produce many possible paths. Each path is then check for feasibility and finally, the best path is selected by executing the steering controls of the sampled input [22].

In order to perform meaningful navigation, position estimate of a robot is a fundamental requirement. With the advent of the availability of Global Positioning System (GPS) based localization, it seems to be an obvious way to obtain position information. After all, it is now built-in into every modern mobile devices. However, GPS is not without its pitfall. Quality of the GPS is not guaranteed especially in an urban canyon environment as discussed in [23]. GPS signals may suffer from multipath errors due to the presence of tall buildings. More crucially in some cases, GPS signal is not available at all, for example when traveling in tunnels and indoor parkings. These vulnerabilities can be partially eliminated by fusing the GPS signals with other type of information, such as Inertial Navigation System (INS) which fuse the GPS data with IMU information.

Even though there are great progress on GPS and INS based positioning system, other types of localization method is explored in order to achieve accurate positioning. This leads to approaches where localization is done with the vehicle comparing it's own observation against a known environment. Since the vehicle is expected to drive in a known space, localization is typically done in 2 stages. In

the first stage, a manual drive is performed to learn the environment. After which, the model obtained as the result from the first stage can be applied to perform localization.

## 1.4 Scope of the Thesis

There are many challenges involved in the navigation of autonomous vehicle. This thesis addresses one of these challenges, namely that of mapping in urban environment. Mapping of the environment is a fundamental requirement that can significantly affects the ability of an autonomous vehicle to navigate. The focus is on urban environments where the environment is connected with dense network of roads, towns, parks and industrial estates. Thus, the thesis concentrates on building a general framework that allows mapping under different kinds of urban environments.

The first objective of the thesis is to ensure the applicability of the proposed mapping framework to many vehicles. Thus, the thesis focuses on minimalist approach through the use of single planar LIDAR to perform mapping. The single LIDAR, mounted in a tilted-down position at the frontal part of vehicle is used to collect observations of the environment. Then, different types of sensors are synthesized based on this configuration to show the ability of using a single LIDAR to augment obtained observations.

Next, in order to allow mapping to be done efficiently, a specific scenario of matching between synthetic LIDAR is considered that quickly search for best solution between two observations. The normal based synthetic LIDAR shows how two observation at different places can be matched that finally contribute to a consistent map. To show how matching speed can be improved using General Purpose Graphics Processing Unit (GPGPU), NVIDIA's CUDA programming language is used as an example.

The mapping framework is designed to be updated in short-medium term. A well developed urban environment usually contains features that are remained stable enough that a long term mapping update is not required. We assume that changes in the environment are temporary and any changes will be restored thanks

to a well-maintained urban environment.

The resultant map can be used for many purposes. To show that the resultant map can be used over an extended period of time, the same map is used to perform localization repeatedly over a stretch of road in NUS campus environment. In the environment, there can be uncontrollable amount of changes occur over this period of time, for example dynamic movement of traffic agents (vehicles and pedestrians) and different weather conditions (day and night, sunny and rainy days).

To discuss on how further mapping can be done on top of an existing map, the same NUS Engineering environment is used as an example to show how topo-metric graph can be extracted. While activity mapping can be applied to any moving objects within a map, only movement of pedestrian is considered to show how mapping of the pedestrian activity can be done.

## 1.5 Contributions

To allow mapping in urban environment using single LIDAR, the thesis proposes a novel notion of synthetic LIDAR as a virtual sensor using observations obtained from a 2D LIDAR. This is a generalized version from our earlier work that proposed curb and intersection virtual sensors. The synthetic LIDAR is constructed in real time with interest points extracted from a 3D rolling window. This is with the basic assumption that many surfaces in the urban environment are rectilinear in the vertical direction.

To improve the correctness of a scan matching, an extended correlative scan matcher is proposed, where the problem of falling into false global maxima can be avoided. Implementation of scan matching is done using different computing architecture, where CPU and GPGPU approaches are benchmark against each other to show the improved runtime by using GPGPU to perform computation. The mapping framework is built to be robust by using Monte Carlo based loop closure detection to perform place recognition.

With the idea of synthetic LIDAR, we develop algorithms that allows accurate localization in a 3D urban environment. The thesis include experiments to demonstrates its accuracy. For the topo-metric graph mapping, the method described



in the thesis allows road network to be extracted by reusing the generated map. This is with the advantage that accurate metric information detected locally can be obtained.

The activity mapping has been developed with a mobile platform, allowing activity to be recorded while vehicles navigate on a road network. By learning agents' spatial activity model, semantic mapping is performed to extract useful information based on this activity map.

## 1.6 Thesis Outline

In this thesis, a general framework for mapping in the urban environment based on single LIDAR is developed for autonomous vehicle. The generated map is then applied in different ways that is important to allow navigation of autonomous vehicles in urban environment.

After this introductory chapter, Chapter 2 reviews previous work done on mapping, localization, road detection and semantic mapping. Chapter 3 and Chapter 4 presents the construction of synthetic LIDAR and the mapping algorithm that includes mapping results on several places in Singapore respectively. The application of the generated map is presented in the next chapter, where localization and road mapping is discussed. This is followed by the presentation of activity mapping. The final chapter presents the conclusions and suggestions for the future work.

# Chapter 2

## Literature Review

### 2.1 Mapping

To allow vehicles to travel in an urban environment, availability of a precise map representing its surrounding is one of the key components. The DARPA Challenge uses a predefined GPS points as the main guidance to the unmanned vehicles. The GPS point represents the knowledge of the robot's current position and it is used to guide the robot where to go next. The use of the GPS requires guarantee of service level by an external agent: availability of satellite service. Although one could argue that it is sufficient to navigate by using only an external reference system, there are many cases where GPS signals are not reliable or even become totally unavailable. For example indoor environment and underground tunnel.

In contrast, by leveraging on the fact that rich environmental features can be collected by the on-board sensors, it is possible to build a library of knowledge about this particular environment. Often refer to as Simultaneous Localization and Mapping (SLAM), there are 2 different method to perform SLAM in the literature: filtering and smoothing. In filtering, the problem is formulated as an state estimation problem, where it seeks to estimate the robot's current position and the map. The estimation is obtained by first predicting the states and is refined by new observations as they become available. Extended Kalman Filter (EKF), Extended Information Filter (EIF), and Particle Filter (PFs) are among the most popular techniques that falls into this category [24–27].

Murphy [28] introduced Rao-Blackwellized particle filters (RBPF) as a solution

to the SLAM problem. In RBPF, the robot's trajectory and the associated map is represented by each particle. The key idea of the RBPF is to estimate the joint posterior  $p(x_{1:t}, m \mid z_{1:t}, u_{1:t-1})$  about the map  $m$  and the robot's trajectory  $x_{1:t} = x_1, \dots, x_t$ . The joint posterior is estimated given its measurements  $z_{1:t} = z_1, \dots, z_t$  and input control of the robot, typically an odometry  $u_{1:t-1} = u_1, \dots, u_{t-1}$ . The factorization of the above posterior can be done in SLAM for RBPF with the following equation:

$$p(x_{1:t}, m \mid z_{1:t}, u_{1:t-1}) = p(m \mid x_{1:t}, z_{1:t}) \cdot p(x_{1:t} \mid z_{1:t}, u_{1:t-1})$$

The equation allows us to first calculate the trajectory of the robot, then compute the map given that trajectory. This technique is often referred to as Rao-Blackwellized, as this approach provides an efficient computation since the map is strongly dependent on the calculated trajectories.

To calculate the posterior  $p(x_{1:t} \mid z_{1:t}, u_{1:t-1})$ , a particle filter is used, in which the initial proposal distribution is seeded by using the odometry, to be refined later by an external observation. As each particle is incorporated with the most recent observation that is part of the map, a particle that is representative of the robot's trajectory will build a map that is associated with the observation. In other words, every one of the particles carries a unique map since every particle represents a possible trajectory of the robot.

One of the more basic particle filters is Sampling Importance Resampling (SIR). In RBPF, the SIR is used to update the posterior distribution of the robot's trajectory incrementally as new measurements become available while the robot is moving in the environment. The RBPF SIR can be summarized in the following [29]:

1. Sampling: Using a probabilistic odometry model, proposal distribution can be generated using a control input  $u_t^{(i)}$  with the associated particle set at  $x_{t-1}^{(i)}$ . Sampling is then done using the generated proposal distribution before finally obtain the next generation of particles  $x_t^{(i)}$ .
2. Importance Weighting: According to the importance sampling principle, each

particle will be assigned with an importance weight  $\omega_t^{(i)}$  where

$$w_t^{(i)} = \frac{p(x_{1:t}^{(i)} | z_{1:t}, u_{1:t-1})}{\pi(x_{1:t}^{(i)} | z_{1:t}, u_{1:t-1})}$$

3. Resampling: Resampling is done by drawing a finite set of particles with replacement according to the importance weight. It is necessary to apply a target distribution that is different from the proposal distribution and to keep a finite number of particles filters to be used to approximate a continuous distribution.
4. Map Estimation: Based on the trajectory  $x_{1:t}^{(i)}$  of each particle, the corresponding map is estimated using the history of observations  $z_{1:t}$ .

An example of smoothing based SLAM is pose graph mapping. Initially proposed by Lu *et al.* in [34], the seminar work shows that solving the error minimization problem can be done efficiently. By exploiting the inherent sparsity of a map, the optimization-based approach proved to be highly effective in solving large scale mapping problems. To perform pose graph mapping, one could use TreeMap [38], TORO [39], iSAM [40], iSAM2 [41] or  $g^2O$  [42].

The construction of a pose graph is not complete without loop closures. In [35], a simple loop closure detection is used by performing pair wise matching for possible loop closures by considering all observations that are within some bounded radius from the current node. A more elaborate loop closure detection is proposed by Granstorm *et al.* [36]. In the work, a strong classifier is trained using AdaBoost. Then, a positive match is used after performing scan registration. Newman *et al.* in [37] described an automated loop closure detection using a probabilistic measure with sequences of images. After an image is recognized to be a loop closure candidate, the corresponding pair of laser scans is retrieved to obtain the relative transformations.

However, the mapping backends in its native form are not able to reject a false loop closure constraint. This is currently an active research area where methods are being investigated in order to build a robustified pose graph that are able to reject false close loop while recovering true close loops directly on a SLAM backends. The developed extensions to current graph SLAM backends are Switchable Constraints

[43], Max-Mixture Model [44] and Realizing, Reversing, Recovering (RRR) [45]. Sunderhauf *et al.* performs in depth analysis comparing these proposed methods that can perform a robustified pose graph mapping [46].

To support pose graph construction, a scan matching algorithm is usually employed. One of the most pervasive algorithm is Iterative Closest Point (ICP) [30]. In ICP, a reference scan is matched with a query scan by aligning them according to a distance metric. Steder *et al.* [31] performed place recognition by using 3D range data. To allow efficient matching, invariant features are extracted from the 3D range data and stored in database. Another approach is by Normal Distribution Transform (NDT) [32] and Spin Images [33] where matching is done by comparing feature histogram.

## 2.2 Localization

Mobile robot localization is the problem of determining the pose of a robot given a map of the environment [47]. As one of the fundamental requirements to realize vehicle autonomy, there are extensive research for indoor robots localization on a planar surface. However, there are many challenges in developing an accurate, robust and low-cost approach for vehicle localization in an outdoor urban environment.

The fusion of Global Positioning System (GPS) and Inertial Navigation System (INS) to estimate a vehicle's position has become the most popular localization solution in recent years [48–50]. This solution works well in open areas; however, it is not suitable for dense urban environment where GPS signals suffer from satellite blockage and multipath propagation problem [51]. To alleviate this problem, road-matching algorithms are proposed, where a prior road map is used for determining motion constraint to update the localization estimation [52, 53]. While this solution achieves a good global localization, it is not designed for precise estimation relative to the local environment.

Map-aided algorithms are proposed for highly precision localization using local features. In [54], single side curb features are extracted by a vertical LIDAR to build a boundary map in the form of line segments to improve vehicle localization. In [55],

lane markers are used as local features from the reflectivity values of LIDAR scans, where a digital lane marker map is used as prior. The performance of the algorithm is similar to those in [54]. While these algorithms reduce lateral localization error considerably, they suffer from longitudinal accuracy.

In another work by Hentschel *et al.* [56], the same approach is used. However, instead of using a self learned map, the prior knowledge is obtained from OpenStreetMap. This effectively eliminates the learning stage since the map is readily available. To perform matching, virtual 2D ranging extracted from 3D point clouds generated from a rotating LIDAR is used [57]. Then, particle filters is used to provide position estimation by coupling together the data from GPS and LIDAR. Propagation of particles is done by using GPS points as the control input and the particles are weighted according to the evidence obtained from the LIDAR map matching before a final estimate of its position is obtained.

Levinson *et al.* in [58,59] 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 by extracting points from road surface. Then, the reflectivity measurements of these points are correlated to a map containing ground reflectivity to update particle weights. One assumption underlying this algorithm is that road surfaces remain relatively constant, which may undermine the performance in some cases. Besides, the need for a costly 3D LIDAR sensor limits its usage.

Baldwin *et al.* in [60] utilizes accumulated laser sweeps as local features. The algorithm first generates a swath 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 always desired when the localization is integrated with other navigation functions. Similarly in [61,62], a 3D point cloud of the environment is obtained by servoing a 2D LIDAR, and a reduced 2D feature is used to perform localization. This algorithm has been shown to work well in an indoor environment with a well structured ceiling features. In [63], a microwave radar sensor is used

to perform SLAM. While the radar has the ability to “see through” obstacles, association of radar targets is a complex task and SLAM processing is done offline.

## 2.3 Road Detection

Road detection is one of the fundamental requirements for autonomous vehicle driving on urban roads. Road surfaces are the traversable areas that give strong prior where vehicles can safely navigate through. Road detection can be used directly for vehicle control and path planning. In addition, it can also provide contextual information to help other perception processes. Current researches related to road detection can be categorized as lane marker detection, and road boundary detection.

Lane marker detection has been studied in the context of Autonomous Driver Assistance Systems (ADAS). Researchers aim to detect and locate lane markers on the road, and utilize the results for lane departure warning, adaptive cruise control and other purposes. Vision has been the most popular sensory modality for lane marker detection. Aly in [64] presents a real-time algorithm to detect lane markers in urban streets. In the work, an Inverse Perspective Mapping (IPM) process is first applied to generate a bird’s eye view of the road surface from the original image, then the transformed image is filtered and thresholded to extract pixels belonging to vertical lanes. Hough Transform and Random Sample Consensus (RANSAC) spline fitting are conducted to get a mathematically compact representation for the detected lanes. Similar works can be found in [65–67]. While vision-based lane detection has shown to be able to achieve good performance, it can be vulnerable to challenging light conditions like shadows and highlights, due to the passive nature of perception vision. On the other hand, there are other studies which use active sensors, particularly by using the intensity values obtained from LIDARs to perform lane marker detection. Related work can be found in [68, 69].

While lane marker detection is only applicable for structured road where markers exist, road boundary detection can be applied for both structured and unstructured road. LIDARs have played a dominant role in this area. The work from Thrun *et al.* uses LIDARs to perform Probabilistic Terrain Analysis (PTA) in

desert driving [70]. The driving terrain is represented by a 2D grid map, and the grids are then classified into different terrain types according to the height difference in their local neighborhood. Zhang proposes a road boundary detection algorithm for urban roads in [71]. A Gaussian differential filter is applied to the range values of each laser scan, and road boundary points are extracted as local maximal of the filter response. Other similar researches can be found in [72–74]. One common feature of the above algorithms is that they process individual 2D scans for road boundary detection.

## 2.4 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 is commonly used to facilitate human-robot interaction. More recent works started to focus on how it can be used to further enhance a robot’s capability to perform advanced reasoning and planning. In the past few years, various methods have been proposed for semantic mapping. Depending on the type of semantic information, these methods can be roughly group into three categories: appearance-based approach, object-based approach, and activity-based approach.

The appearance-based approach is the most popular approach in the research of semantic learning, where semantic knowledge is acquired by interpreting appearance features from sensory data. In [75], 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 [76]. In [77], 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. Some other appearance-based semantic mapping approaches can be found in [78, 79].

Unlike the appearance-based approach 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. In [80], Galindo *et al.*



infer the semantic type of a room by detecting the typical objects in it. In [81], Vasudevan *et al.* propose to perform place classification using not only object count information, but also the position relationship between objects.

The activity-based approach is to learn the semantic knowledge of an environment based on agent activities in it. Compared to the extensive literature for the appearance-based approach, relatively few are found in this category. In [82], Wolf *et al.* build a semantic 2D grid map according to the occupancy of the space by dynamic entities. Activity-related features are used to classify a place into two semantic types, “street” or “sidewalk”. In [83], Xie *et al.* present a method to localize functional objects that affect people behavior in surveillance videos.

# Chapter 3

## Synthetic LIDAR

An accurate estimation of a vehicle’s position is a fundamental requirement for autonomous driving. In this thesis, a map-based localization is developed for autonomous navigation. We propose the use of single LIDAR to perform mapping. However, measurements obtained from a single LIDAR are sparse as it only measures one 2D layer at a time. The following chapter describes how a single LIDAR can be used to augment the observed data in such a way that it is expressive enough for map building in order to perform autonomous navigation.

### 3.1 Road Network

An urban environment typically contains expansive road networks that are unique to different cities around the world. Streets in the city typically contains pedestrian sidewalk and road surface separated by curbs. The curbs provide rich information of the road network both topologically and metrically [84]. As one of the most dominant features in an urban environment, the curb provides excellent lateral information of the vehicle while traveling on a straight road, while a cornering road can provide vehicle’s position longitudinally.

The intersections on the other hand, where there are no curbs exist, carries rich longitude information. This is the main motivation on using only curbs and intersections as the only observable features. This section shows a single LIDAR can be used to detect the curbs and intersections.

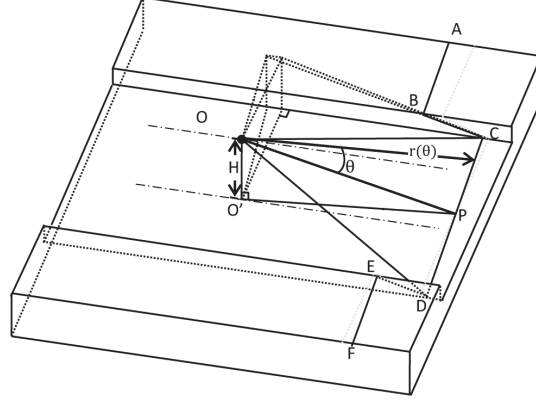


Figure 3.1: Road model for curb detection

### 3.1.1 Curb Points Detection

A typically urban street contains road surface and curb plane, as depicted in Fig. 3.1. The lines,  $AB$ ,  $BC$ ,  $CD$ ,  $DE$ ,  $EF$  represents the observed measurements from a single plane LIDAR located at point  $O$ . The extraction of curb points ( $BC$  and  $ED$ ) started with a second-order differential filter, which is given by:

$$r_f(\theta) = \sum_{i=-5}^{i=-3} r(\theta + i \times \mu) + \sum_{i=3}^{i=5} r(\theta + i \times \mu) - \sum_{i=-2}^{i=0} r(\theta + i \times \mu) - \sum_{i=0}^{i=2} r(\theta + i \times \mu) \quad (3.1)$$

where  $\mu$  is the angular resolution of the LIDAR and  $\theta \in [-\pi/2 + 5\mu, \pi/2 - 5\mu]$ . To obtain the edges where transitions occur from one surface to another, local minima/maxima points are used to extract segments of scan. Fig. 3.2 shows one typical scan with filter's response.

Next, to extract scan segments that are belong to curb points, the edge points  $C$  and  $D$  are first selected by locating center of 2 sequential edges that are closest to the sensor origin,  $O$ . The curb lines,  $BC$  and  $ED$  can then be determined. This simple model allows measurement of the curb to be extracted. However it does not always correspond well in practice. There could be a moving vehicle or a pedestrian passing by resulting in a false detection. Hence, a series of verification processes is performed by adding constraints to ensure that only real curb points

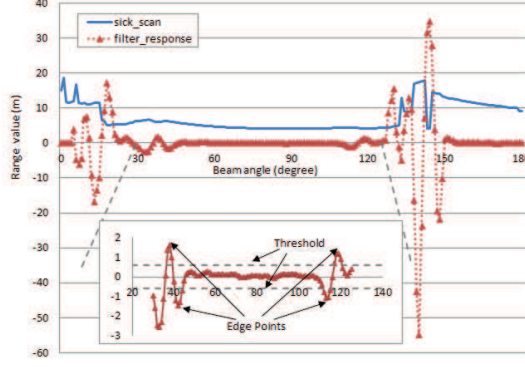


Figure 3.2: A typical filter response from a single reading of LIDAR

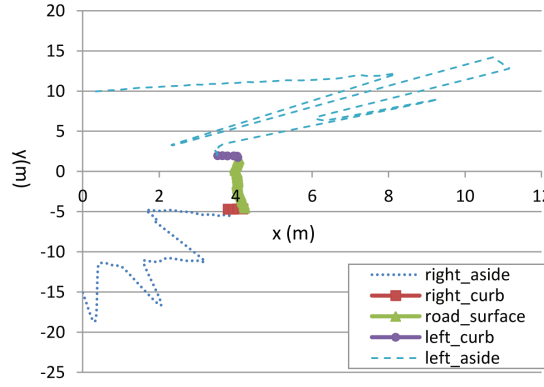


Figure 3.3: Curb feature classification

are extracted.

To ensure moving objects are not included as part of the observation, the length segment  $CD$  must be larger than an expected road width. This filter is strengthened by ensuring that segments  $BC$  or  $ED$  should be below a height threshold and the number of laser beams on  $BC$  or  $ED$  should be more than a chosen height. Only when all these criteria is satisfied, the segment is thought to be valid and included as the detected curb point. Thus most noise like vehicles and pedestrians is filtered. One typical classification result is shown in Fig. 3.3

### 3.1.2 Constructions of Virtual Sensors

With the curb measurements extracted, construction of virtual sensors are performed, namely the curb sensor and intersection sensor. The construction of curb sensor is obtained directly from the result of curb measurements. When a new LIDAR obtains a new measurement, the curb segments,  $BC$  and  $DE$  are projected onto a virtual flat ground, forming the initial part of the virtual laser beam. The



## 3.2 Synthetic LIDAR

In the previous section, curb and intersection features extraction using single LIDAR is presented. Although this is sufficient to perform navigation as we will show in Chapter 5, the application is limited to environments that are similarly structured that contains curbs and intersections.

To allow safe navigation, adaptability of a virtual sensor responding to different types of environment in urban scenario is essential. Therefore a much richer set of features should be used. While a tilted down LIDAR allows detections of the road boundary, this configuration when set in motion is making a cylindrical sweep throughout the whole region that falls within the detection range of a LIDAR. In other words, a titled down LIDAR in motion able to cover a large surface area of the environment. This allows a full reconstruction of an environment where a much richer set of features can be used.

To enable feature recognition of an environment, an accurate model of the world is required. Different from conventional methods which uses nodding LIDAR and Velodyne, we show that how a fixed, tilted down single planar LIDAR can be used to enable the reconstruction of the environment accurately. The main motivation of using only single planar LIDAR is with the emphasis of minimalist design principle. Not only a rigidly mounted single beam LIDAR costs less comparing with other laser based sensors, this also allow one to design algorithms that is effective and efficient since only minimal sensory data need to be processed.

We proceed to show how a single fixed LIDAR is used to perform feature extraction in real time specific to this configuration.

### 3.2.1 3D rolling window

To reconstruct the environment, a rolling window sampling is used by maintaining a variable amount of scan lines. The scan lines are collected and stored as point clouds, where a point in the environment is defined by its  $x, y, z$  coordinates. By using a rolling window, the point clouds form a collection of observations that combines temporal features with high probability of reflecting more recent samples by the ranging sensor. In a way, a 3D rolling window is used to accumulate different

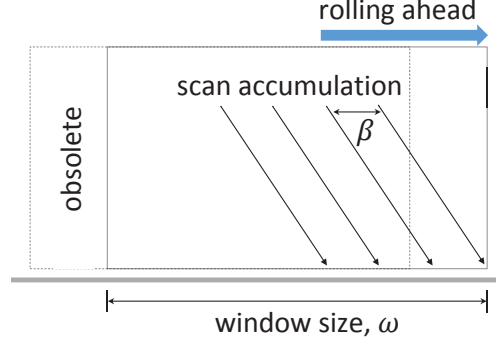


Figure 3.6: A 3D rolling window

scans recorded in a short distance. The size of the window is flexible and the rolling window forms a local map of the environment, i.e., it rolls together with the vehicle, where new incoming scans actively added into the window, and the old samples get discarded.

A 3D rolling window is defined as the following: Given the window size  $w$ , the points  $p$  in  $n$ -th scan,  $P_n$  is accumulated according to

$$P_n = \bigcup_{k=n-\omega} \{p_k, \dots, p_n\} \quad n > \omega \quad (3.2)$$

As shown in Fig. 3.6,  $\omega$  is used to control the number of accumulated scans such that the size of the window is fixed. Also, a new scan is only inserted when a sufficient distance, denoted by  $\beta$ , is achieved. Consequently, a small  $\beta$  will have denser points but the overall window size will become shorter. Conversely, a large  $\beta$  can reconstruct a larger area but with sparser points.

The rolling window works in the odometry frame of the system, where each scan from a physical LIDAR is projected based on the odometry information derived from IMU and wheel encoder as discussed in A.6. Although this means that some form of odometry is required, (usually provided by IMU and wheel encoders, see Appendix A.6) these sensors are becoming standard parts for modern vehicles equipped with anti-lock braking system (ABS) and electronic stability control (ESC). Thus, odometry information can be obtained without incurring additional cost.

A LIDAR usually output a range data consist of  $K$  total ranges. Each range data  $r_k$ ,  $k \in \{0 \dots K\}$  is sampled at a fixed angle increment with a fixed amount

of time  $\Delta_t$ . To perform LIDAR projections, a constant motion is assumed where transformation between each point in the LIDAR is interpolated linearly and spherically. Then, each point of the LIDAR can be transformed into the odometry frame given the transformations of the vehicle base,  ${}^O_B T_n$ .

### 3.2.2 Feature Extraction

In this section, the reconstructed environment is further analyzed to provide salient features that can be used to uniquely define an environment. Two kinds of features are discussed: normal surface and intensity. The difference between these two features is a prominent one, the first feature is purely geometrical based, while the other is texture based.

#### Normal surface

To extract features that are perpendicular to the ground, estimation of surface normal is used. While many method exists [85], we used normal estimation proposed by [86]. It is based on a 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  $P^K$  [87]. 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

$$d_i = (p_i - x) \cdot \vec{n} \quad (3.3)$$

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  $P^K$  [88]:

$$C = \frac{1}{k} \sum_{i=1}^k (p_i - \bar{p}) \cdot (p_i - \bar{p})^T, C \cdot \vec{v}_j = \lambda_j \cdot \vec{v}_j, j \in \{0, 1, 2\} \quad (3.4)$$

Where  $k$  is the number of points in the local neighborhood,  $\bar{p}$  as the centroid of the neighbors,  $\lambda_j$  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  $\lambda_0$ . 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.

To find the local neighborhood points efficiently, KD-tree [89] is built from all the points obtained from the rolling window, then a fixed radius search at each point can be performed efficiently. 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, where

$$P_{n+1}^\phi = P_n^\phi \cup \Phi(P_{n+1} \setminus P_n) \quad (3.5)$$

$\Phi$  can be any points classification function,  $P^\phi$  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 perform on the newly accumulated point cloud and the processed points from the previous instance can be reused.

### Intensity

A modern LIDAR also measures the intensity return from a laser beam. Fig. 3.7 shows a reconstructed environment using false color based on the intensity return. This is especially useful in extracting meaningful textual information from road surface.

A naïve approach is to simply perform a global threshold such that any objects with high reflectivity can be used as the feature. The other more comprehensive feature extraction is to use techniques developed from vision community. For example adaptive threshold, edge detection, etc.

To take advantage of common image processing pipeline, each scan point obtained by the LIDAR can be aligned in a more conventional pixel method, with some loss of geometry information. In this case, an image can be composed by generating a 2D image with its horizontal axis as angular increment and vertical axis as accumulated distance. In other words, each point collected in the rolling windows is transformed to an image frame by simply copying the point buffer to

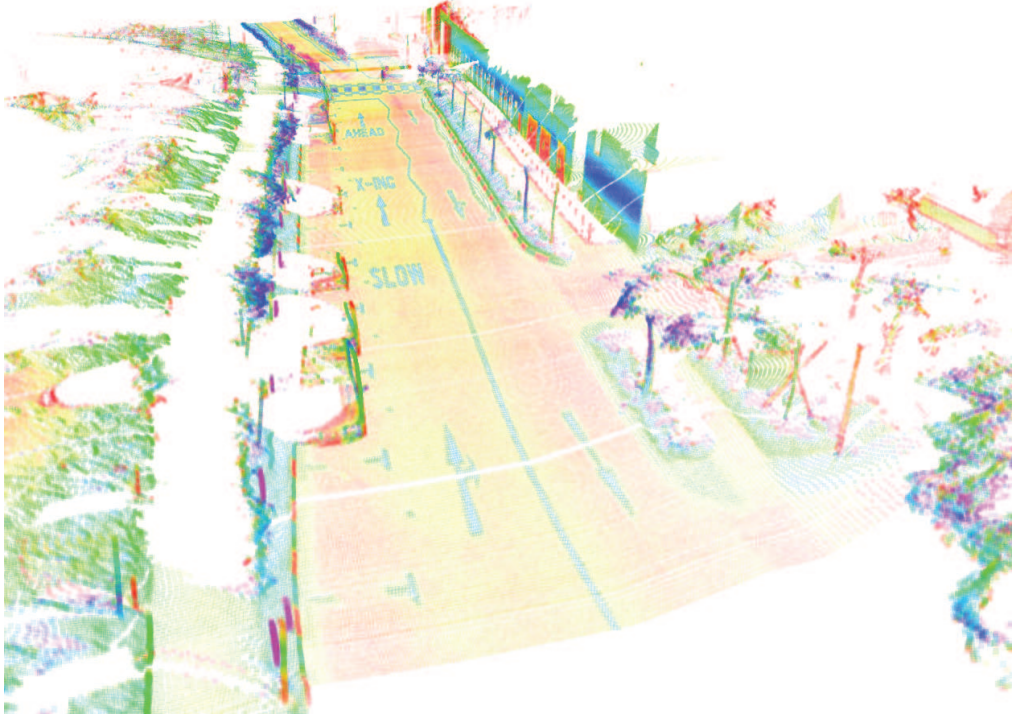


Figure 3.7: Environmental reconstruction through rolling window. The false color reflects the return of intensity value by the LIDAR

an image buffer. This process involves only memory transfer, hence the resultant frame can be generated immediately.

The frame is then passed through an edge detection operator for feature extraction. The Canny edge detector was found to be effective in detecting distinct road marking. The filtered image can then be used as a binary mask to perform point classification in a rolling window.

### 3.2.3 Synthetic LIDAR construction

The result from the classified points consists of collection of interest points in 3D. For the construction of synthetic LIDAR, the interest points in 3D are projected into a virtual horizontal plane parallel to  $X-Y$  plane. It can be seen that this synthetic LIDAR has a very special feature: the ability to “see through” the obstacles. This is possible since feature extractions are performed in 3D point clouds.

This synthetic LIDAR is comprehensive in the sense that it is as if having multiple short ranged LIDARs arranged at different height at a forward facing configuration, with the height adapted to wherever there exists a vertical surface

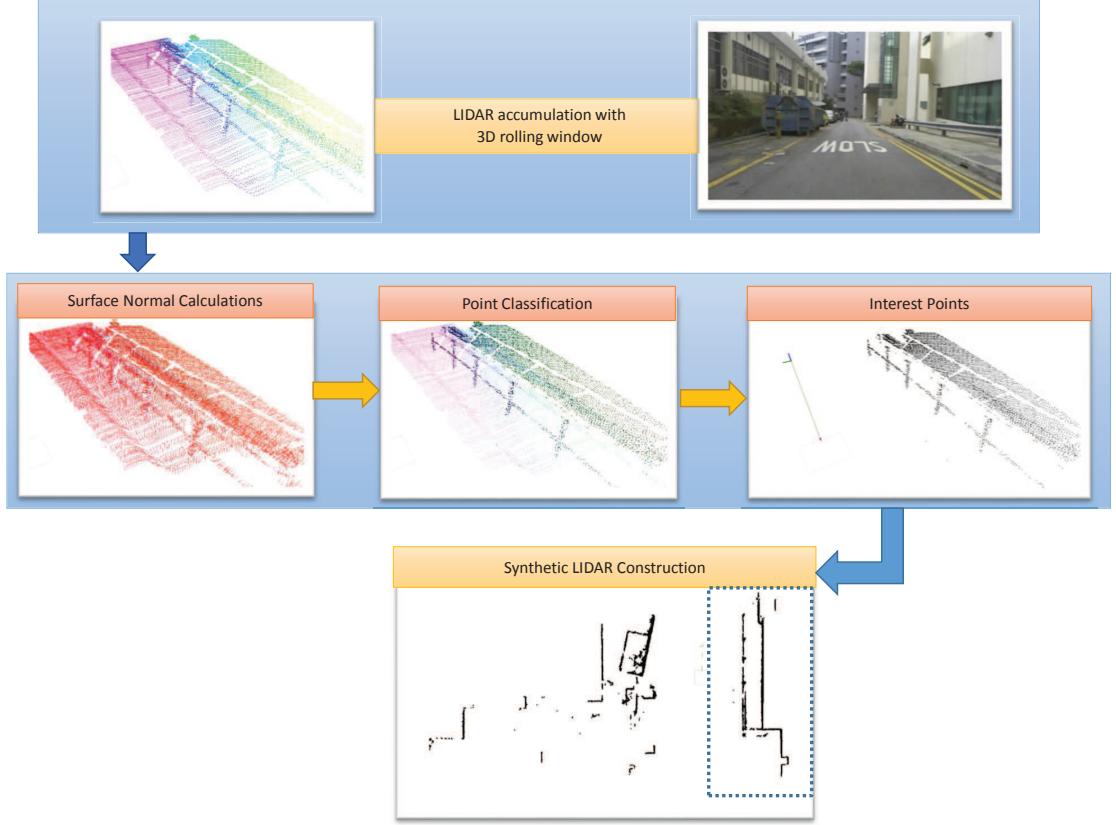


Figure 3.8: Construction of synthetic LIDAR using surface normal

in the environment. The construction of synthetic LIDAR is completed by inserting the virtual sensor's origin at the base of the vehicle and perform transformation of all the interest points from odometry to this reference frame.

In many applications where a standard LIDAR is desired (equally spaced angle increment), the synthetic LIDAR can be further reconstructed to fulfill this constraint. This would involve performing ray tracing at each fixed angle incrementally and obtain minimum range value from the possible end points. The overall 3D perception is summarized in Fig. 3.8. The 3D perception is done with the Point Cloud Library [90].

Fig. 3.9 shows an example of a constructed synthetic LIDAR by extracting different features by using a 3D rolling window. Fig. 3.9(a) shows the result using method described in 3.2.2. In the figure, texture information of the road surface can be observed, where zebra crossing is featured prominently in the top part of the image. The second figure, Fig. 3.9(b) shows the synthetic LIDAR constructed using surface normal. It can be seen that the road boundary is clearly outlined with the building facade located at the right side of the road. The vegetation features,

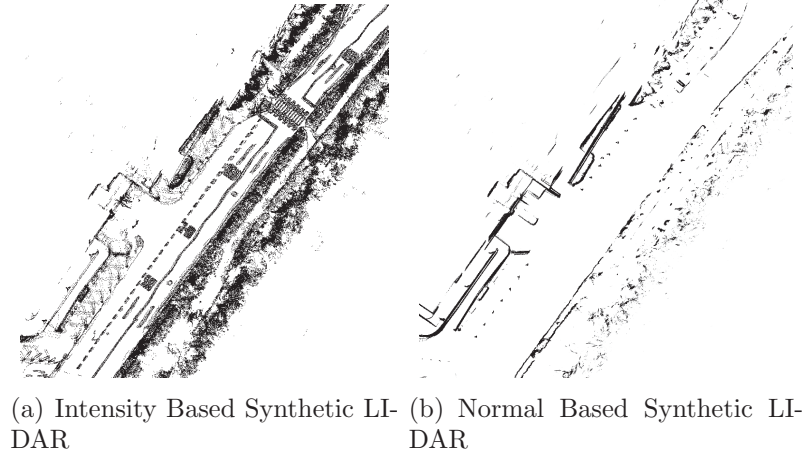


Figure 3.9: 2 different synthetic LIDARs

tree trunks and branches are recognized as part of the feature from normal surface extraction.

### 3.2.4 Probabilistic Characteristic

In order to understand how the accumulated points can be affected by a moving vehicle, probabilistic characteristics of the accumulated 3D data is derived. A general analysis is given for the accumulation process followed by an example on how a point is distributed along z-axis.

Fig. 3.10 shows different coordinate frames that is used to determine the probabilistic characteristic of the accumulated points. During the accumulation process, each point originated from *LIDAR* frame is transformed into *Base* frame and projected into odometry frame according to rolling window. When accumulation is done, the points in a rolling window are then transformed into *VBase* frame, for the eventual reconstruction. This can be expressed as:

$${}^{B_v}p_n = {}^B_B T_n {}^B_L T_n {}^L p_n, \quad (3.6)$$

where  ${}^{B_v}p_n = (x_n, y_n, z_n, 1)^T$  is the augmented points  $p_n$  in  $VBase_t$  coordinate frame, and  ${}^B_B T_n$  denotes the  $4 \times 4$  homogeneous transformation matrix relating the frame  $Base_t$  to frame  $VBase_t$ , Transformation matrix  ${}^B_L T_i$  gives the relative frame  $LIDAR_n$  relative to frame  $Base_n$ . Hence, distribution of  ${}^{B_v}p_n$  is affected by  ${}^B_B T_i$  and  ${}^B_L T_i$  matrices. Assuming that the LIDAR is rigidly attached to the *Base*

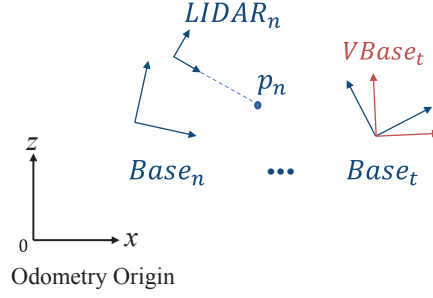


Figure 3.10: Coordinate frames in a 3D rolling window

frame, then the distribution of  ${}^{B_v}p_n$  can be described by  ${}^{B_v}T_i$ , which estimates the vehicle pose.

Let  ${}^{B_v}X_n$  represent the position of *Base* with respect to  $Base_v$ , and let  ${}^{B_v}\bar{X}_n$  as the estimated value for  ${}^{B_v}X_n$  from vehicle odometry system, the probability distribution function of  ${}^{B_v}X_n$  can be approximated by a Gaussian Distribution with mean value as  ${}^{B_v}\bar{X}$ , and the covariance matrix  ${}^{B_v}\Sigma_n$ :  ${}^{B_v}X_n = (x, y, z, \alpha, \beta, \gamma)^T$ ,  ${}^{B_v}X_n \sim N({}^{B_v}\bar{X}_n, {}^{B_v}\Sigma_n)$ .

Vehicle orientation is represented in Euler angles as discussed in Section A.4.  ${}^{B_v}\Sigma_n$  reflects the uncertainty of the dead-reckoning system, increasing with vehicle driving distance and turning angle. When the LIDAR mounting is fixed,  ${}^{B_v}p_n$  is only a function determined by  ${}^{B_v}X_n$ , which is  ${}^{B_v}p_n = f(x, y, z, \phi, \theta, \psi)$ . By linearizing the function  $f$ , distribution of  ${}^{B_v}p_n$  can be represented as a Gaussian Distribution:

$${}^{B_v}p_n \sim N({}^{B_v}\bar{p}_n, \Sigma_p), \quad (3.7)$$

Where

$$\begin{aligned} {}^{B_v}\bar{p}_n &= f(\bar{x}, \bar{y}, \bar{z}, \bar{\phi}, \bar{\theta}, \bar{\psi}) \\ \Sigma_p &= F {}^{B_v}\Sigma_n F^T \end{aligned} \quad (3.8)$$

$F$  is the Jacobian Matrix of function  $f$  with respect to  ${}^{B_v}X_n$ . The full distribution for single point  ${}^{B_v}p_n$  can be obtained with the above equation. Let's focus our derivation for point distribution along  $z$  axis as an example. Since the tilted-down LIDAR is affixed at frontier of the vehicle and its mounting position is  ${}^B_LX_n =$

$(x_l, y_l, z_l, 0, \theta_l, 0)$ , which determines the transformation matrix  ${}^B_v T$ . Then we have:

$$z_n = f_z(\theta, \psi, x_l, y_l, z_l, \theta_l, r, k) \quad (3.9)$$

Where

$$\begin{aligned} f_z = & -\sin \theta (\cos \theta_l (r \cos k) + x_l) + \cos \theta \sin \psi ((r \sin k) \\ & + y_l) + \cos \theta \cos \psi (-\sin \theta_l (r \cos k) + z_l) + z. \end{aligned}$$

$r$  denotes the range value in laser scan, and  $k$  is the laser beam angle. To calculate the variance of  $z$ ,  ${}^B_v \Sigma_n$  can be simplified into a diagonal matrix similar to [70], keeping in mind that these six state variables can be correlated  ${}^B_v \Sigma_n = \text{diag}(\sigma_x^2, \sigma_y^2, \sigma_z^2, \sigma_\phi^2, \sigma_\theta^2, \sigma_\psi^2)$ .

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

$$\begin{aligned} F_z = & (0, 0, 1, 0, \frac{\partial f_z}{\partial \theta}, \frac{\partial f_z}{\partial \psi}) \\ \approx & (0, 0, 1, 0, -r \cos k \cos \theta_l - x_l, r \sin k + y_l). \end{aligned} \quad (3.10)$$

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

$$\begin{aligned} z_i & \sim N(\bar{z}_i, \sigma_{z_i}^2), \\ \bar{z}_i & = f_z(\bar{\theta}, \bar{\psi}, x_l, y_l, z_l, \theta_l, r, k), \\ \sigma_{z_i}^2 & = \sigma_z^2 + (r \cos k \cos \theta_l + x_l)^2 \sigma_\theta^2 \\ & \quad + (r \sin k + y_l)^2 \sigma_\psi^2 \end{aligned} \quad (3.11)$$

Given the fact that the width of the rolling window is small, and the road surface is generally horizontal,  $\sigma_z^2$  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, roll and pitch values are inferred directly from IMU. Their variances are approximated as:  $\sigma_\psi = a_1 \frac{d\psi}{dt} + a_2 \psi$ ,  $\sigma_\theta = b_1 \frac{d\theta}{dt} + b_2 \theta$  where  $a_1, a_2, b_1,$

$b_2$  are fixed parameters.

### 3.3 Summary

In this chapter, synthetic LIDAR is introduced. We discussed how information of road network can be extracted using single tilted down LIDAR. The curb points in particular, can be uniquely identified and thus detection can be done. We showed how a collection of curb points can be used to form a virtual sensor. The curb sensor is constructed by projecting a small sequence of curb points onto a virtual plane using vehicle's odometry. The intersection points, working in tandem with the lack of curb points, form the second virtual sensors that describes the road network.

Synthetic LIDAR generalizes the curb and intersection virtual sensors, where features in the environment are determined by a set of descriptors. Using rolling windows to reconstruct the environment in point clouds, interpretation of the surrounding is done in 3D to produce a unique fingerprint for each different places. Examples are included on how synthetic LIDAR can be constructed using normal and intensity features. Finally, we conclude the chapter by characterizing the probabilistic nature of a rolling window to understand the possible source of uncertainty from the collected observations.

# Chapter 4

## Mapping with Synthetic LIDAR

The mapping process starts with the acquisition of raw sensor data produced by 2D LIDAR, encoders and IMU. The raw sensor data is then processed through a real time 2D synthetic LIDAR, as discussed in chapter 3. In this chapter, we discuss how these features are used to build a consistent map of the environment.

The overall mapping process is depicted in Fig. 4.1, where it can be separated into two processes, front-end and back-end. The front-end usually has different modules depending on the type of sensor used, the back-end usually consist of least-square optimization depending on the information matrix given by the front-end.

In the following discussion, we show how scan matching works together with loop closure detection to generate a graph which form part of the front end. The graph is then used by the back-end to generate a globally consistent map, while optionally performing loop constraint rejection.

### 4.1 Scan Matching with Synthetic LIDAR

The correlative scan matcher (CSM) [91] is a family of cross-correlation scan matching algorithms. It employs probabilistic framework to search for a rigid transformation that maximizes the probability of having the observed data. While many matching algorithm exist, as discussed in Section 2.1, correlative scan matching is robust to large initialization error while taking advantage of fast computation using lookup-table rasterization.



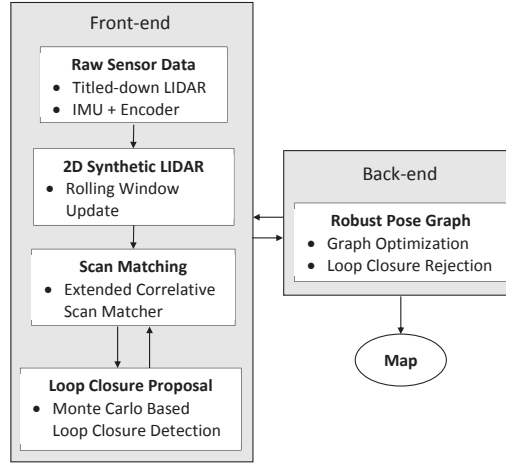


Figure 4.1: The mapping framework

To understand how a correlative scan matcher works, let's start by having a robot moving from one point  $x_i$  to another point  $x_{i+1}$ . Then, the robot position, given some motion input  $u$  and observation  $z$  that is dependent on environment model  $m$ , the posterior distribution of the robot's position is given by  $p(x_i|x_{i-1}, u, m, z)$ . Applying Bayes' rule, the posterior can be written as

$$p(x_i|x_{i-1}, u, m, z) \propto p(z|x_i, m)p(x_i|x_{i-1}, u) \quad (4.1)$$

While the second term  $p(x_i|x_{i-1}, u)$  is usually known as multivariate Gaussian distribution, the observation model is usually complex in nature and can contain many local extrema. By assuming each point in the synthetic LIDAR is independent, we have:

$$p(z|x_i, m) = \prod_j (z_j|x_i, m) \quad (4.2)$$

To approximate the probability distribution for the whole synthetic LIDAR points, ray casting calculation, occlusion and visibility is neglected and thereby allowing us to approximate  $z_j$  in terms of Euclidean distance from a surface  $m$ .

### Lookup-table Rasterization

The rasterization process starts with a map  $m$ . Then for each point observed by the LIDAR, the conditional probability that a feature is detected can be calculated.

The computation of  $p(z|x_i, m)$  is accelerated by using a lookup table, where a log probability is computed for each of the observed points. Since the lookup table must be view point independent, it makes sense that the shape of the distribution has a radially-symmetric function. More specifically, in each  $(x, y)$  position in the world, we compute the probability of the nearest distance selected from LIDAR points under a zero-centered Gaussian distribution with a variance  $\sigma_{hit}^2$ .

### Optimization through Multi Resolution

Correlation method at its simplest implementation involve brute-force search over 3D  $x, y, \theta$  volume. This can be very costly. At each evaluation on  $p(z|x_i, m)$ , a single point is expected to perform the following procedures:

- Projection of points into a selected 3D slice
- Calculating matching score using the lookup table
- Memorize the value

A multi resolution can reduce the look up table process significantly, hence significant acceleration can be achieved. The correlation process then becomes:

- Evaluate  $p(z|x_i, m)$  over desired search window at low resolution
- Denote the 3D slice that has the highest probability
- Evaluate  $p(z|x_i, m)$  inside this slice at higher resolution

The steps above can be cascaded at different resolutions to achieve desired search speed while avoiding falling into local maxima. In practice, 3 different resolutions are used that achieve a good balance between speed and the search of the global maxima.

### Multiple Lookup-table Rasterization

Instead of using only point information in the lookup-table  $m$ , other features associated with the point can be included in multiple lookup-table  $m_n$ , where  $n$  corresponds to the number of different features. In a single lookup-table rasterization, each point  $m_i$  in a map contains a value which describes the conditional

probability that a nearby point  $p_i$  can be observed. In a multiple lookup-table rasterization, other features of the point  $p_{n_i}$  are included. This can either be the point's curvature, normal, depth/height, or color value. To encode an additional lookup-table for a different feature, the nearby point  $p_i$ 's feature value is used. Loosely speaking, the feature value from a nearest point is written to an additional lookup-table, taking the value at  $m_{n_i}$ . The observation model is then becomes:

$$p(z|x_i, m, m_n) = \prod_j p(z_j|x_i, m, m_n) \quad (4.3)$$

The additional term  $m_n$  represents the new features that are encoded in the additional look-up table. To obtain the value of  $P(z_j|x_i, m, m_n)$ , a weighted sum of each feature can be used:

$$p(z_j|x_i, m, m_n) = \sum_n w_n p(z_j|x_i, m) f(z_j, x_i, m_n) \quad (4.4)$$

where  $\sum w_n = 1.0$  and  $f(z_j, x_i, m_n)$  is an evaluation function that has value of  $[0, 1]$ . Should none but only the Euclidean point information is used, by having  $f(x_i, m_n) = 1.0$  the equation above becomes equivalent to Eq. 4.2.

Different evaluation functions are used for specific types of additional features that is to be encoded. In this work, surface normal of a point is used in addition to a point's euclidean information. To build the look-up table, encoding of the elements in the table is done by obtaining the surface normal value of the point, derived from  $\vec{v}$  of the  $x$  and  $y$  axis. This information is readily available as the result from the construction of the surface normal based synthetic LIDAR. Fig. 4.2 shows side-by-side comparison of look-up tables with the additional normal as the corresponding prior. Please note the seemingly striking contrast is expected since the normal is operate on a continuous circle of  $[-\pi, \pi]$ . To speed out the initialization of the raster table, only grids that are within the logarithmic bound of the distance from a point is initialized.

Then, the evaluation function can be constructed by the following equation  $1 - |\vec{w}_{xy} - \vec{v}_{xy}|/\pi$  where  $\vec{w}_{xy}$  is the rotation value along  $xy$  axis of the normal surface from the matching points, and  $\vec{v}_{xy}$  is the value obtained from the additional look-up table. The normalization factor  $\pi$  is used since that is the maximum

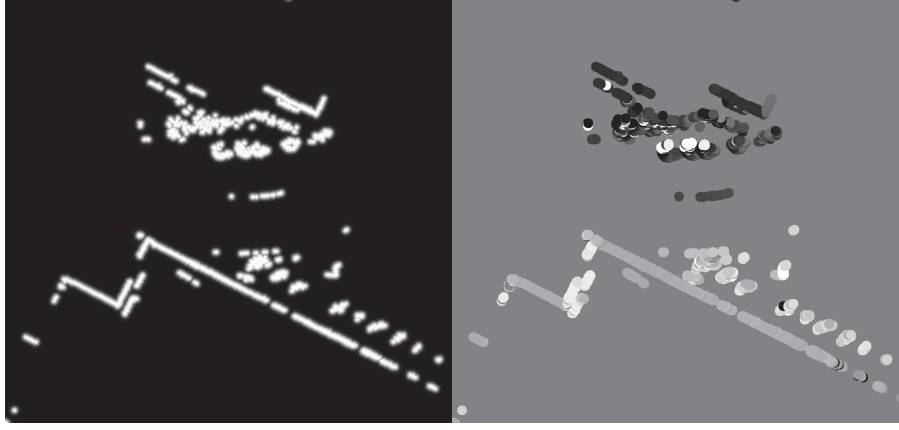


Figure 4.2: Multiple look-up table rasterization. Different features are encoded showing points feature obtained from the euclidean distance (left) metric and surface normal (right) of the point

range expected. This occurs when the rotation of the normal surface between the matching points and the prior points is completely out of phase by  $180^\circ$

In order to quickly identify areas that might contain the correct match, the same approach of a multi-level resolution correlation can be used. The correlation is started with an exhaustive search on the low resolution search window, covering large volumes of 3D search space  $(x, y, \theta)$ . The search is to identify slices that might have the global maximum. By reducing the amount of slices need to be searched, a scan matching can be accelerated.

### Scan Matching Verification

While so far the focus has been on searching for the maximum likelihood of a scan based on a prior, directly inferring the score from the scan matching result can sometime ended up with a wrong match. This is especially true when covering a large urban environment where two places that are far away from each other can share many similar structures geometrically.

The verification is essentially the reversed process of a normal scan matching. Since the transformation of the supposed match is known, a scan matching can be done without going through the search space. While this takes a performance penalty since lookup table need to be reinitialized, a correlative matching is mostly dominated by optimization in a 3D slices, hence only a fraction of the time is needed to perform verification.

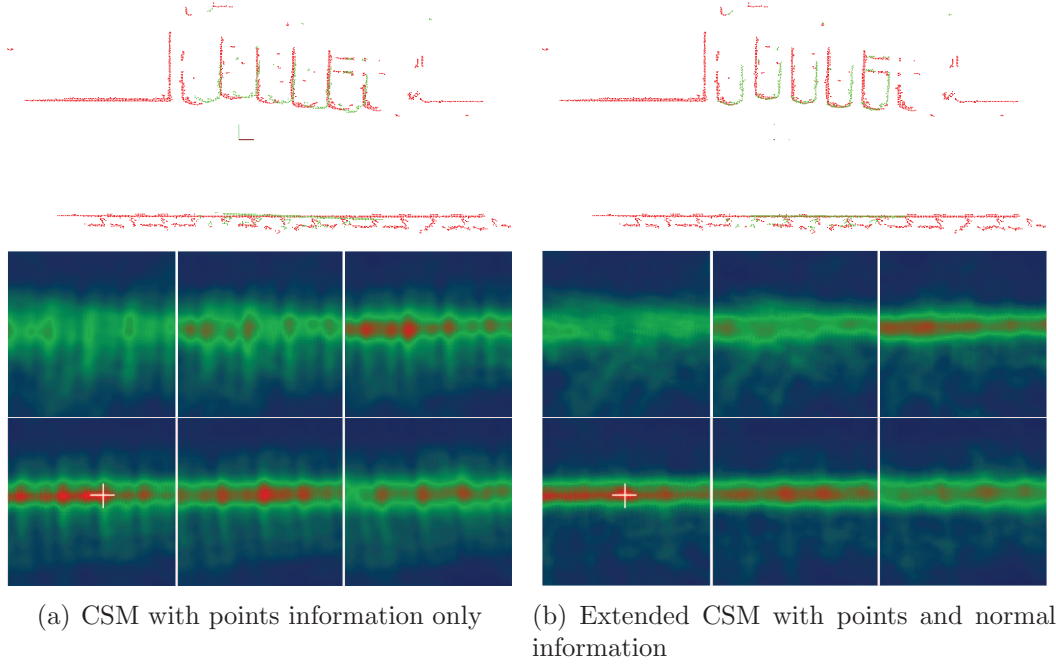


Figure 4.3: Comparative examples with extended correlative scan matcher. The cost function is visualized with each tile represents a slice of the cost volume for a fixed  $\theta$ . In each example, the maximum numerical value is marked with a cross-hair

The comparative result of using multiple raster table is shown in Fig. 4.3, and the 3D correlation cost function of the matching is provided. As evidenced from the matching result, the use of multiple raster tables allowed precise matching and correctly inferred a true match. The value of the cost function also reveals the nature of a match that can contain many local maxima. By using multiple raster table, the true global maxima can be correlated correctly. An example of the use of the verification process is shown in Fig. 4.4. Here, the scan matching produced a high score although it is a wrong match. This is successfully resolved through the verification process that lowers the score substantially.

### GPU Based Scan Matching

A correlative scan matching lends itself favorably to parallel process due to the heavy usage of lookup table. In this work the CUDA framework from NVIDIA is used. Different from a conventional programming, a GPU programming model for CUDA requires declaration of a kernel with a grid of thread blocks, signifying its parallel programming paradigm.

To further increase the efficiency of GPU computing, rasterization is not per-

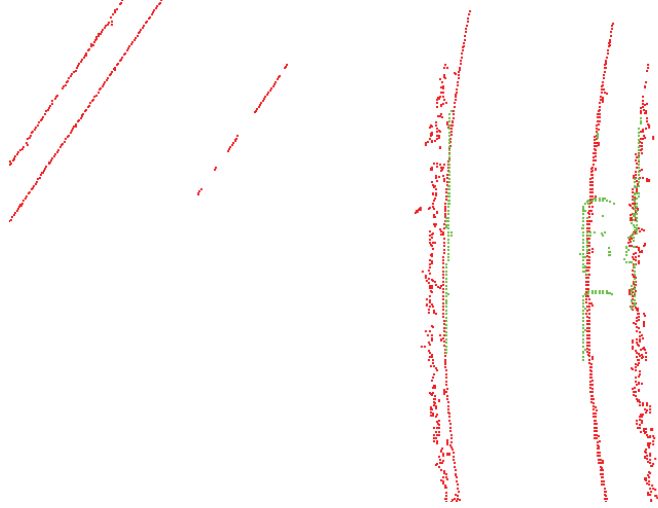


Figure 4.4: A wrong matching resolved through scan matching verification. The scan matching score was 64% while verification through reversed process shows the score of only 17%

formed at first. Instead, a Voronoi diagram is generated to represent the key points of a map surface. This way the extended features describing one particular point can be calculated on the fly without the need of rasterization. Initialization of Voronoi diagram is implemented using a variant of Jump Flooding Algorithm (JFA) [92].

In a simple flooding [93] process, a content located at  $(x, y)$  can be shared to all other pixels by passing the information to its nearest eight neighbors, with position  $(x + i, y + j)$  where  $i, j \in \{-1, 0, 1\}$ . This is depicted in Fig. 4.5. The other way where jump flooding can be done is through step halving. At the beginning of an iteration, a step size is selected at least half the size of the map. Then at each iteration, the step size is halved until the step length of 1. Using the same idea, information passing can be done in parallel to make full use of the massive parallel computation available from a GPU.

For the purpose of Voronoi construction, each point from the synthetic LIDAR is populated as the seeds  $P_j$  on a map surface, where the seed is a pointer to a more descriptive point, for example  $x, y, norm_x, norm_y$ . The map surface is initialized at a given desired resolution  $\varepsilon$  and search space  $(S_x, S_y)$ . For each round with step  $k$ , corresponding pixel on the map surface is filled with the nearest seed. When  $k = 1$ , each pixel in the map would contains the seed that have the smallest Euclidean Distance and a Voronoi is constructed. This process is illustrated in Fig 4.6.

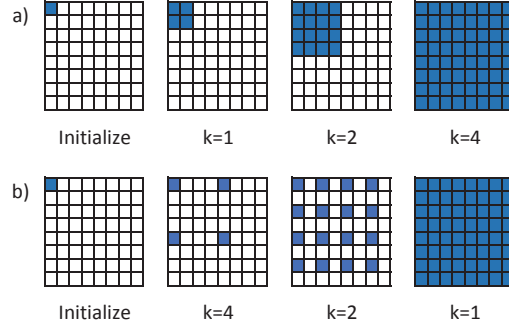


Figure 4.5: A jump flooding example: a) doubling step length b) halving step length

Fig. 4.7 shows computation time needed to construct a Voronoi diagram using JFA across different CPUs and GPUs. In all cases, GPU show significant improvement over CPU implementation. The obtained Voronoi diagram is then bound into GPU texture in a 2D array. A texture memory in CUDA is a type of read-only memory. Although designed for classical OpenGL and DirectX rendering pipeline, texture memory excel in memory access where it exhibit a spatial locality pattern. This can be exploited to provide speed boost since correlative scan matching mostly involves a look-up table with entries that are close to each other.

Once a Voronoi diagram is constructed, scan matching correlation between a pair of synthetic LIDARs can be performed. The matching process is started with each search space given by its rotation range and resolution. At each rotation interval, transformation of points is performed by CPU, then the rotated points is passed into a CUDA kernel in order to evaluate the most probable matching at one particular translation.

Each kernel is initialized with a 2D grid size and 1D thread, where the 2D grid size represents the total steps required to cover the whole translational search space. This setup also allows reduction of best probable match to be performed in a parallel fashion. The size of a kernel's thread is determined by the number of matching points. Since the thread size is limited, only a subset of the points from the observed data is processed. In each loop, CUDA Stream programming model is used to encourage concurrent running of CPU and GPU codes.

In the kernel, each rotated point is evaluated according to its thread index and grid indices. The grid indices specify the translation offset while the thread



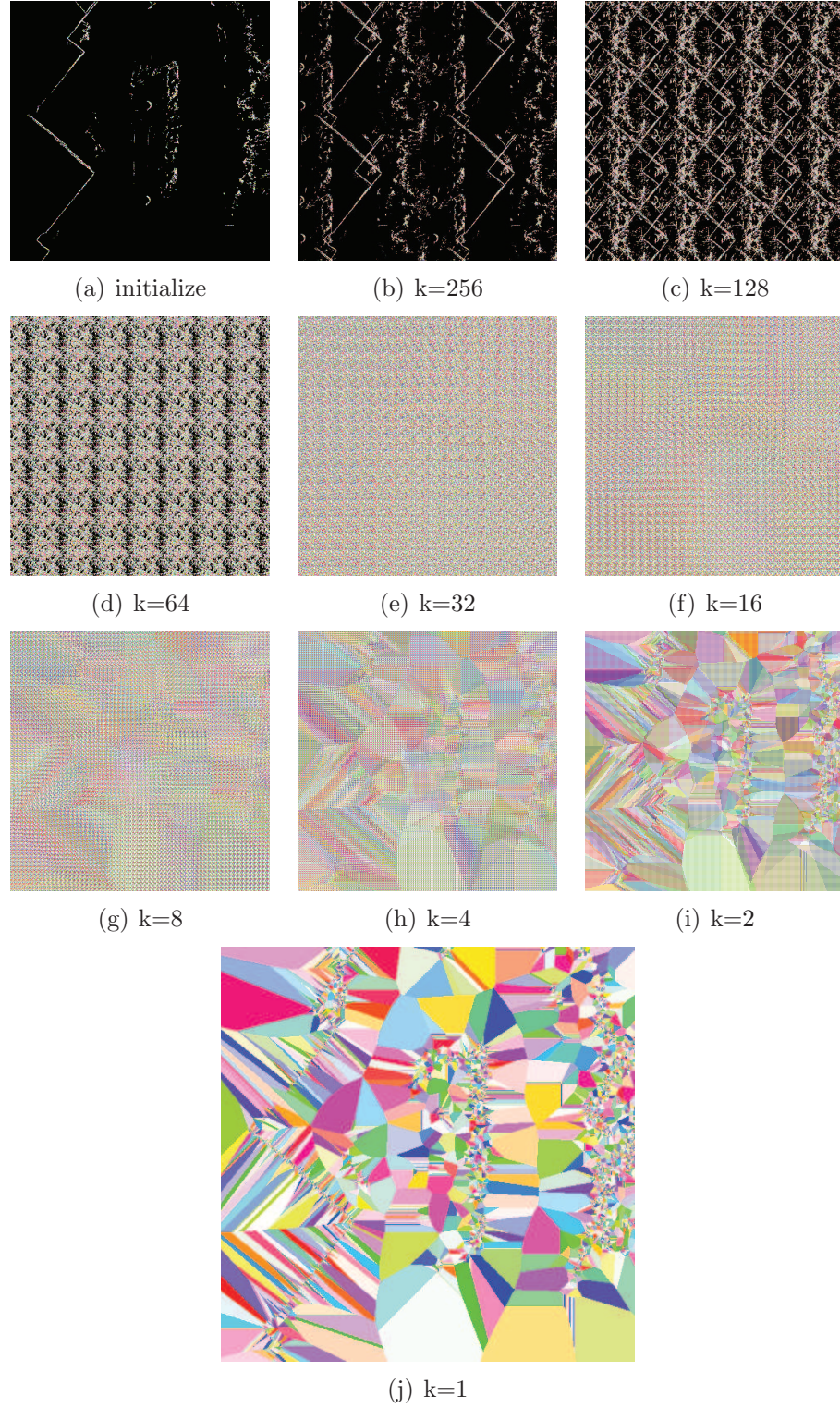


Figure 4.6: A JFA with a 500x500 image with step halving process



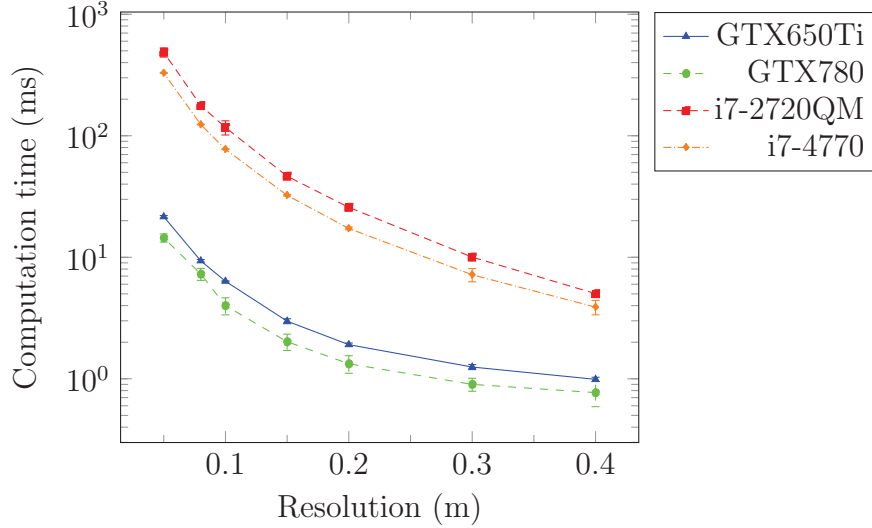


Figure 4.7: Acceleration of Voronoi construction through GPU.

index gives the rotated point index. The correlation occur by reading the Voronoi data from texture memory, and perform necessary cost function as specified by Equation 4.4. Communication between threads happen over the shared memory where summation reduction is performed after every point is correlated. The result is then transferred back to main memory before finally deduce the most probable match.

Fig. 4.8 shows performance of ECSM using different approaches. In this benchmark, 3 set of matching are chose in order to include different matching conditions. In each test, the search space of the matching is set at  $5\text{ m} \times 5\text{ m}$  at  $45^\circ$ , with the resolution of  $0.1\text{ m}$  and  $5^\circ$ . From the result, it is clear that GPU continues to enjoy improved performance across different set of matching.

## 4.2 Mapping with Pose Graph

In the previous section, the construction of synthetic LIDAR and how 2 observations can be matched using extended correlative scan matching is described. In this section, we will show how each of the synthetic LIDAR is used to obtain a coherent map.

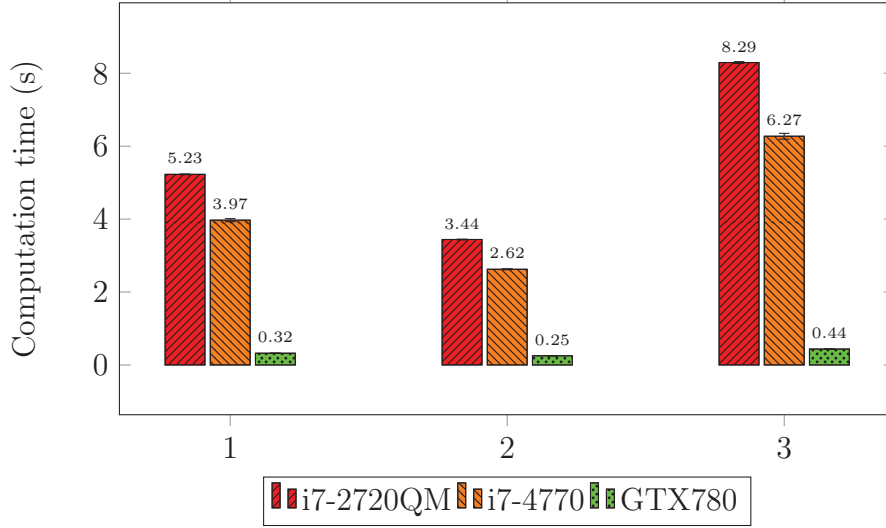


Figure 4.8: Comparison of ECSM matching computation time using different CPUs and GPUs.

#### 4.2.1 Pose Graph Construction

A pose graph is a way to represent mapping problem by using a graph. In the graph, each robot’s pose is represented by a node. An edge connecting 2 nodes represents the constraint according to its spatial relationship. This connection is usually obtained directly from a robot’s odometry.

A simple construction of a pose graph based on synthetic LIDAR can be done by treating the synthetic LIDAR as though it is just a normal planar LIDAR. In this case, the addition of nodes occurred only at a distance larger than the width of the rolling window. By defining a system that have a set of poses  $x_{1:T}$  with odometry constraints  $u_{1:T}$ , a pose graph can be constructed.

More accurate pose graph should include all the original scan lines that are originated from one single instance of a synthetic LIDAR. In other words, each measurement obtained from the LIDAR is given a node to construct a pose graph. In addition, a major node is added at the same distance as the width of the rolling window. This way, the pose graph’s construction matches the physical observation model of a synthetic LIDAR.

Although this adds more constraints into the graph, the work load on the front end remain the same by restricting the loop closure search on the major nodes only. Then, the least square problem can be solved quickly with an efficient solver. With the odometry based pose graph built, the graph is now ready for optimization

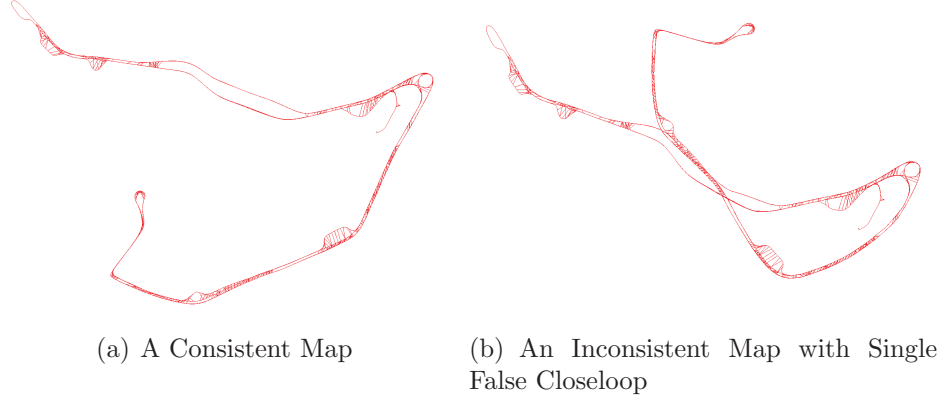


Figure 4.9: A Comparison of pose graph optimization with a false loop closure

by adding loop closures  $c_{ij}$  between 2 poses  $x_i$  and  $x_j$ . Because a pose graph is formulated as a least square problem, a single false positive can adversely affect the final optimized result, yielding an inconsistent map. Fig. 4.9 shows an optimized graph when a wrong loop closure is added to the mapping system.

To ensure that this will not occur during mapping, a supervised loop closure can be done. Whereby a hard constraints are injected at the beginning of a graph optimization cycle by visually identifying loop closure manually. This ensures a stable graph and any addition of constraints should results in a smoothed graph updates.

The mostly consistent graph at the macro scale can now concentrate on adding loop closure locally. This can be done by performing a simple best match search around neighborhood nodes. Optionally, the matching algorithm can be made to restrict the search window to encourage loop closure at different orientation, by having a smaller search space on the scan matching algorithm. For example, 2 different search windows at different rotation:  $[-\frac{\pi}{2}, \frac{\pi}{2}]$ ,  $[-\frac{\pi}{2}, \frac{3\pi}{2}]$  are used at each pass. Effectively, this allows potential loop closures to be included in both directions, which covers most type of the close loop encounters in an urban environment.

### 4.2.2 Automatic Loop Closures

The pose graph construction described above involve a rather laborious step where hard constraints are added manually. To allow a fully automated mapping process, particle filter is proposed to recognize the loop closures.

The particle filter is used extensively for robot's localization. The well-known

Monte Carlo Localization [94, 95], which uses particle filter has been applied successfully that provide accurate positioning [74, 96] of a robot moving within known environment. The use of the particle filter is the key that enables localization to be done efficiently. Liu *et al.* in [97] used particle filter to perform loop closure detection between 2 visual pair images. Here, the particle filter is adapted to take the advantage of the view point invariance of a synthetic LIDAR.

In a particle filter based loop closure detection, each particle corresponds to a node in the map and the particle set  $X_t$  describes the posterior probability distribution of the current active node. As more nodes are being added to the map, loop closure is detected when there is a high probability density on a different node. The details of the loop detection process is described below.

### Motion Model

A motion model is applied to propagate the particles when a new node is being added, which has the following form:

$$x_t = p(x_t | u_t, x_{t-1}) \quad (4.5)$$

The motion model is part of the important function in the iterative nature of particle sampling framework. Due to 1D nature of the graph topology, a simple forward motion is used with large probability of particles moving from a node to another at each step. In a LIDAR measurement, the forward motion can be ambiguous since the measurements are rotational invariant. A forward motion may not necessary imply an increment to the node. To overcome this ambiguity, heading information from the scan matching result is used when propagation of the particles is performed. The motion models is given as follows:

$$p(x_t = \varrho \cdot n | x_{t-1} = n, u_t) = p_0 \quad (4.6)$$

$$p(x_t = \varrho \cdot (n + 1) | x_{t-1} = n, u_t) = p_1 \quad (4.7)$$

$$p(x_t = \varrho \cdot (n + 2) | x_{t-1} = n, u_t) = p_2 \quad (4.8)$$

$$p(x_t = \varrho \cdot (n + 3) | x_{t-1} = n, u_t) = p_3 \quad (4.9)$$

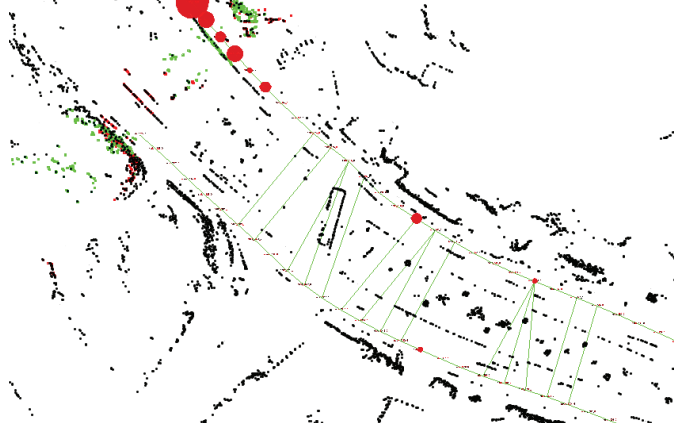


Figure 4.10: Visualization showing the mapping process.

Here  $p_1$  is normally assigned with high probability while a small value is given to the rest of the  $p_n$ , to account for the noise in the system.  $\varrho$  is the value that is initialized from the rotation value,  $\phi$  from the result of scan matching, where

$$\varrho = 1 - \frac{2|\phi|}{\pi}, \quad -\pi \leq \phi < \pi \quad (4.10)$$

### 4.2.3 Overall Algorithm

The overall process is summarized in Alg. 1. The algorithm is activated with each addition of node. The input to the algorithm are the previous particles  $X_{t-1}$ , the updated measurement of each particle  $z_t$  and current total number of nodes  $N$ . Parameter  $M$  is the total number of particles used. From line 12-16, random samples is added to maintain diversity of the particles.

To add a random sample, different random number generators are used. To generate  $j$ , the particle index is drawn according to the weighted Euclidean distance from the current node based on the latest updated position available from the optimized graph. This is to allow more chance of discovering close loop even when low number of particles is used. On the other hand  $k$  is generated according to a discrete uniform distribution.

Fig. 4.10 shows a snapshot of the mapping process. In the mapping, 50 particles are used for loop closure detection, and a loop closure that has a difference in residual error below 10 is accepted as a loop constraint. The green lines are the factor graph and red circles represent the position of the particles. The red circle has radius drawn according to the total number of particles in a node, a larger

**Algorithm 1:** Monte Carlo Close Loop Detection

---

**Input:**  $X_{t-1}, z_t, N$   
**Output:**  $X_t, \hat{x}_t$

```

1  $X_t = \bar{X}_t = \emptyset;$ 
2 for  $i = 1 \rightarrow M$  do
3    $x_t^{[i]} = p(x_t | u_t, x_{t-1}^{[i]});$ 
4    $w_t^{[i]} = p(z_t | x_t^{[i]});$ 
5    $\bar{X}_t = \bar{X}_t + \langle x_t^{[i]}, w_t^{[i]} \rangle;$ 
6 end
7 for  $i = 1 \rightarrow M$  do
8   draw  $j \in \{1, \dots, N\}$  with probability  $\propto w_t^{[j]}$ ;
9   add  $x_t^{[j]}$  to  $X_t$ ;
10 end
11 for  $i = 1 \rightarrow \alpha M$  // addition of random samples
12 do
13    $j = R[1, N];$ 
14    $k = U[1, M];$ 
15    $x_t^{[k]} = j;$ 
16 end
17  $\hat{x}_t = \text{mode}(X_t);$ 
18 return  $X_t$ ;

```

---

radius represents higher density of particles. The visualization shows that the scan matcher is able to perform accurate matching of 20  $m$  offset at the opposite direction. The other challenging area present in this environment is along the overhead bridge connecting between NUS main campus and NUS UTown (Fig. 4.11b). The long stretch of highly uniform environment ( $\sim 250 m$ ) poses a stiff challenge to the mapping framework. In this area, the use of Monte Carlo loop detection and loop constraint rejection successfully created an accurate map of the environment. In this map, the vehicle travels more than 6 km while covering an area of  $720 \times 900 m$ .

## 4.3 Map Updates

While the discussion so far only involves a single vehicle perform mapping in one particular region within a same period of time. By introducing anchor points that share a single common global frame, multiple maps with independent trajectories can be combined and updated. This is true even when the map is performed at

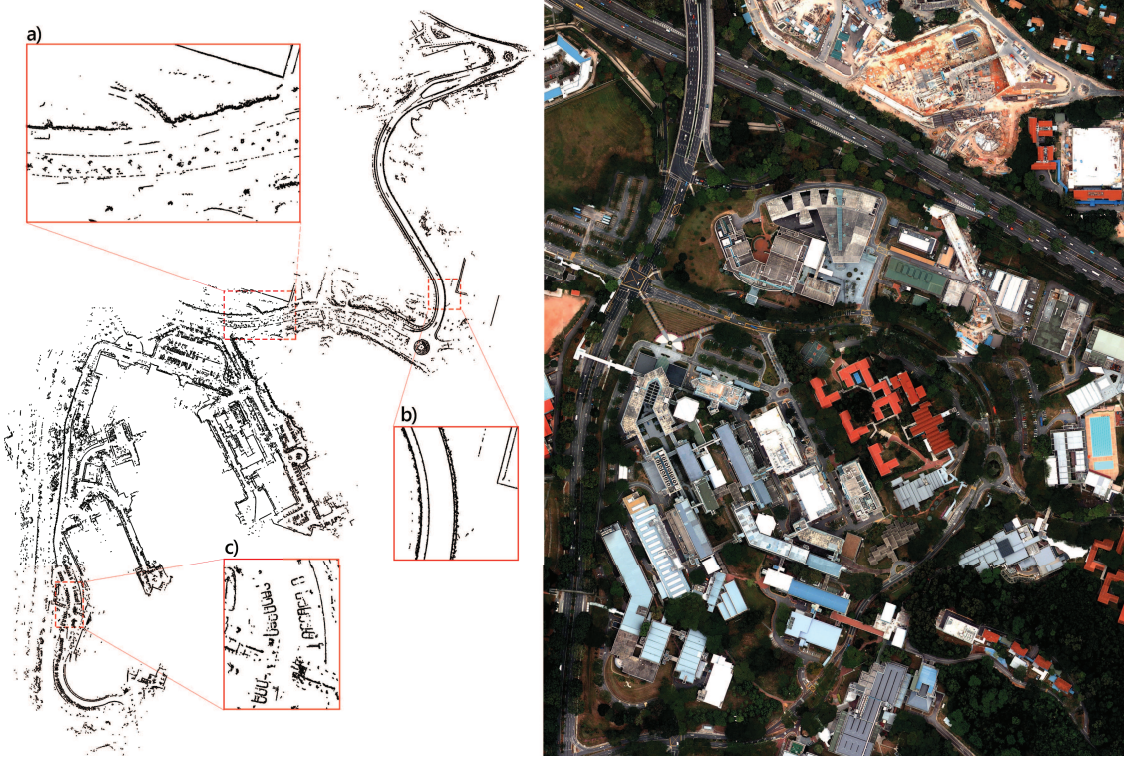


Figure 4.11: Mapping of the NUS campus area

different time with different vehicles. This concept is introduced by Kim *et al.* [98].

The key that allows map merging is the discovery of encounters, in which additional constraints connecting 2 different graphs are added. In other words, as long as there are some overlapping region, different map performed at different time by different vehicles can be updated into one consistent map. More formally, robot trajectories  $r_i, i \in 1 \dots R$  for  $R$  robots, an anchor  $A_r$  is added to each trajectory, sharing a same global coordinate.  $A_r$  specifies the rigid transformation of a trajectory relative to the global coordinate. This formulation allows optimization to perform jointly with all the pose graphs, connected by encounters. Fig. 4.12 shows the result of a map merging with dataset collected at time separated by 3 weeks.

The single consistent pose graph is further processed to give a good summary on the environment, thereby providing a simpler, more coherent way to store the map and keep it updated. One of the most common map representation is using a 2D occupancy grid. Due to the see through nature of a synthetic LIDAR, a lot a features can be lost due to the ray casting operation on 2D. Instead, the probability at each cell being occupied is estimated by a Bayes filter normalized by its neighboring cells. A free space is initialized as a polygon based on the geometry



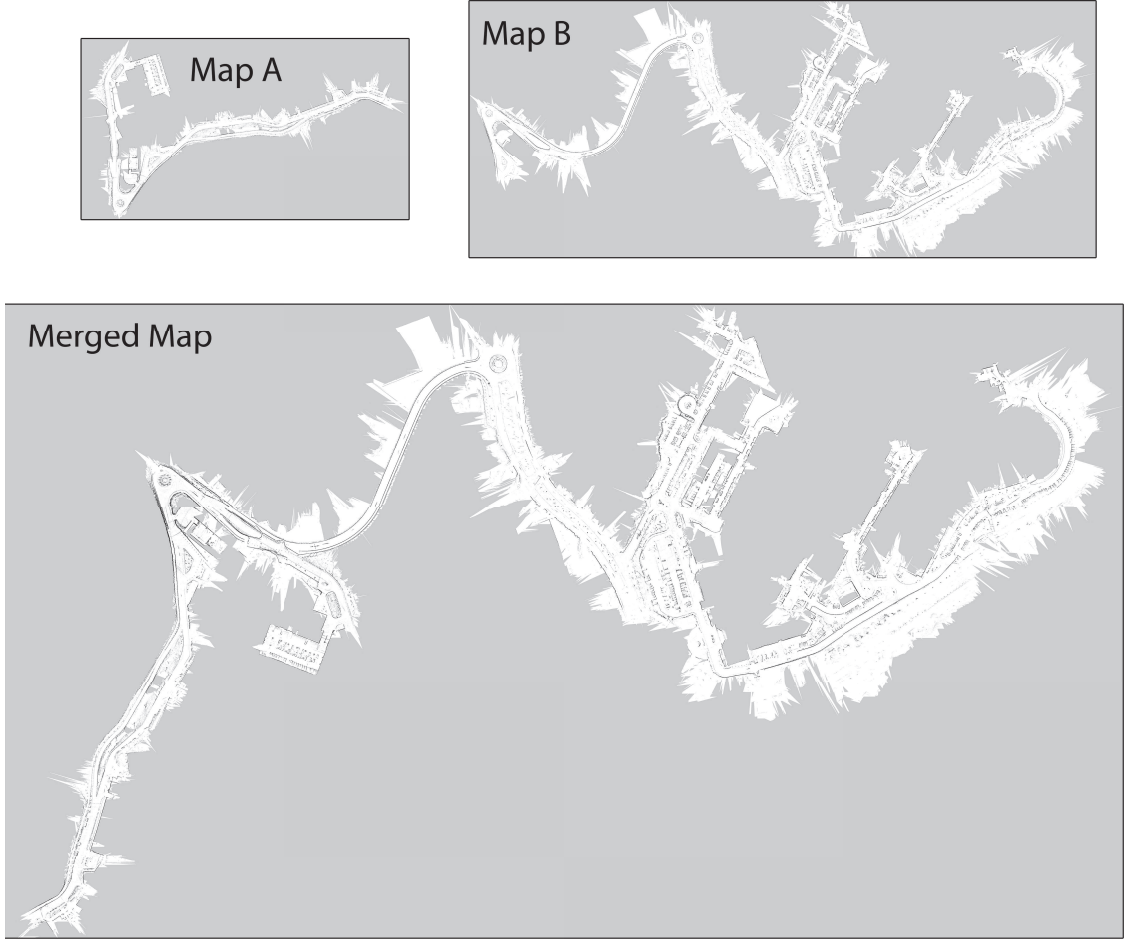


Figure 4.12: A merged map

of the extracted features from a single LIDAR.

On the other hand, a compact representation of a 3D space can be constructed with an Octree [99]. The Octree is specified by a minimum voxel size, where a single volume is the result of a combined volume that is made up of 8 voxels. As a result, a hierarchical structure is maintained to represent this 3D space, where one layer is subdivided into 8 layers, and additional layers are added until the smallest voxel size is achieved.

The implementation of update on each voxel is performed in a probabilistic manner. To build an Octree based map, all the raw observations from an optimized graph are integrated. Using the poses available from the pose graph, each of the raw observation is integrated using ray casting. Finally, a threshold value is used to determine the occupancy of a voxel. One immediate improvement that can be seen from this process of map update is the use of a more complete model of sensor



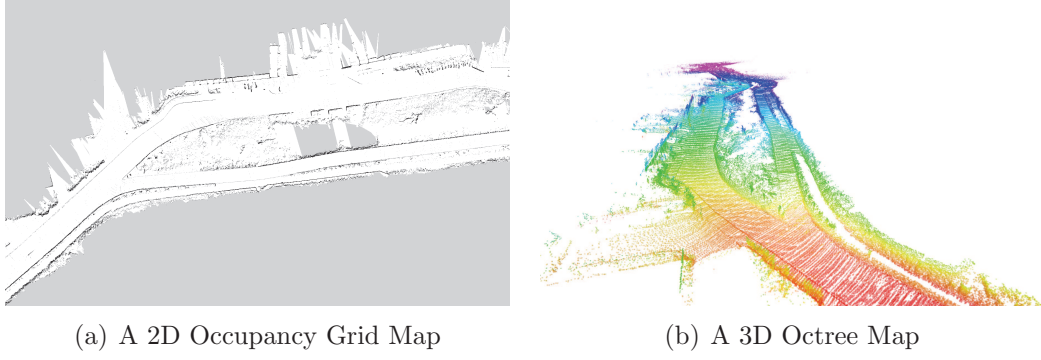
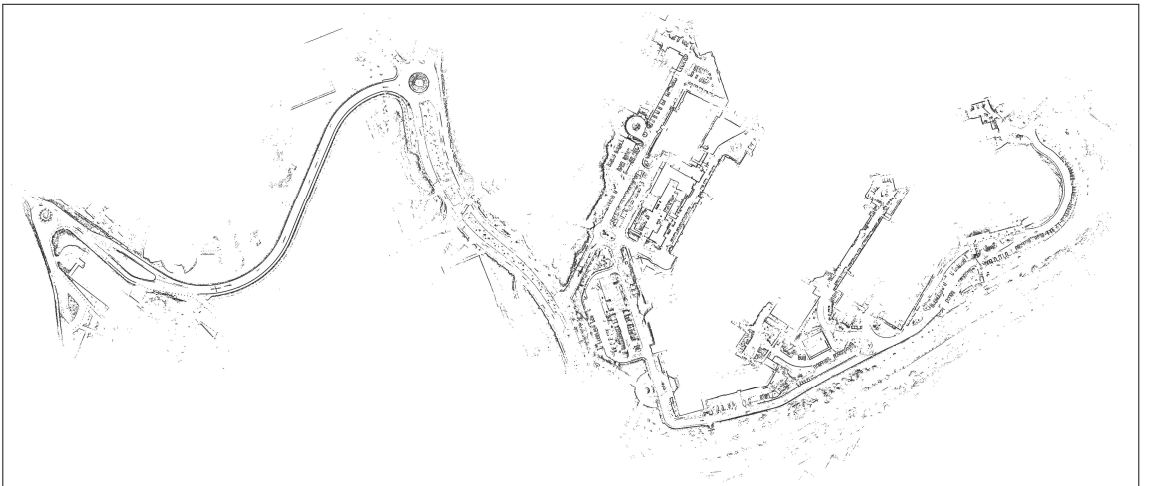


Figure 4.13: Map updates in 2D or 3D, showing the same section of the route

fusion method, where by each voxel is updated by performing ray tracing. This way, changes in the environment are included where each cell may become occupied or empty, depending on the probability of the cell. For example, if a parked car become disappeared in a merged map, the existence of the car will be removed. Fig. 4.13 shows the different map generated separately through 2D and 3D update process.

## 4.4 Mapping Results

This section includes mapping that is done on different places in Singapore. The resolution is fixed to 0.1 m and the corresponding statistics is reported. For clarity, figures of the dataset only visualize the occupied cell of the map. The file size reported in the result reflects the amount of memory required to store the 2D occupancy grid map in lossless bitmap image format using Portable Network Graphics.



**NUS Engineering:**

Map size: 1086.4 m x 454.8 m

Distance: 6.5 km

File Size: 999.8 kb

This map captures part of the NUS campus hilly roads that host most of the engineering buildings. It contains a sizable amount of parking lots and wide public roads. The map includes the a overhead bridge that connects between NUS main campus and NUS UTown.



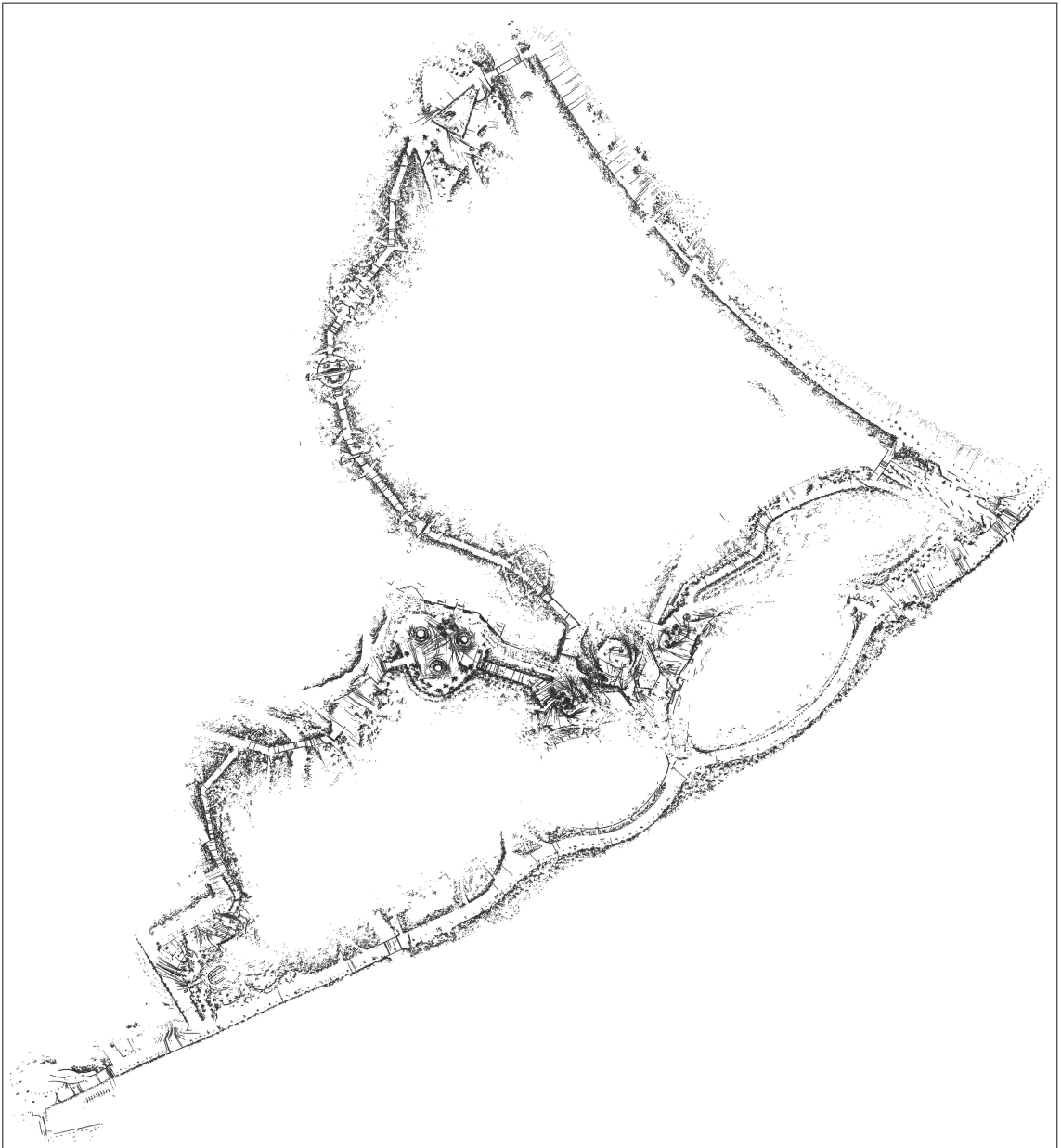
**NUS UTown:**

Map size: 490.2 m x 599.9 m

Distance: 5.2 km

File Size: 999.8 kb

This map contains a good mixture of indoor and outdoor environment. The vehicle travels through inner part of the campus comprising offices and residential area. This demonstrates the capability of the synthetic LIDAR working under different condition and stresses the mapping framework.

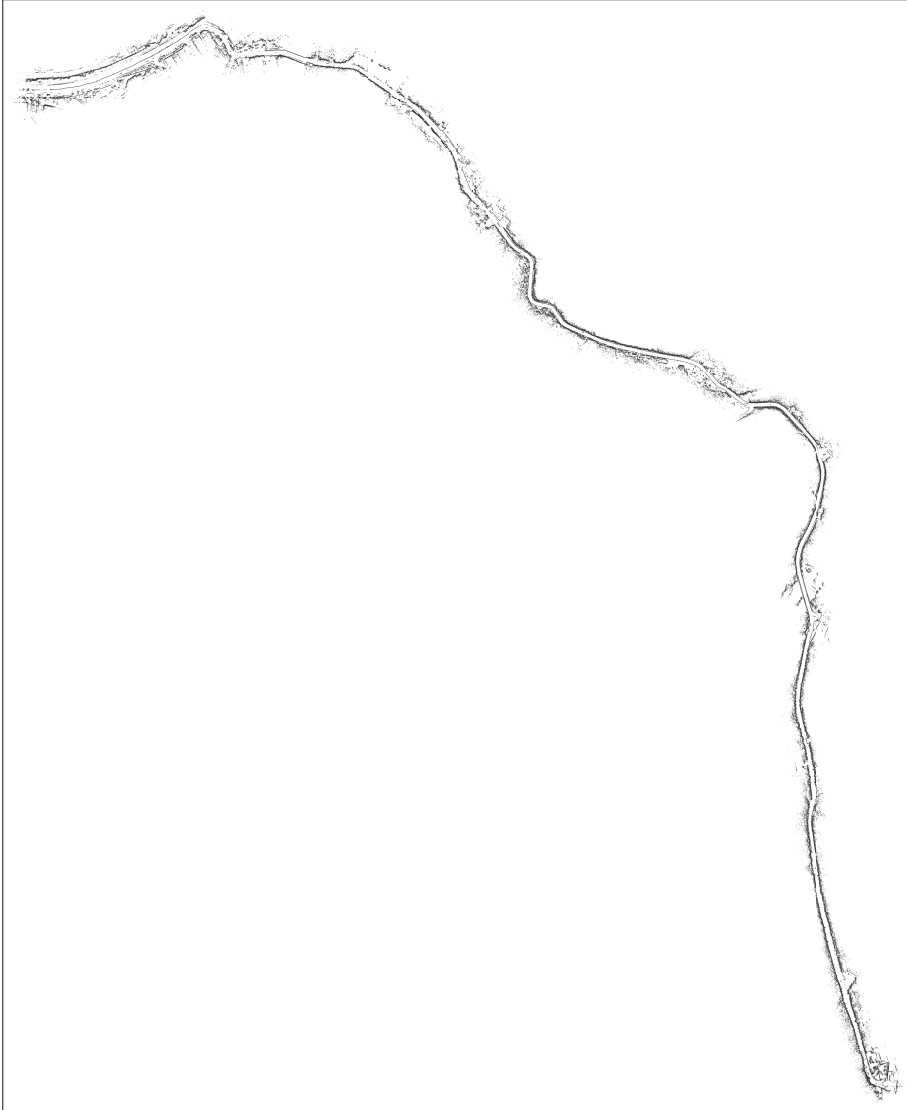
**Gardens by the Bay:**

Map size: 887.4 m x 966.6 m

Distance: 5.3 km

File Size: 1.95 MB

A rare chance as we brought Rudolph to the Marina Bay Sand. We manage to perform mapping at the Garden by the Bay. The garden contains man-made sculptures, natural foliage and buildings. The mapping framework continues to adapt well into different environment.



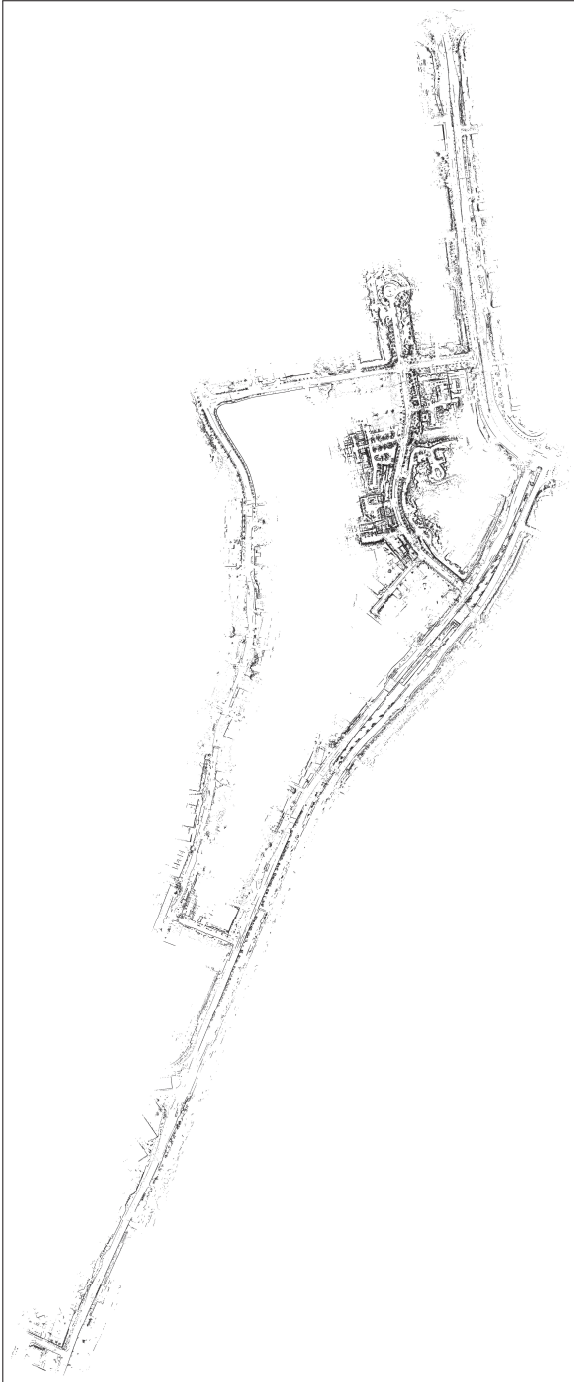
**Riffle Range Road:**

Map size: 1806.1 m x 2207.9 m

Distance: 6.9 km

File Size: 2.7 MB

A long and windy road of Riffle Range Road, where the road is covered by dense tree canopy.

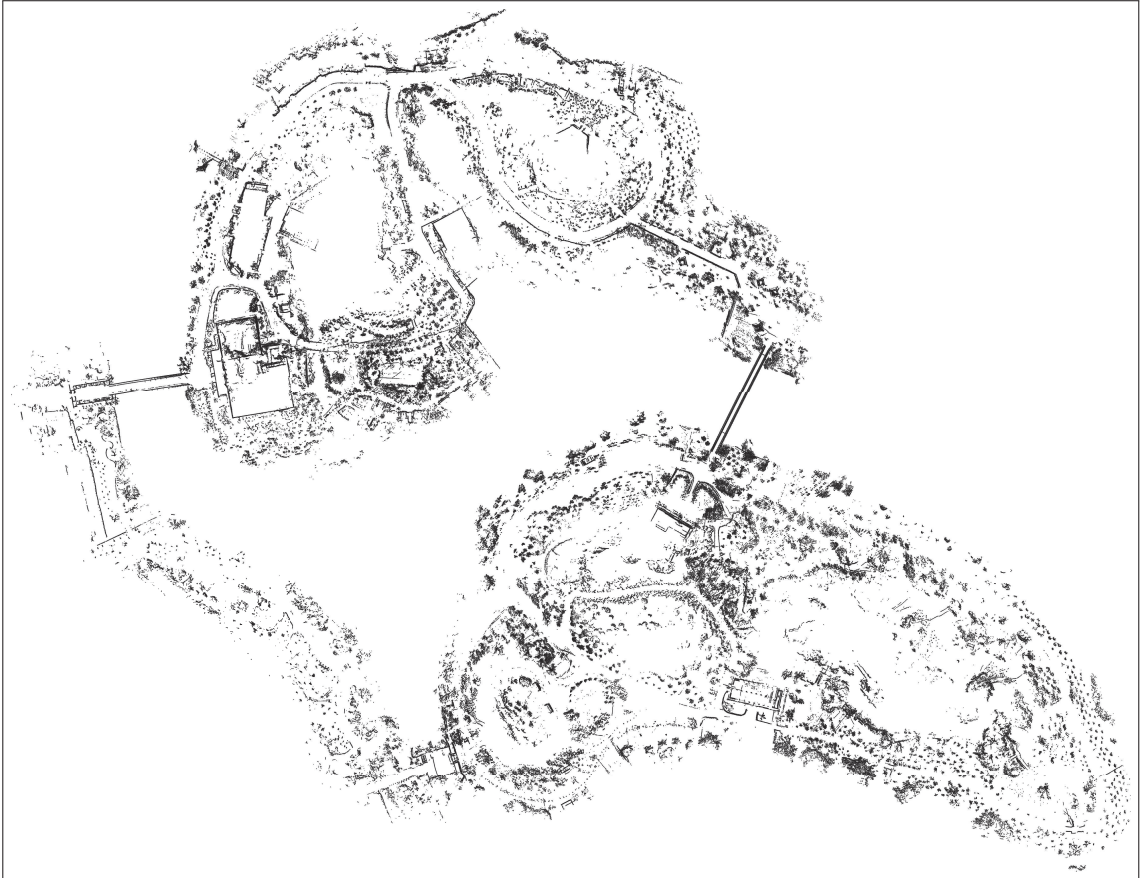
**Commonwealth Ave:**

Map size: 477.8 m x 1166.5 m

Distance: 13.3 km

File Size: 3.26 MB

One could find almost every traffic elements in this environment. This map includes 2 MRT stations with its surrounding road networks, truly represent an urban environment. The road network provides connection to many residential areas with traffic lights and pedestrian crossing.

**Chinese and Japanese Garden:**

Map size: 683.4 m x 887.4 m

Distance: 6.9 km

File Size: 2.7 MB

Participation in Jurong Lake District Project allows an opportunity to perform mapping in the gardens. These 2 gardens is connected by bridges and share very similar environment with the Gardens by the Bay.

## 4.5 Summary

The synthetic LIDAR is used to perform mapping of the environment. One of the contributions is the proposal of extended correlative scan matching. To include more features available from a synthetic LIDAR, multiple lookup-table rasterization is used. An example is included where we show how the inclusion of surface normals can improve the scan matching result by locating an accurate global maxima.

The thesis also includes the use of GPU based scan matching algorithm and shows significant speed up when performing a match. In particular, the lookup-table rasterization is replaced by Voronoi construction through Jump Flooding Process. The process is further accelerated by the use of the CUDA framework's texture and shared memory.

The map is constructed using pose graph. Although it is shown to be highly effective, a single false loop closure renders an inconsistent map, where the reconstruction does not correspond correctly to the physical dimension of an environment. To this end, we show how it can be efficiently done automatically through particle filter based loop closure detection. The algorithm, running in parallel with the graph construction, allows mapping to be done concurrently while the vehicle is exploring the environment.

Finally, the map is updated using a 2D occupancy grid or a 3D Octree based map. This way, the map can be updated constantly by incorporating new measurements. To evaluate the mapping framework, the map generated from many different places are presented.



# Chapter 5

## Map application

Previous chapters show how a precise map can be built using synthetic LIDAR. Having an consistent map is one of the key components in autonomous navigation, as the map serves as the backbone of the underlying applications. In this chapter, we show how the map can be used to solve problems related to vehicle navigation in an urban environment, specifically localization, road mapping, and graph learning of an urban road network.

### 5.1 Localization

In this section, we examine how a map is used to perform localization. We first look at how virtual sensor proposed in Section 3.1.2 is used to perform localization. In the localization, we assumed that the map is given. As part of our earlier work, the curb lines are extracted manually with reference to a satellite map represented by occupancy grid. In the map, an occupied cell represents curb while a free space is the road surface.

To perform robust localization, Adaptive Monte-Carlo Localization (AMCL) is used [100]. In AMCL, particle filter is employed to estimate the position of a vehicle. More formally, the position of a vehicle is represented by a belief  $bel(x_t)$  which is represented by a set of  $M$  particles  $x_t^{(m)}$ :

$$bel(x_t) \sim \{x_t^{[m]}, \omega_t^{[m]}\}_{m=1}^M \quad (5.1)$$

where  $w_t^{[m]}$  is the importance weight. The estimation of the position for each



particle is performed recursively using the following steps:

1. Prediction: Control input at time  $t$ ,  $u^t$  and particles set at previous time step,  $\{x_{t-1}^{[m]}, \omega_{t-1}^{[m]}\}_{m=1}^M$  is used to generate a new set of particles  $\{x_t^{[m]}, \omega_t^{[m]}\}_{m=1}^M$ , given a motion model  $p(x_t|u_t, x_{t-1})$ .
2. Correction: The importance weight, associated with each particle  $\{x_t^{[m]}, \omega_t^{[m]}\}_{m=1}^M$  is then updated with a new measurement  $z^t$ , with a measurement model  $p(z_t|x_t, m)$ .
3. Resampling: The particle set will be resampled when the importance weight does not contain a good variance. After resampling, position of the vehicle  $bel(x_t)$  can be obtained by approximating the distribution of the particles.

The constructions of curb and intersection sensors allow tight integration into AMCL. This is achieved by using 2 different sensor models: beam and likelihood model.

**Beam Model** A beam model closely resemble the physical working principal of a LIDAR. Assuming that each line of observation is independent, where a scan  $z$  contains  $K$  measurements,  $z = z_1, z_2, \dots, z_k$ , then the probability of a measurement  $z$  given that a robot is at  $x$  with a known map  $m$  is:

$$P(z|x, m) = \prod_{k=1}^K P(z_k|x, m) \quad (5.2)$$

To specify  $P(z|x, m)$ , the following scenario are considered:

1. A normal measurement reflected by an obstacle

$$P_{hit}(z|x, m) = \eta \frac{1}{\sqrt{2\pi\sigma_{hit}^2}} e^{-\frac{1}{2} \frac{(z - z_{exp})^2}{\sigma_{hit}^2}}$$

2. A noisy measurement due to moving objects

$$P_{mov}(z|x, m) = \eta \lambda e^{-\lambda z}$$

3. A random measurement,

$$P_{rand}(z|x, m) = \eta \frac{1}{z_{max}}$$

4. An out of range measurement,

$$P_{max}(z|x, m) = \eta \frac{1}{z_{small}}$$

All the different distributions above are mixed together by a weighted average, where the sum of  $z_{hit}$ ,  $z_{mov}$ ,  $z_{rand}$ ,  $z_{max}$  is 1.0. These are all intrinsic parameters.  $\sigma_{hit}$  is another intrinsic parameter of the measurement model which encapsulate the inherent noise in a good measurement value. One way to obtain these parameters is to learn from actual data through maximum likelihood estimators [47].

**Likelihood Model** Although a beam model closely model physical causes of a measurement, it is not without its drawback. A beam model requires ray casting which is computationally expensive. On top of that, in order for a measurement to hit a reasonably likely obstacle, a large number of samples are required. This is because a nearby points can have very different likelihood and hence lack of smoothness. A likelihood model overcomes this by not considering the beam but only the end point of the LIDAR. This is different from a beam model where each observed point needs to compute the line of sight to obtain its probability value. In a likelihood model, each end point of the LIDAR is model as the probability of the point being accurately measured. Typically the model is given as

$$p(z|x, m) = \frac{1}{\sqrt{2\pi\sigma_{hit}^2}} e^{-\frac{d^2}{2\sigma_{hit}^2}}$$

where  $d$  the measured distance to the nearest obstacle. The implication of using likelihood model is that the evaluation on a single measurement do not need to perform ray tracing, and the likelihood filed can be calculated by a precomputed lookup table. Hence, this greatly speed up the probability calculation.

For curb sensor, likelihood model is used for its computation efficiency and for its less sensitive to noise. On the other hand, a beam model is used for intersection

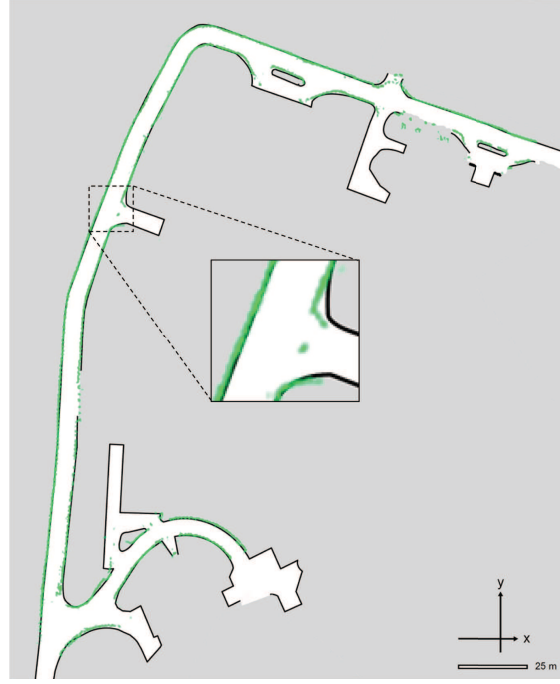


Figure 5.1: A curb map. The black curves shows the curb line drawn at priori with detected curb drawn with green curves

sensor. To exploit a beam model's tight relation to the geometry and physics of a LIDAR, an intersection point is represented by a maximum range. This correspond well to an intersection's free space towards the road side.

Experiments are performed to evaluate the capability of the virtual sensors. Fig. 5.1 shows the manually drawn map of size  $200m \times 240m$  overlaid with points output from the curb sensor. Using an instrumented vehicle, a rough initial position is given at S, and driven autonomously for about 430 meters to G. An overview of different localization solutions are shown in Fig. 5.2. Fig. 5.4 shows the standard deviation of the particles while the vehicle is navigating through the route. In general, longitudinal variance is higher than the latitudinal. This is expected as the vehicle's movement mostly parallel with the curb lines. Section S-A shows a curvy section of the road, where the particles' standard deviation remain well within 1.5 m. The same can be said at section C-D-E. Points A, C, E and G shows the effect of recognizing intersections whereby particles homing to the intersection region. From Table 5.1, 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 (similar to B). The errors are obtained by having the vehicle stops at the same designated virtual stop point repeatedly and compare the difference

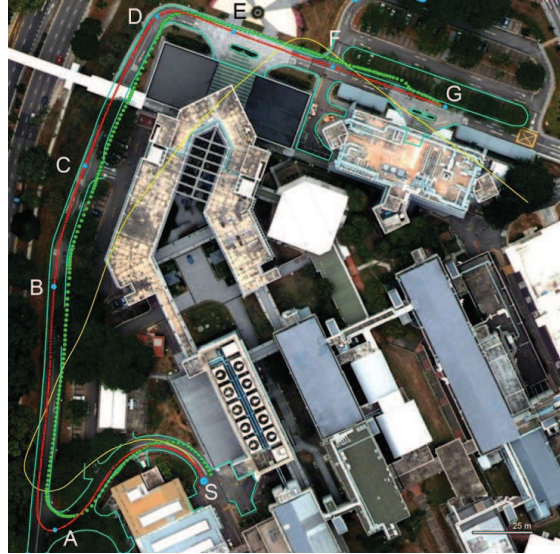


Figure 5.2: An overview of position estimates based on different types of localization solutions. The light-blue lines denote road boundary. The red line marks the localization result of the curb-intersection feature based MCL, and raw odometry trace is shown by yellow line. GPS output from GPS/INS module (Ublox EVK-6R) is included showing as green dotted line.

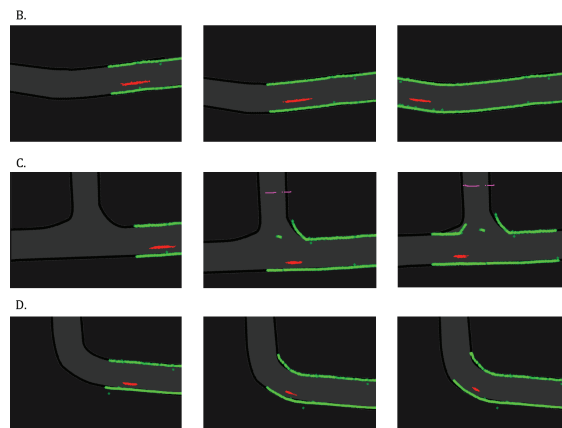


Figure 5.3: Typical particle behaviors at points marked in Fig. 5.2

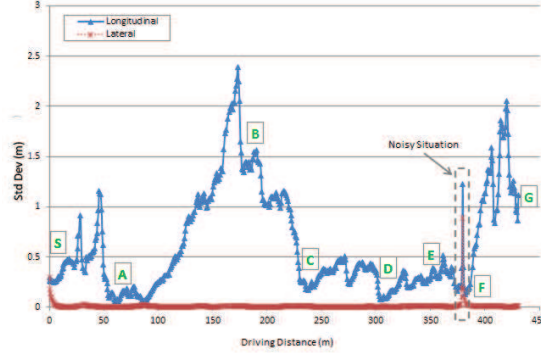


Figure 5.4: Position estimation variance

by markers placed on the ground.

To get a sense of how the virtual sensors work, Fig. 5.3 shows the particles' behaviors at different places of the map. The elongated particles running parallel with the curb is the typical behavior expected from this sensor model. This is attributed to the uniform longitudinal features of the road section. Even with a small curve, it shows that there is enough feature differentiation. Hence, a more precise location estimates is achieved where it is reflected by more confined particles. This is more evident in *D*, where a similar argument can be used to explain the phenomenon of the increased in confidence of the estimates, reflected by higher density of particles concentrated in a smaller area. *C* shows the effect of the activation of the intersection sensor. In this scenario, particles that falls outside the intersection that is not in agreement with the existence of non-curb observation have a smaller importance weight. With lower probability to survive in the resampling step, particles that are located within the intersection have higher chance to survive.

More experiments are performed using synthetic LIDAR, which is constructed as discussed in Section 3.2. By extracting interest points from a reconstructed detailed environment model, the synthetic LIDAR shows excellent adaptability with different types of urban environment scenario. As with before, the localization

Table 5.1: 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 5.5: Mapping of the NUS engineering area

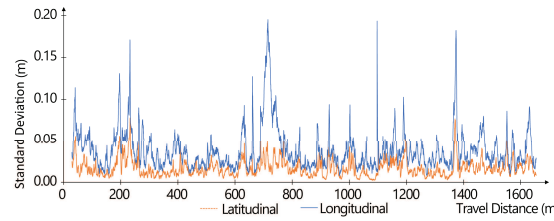


Figure 5.6: Synthetic LIDAR localization results

method uses MCL scheme to estimate the vehicle pose. Since a synthetic LIDAR is more expressive in its observations, i.e. it can have more than one point within a small angle, likelihood measurement model is used. The likelihood model, where only a beam's end point is required to perform calculation is translated directly to the measurements from a synthetic LIDAR.

Fig 5.5 shows the map we used to perform the localization experiment overlaid on top of a satellite image. When compared with the previous localization experiment, it covers large part of NUS engineering campus with an area of  $550\text{ m} \times 487\text{ m}$ . The map generation technique is described in detailed in Section 4.3. Fig 5.6 shows the particles' uncertainty while the vehicle is driven around the campus covering all the traversable region in the map 5.7. Traveling for about 1.6 km, the experiment results show much lower standard deviation compared with previous





Figure 5.7: Vehicle path estimated using synthetic LIDAR localization

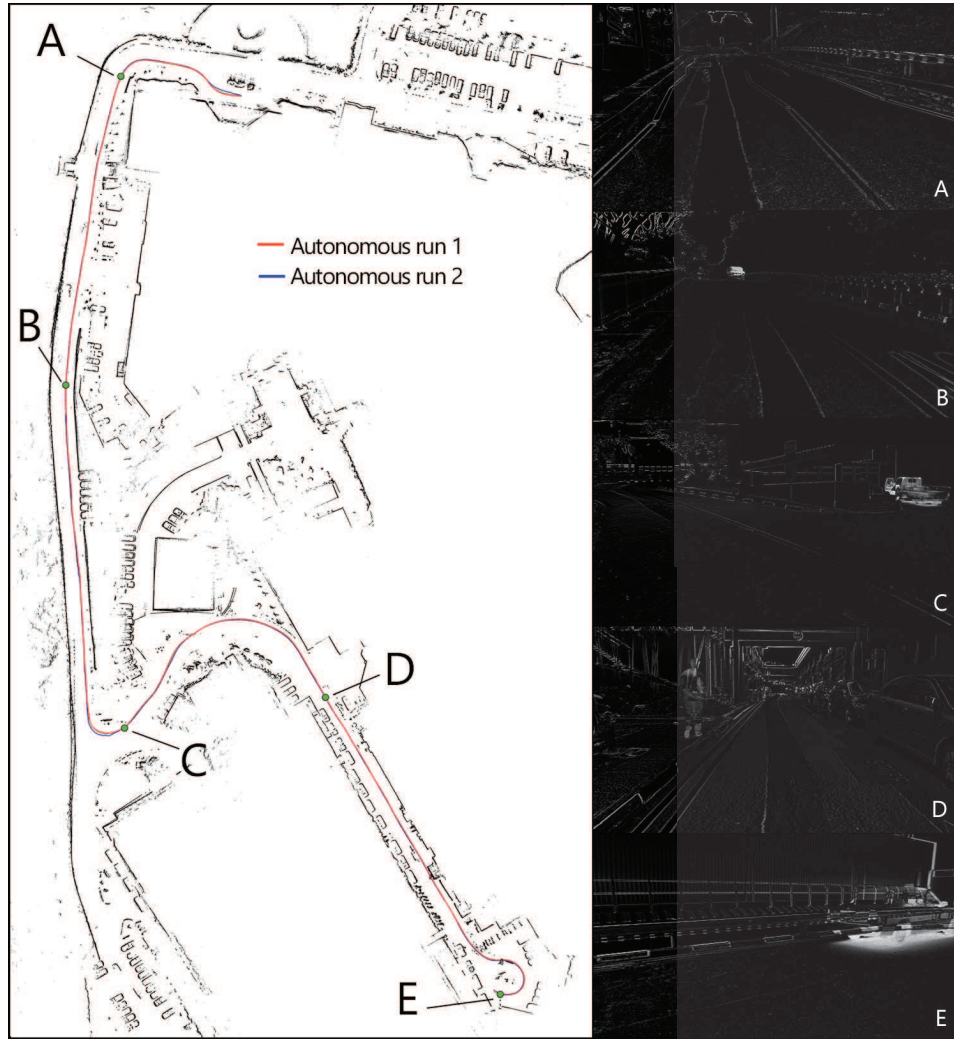


Figure 5.8: Autonomous navigation with synthetic LIDAR. Images on the right from top to bottom correspond to visual validation of localization repeatability from checkpoint A to E

method with maximum standard deviation value of 0.2 m. As with curb and intersection based localization, the position estimation exhibit higher confidence along the lateral direction. This make sense as features are richer on the lateral direction. One should note that the longitudinal standard deviation is almost as good as the latitudinal values. This observation shows that the longitudinal features also quite as rich providing strong indication of the capability of the synthetic LIDAR to carry equally descriptive features along the longitudinal direction.

Using the same localization method, autonomous navigation is performed at one section of the map. 2 separate autonomous navigations are performed on the same path to validate the precision of the localization. Fig 5.8 shows the path where the vehicle drove autonomously from A to E. To compare the localization



results, 5 checkpoints are selected where a camera mounted looking forward is used to capture the scene once the vehicle is arrived at this point.

2 pictures captured at each run are processed by obtaining an absolute difference between each pixel of the images. 5 smaller images from Fig 5.8 show the result of the subtraction. Subtraction of 2 equal images result in zero pixel values, and hence are completely dark. So darker patches indicate better accuracy. The sharp edges of all the images show strong evidence that the synthetic LIDAR localization is performed precisely. The bright spots observed at images B, C and E are actually moving objects.

We should emphasize that we are not explicitly handle the unexpected reading from the sensor, for example a newly parked vehicle or other moving objects. Rather we treat them as part of the sensory noise. In experiment, this has shown to be effective in handling changes occur in an urban environment. Part of what made this possible is the interpretation of sensory data in a 3D model. This give more expressive view of the environment. Although part of the view is hindered by a parked vehicle or any other objects, there are always other features that are still observable beyond the object. In short, as long as the salient features presented in the map are not overwhelmed by the unexpected obstacles, the localization system will work properly.

## 5.2 Road Mapping

In previous section, we discuss how the map can be used to perform localization. In many other scenarios, it is more desirable to obtain interpreted data. Since we are dealing with vehicle traveling in an urban environment, road surface information is critical since this is where the vehicle is expected to travel on. In this section, we describe methods on how the road surface is extracted based on the map by performing region growing and classification.

### 5.2.1 Region growing Method

Region growing algorithm is used to perform segmentation on the map that has similar properties: the road surface. In this work, we make use of the observation

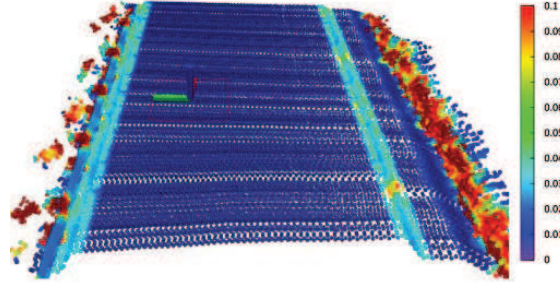


Figure 5.9: Curvature of a road surface

that the road surface is generally smooth in the center, with some apparent edges towards the boundary of the road. Where region growing was originally designed for image segmentation, it can be generalized to 3D point clouds as well [101]. To perform region growing, surface properties  $nx, ny, nz, c$  are calculated for each point cloud on a map  $x, y, z$ , where  $nx, ny, nz$  are the surface normals, and  $c$  is the point's local curvature.

Fig 5.9 shows a short segment of the map with calculated curvature. Observe that the road surface has a low curvature value and higher curvature values exist around the road edges. To extract this planar region, a seed point is selected from within this region. Then the algorithm will recursively travel through this segment and hence the road surface can be extracted. By assuming that the vehicle always travel on the road surface, we can reprocess the map data in a rolling windows style, i.e. simulating a vehicle traveling through the exact same path as how the map is collected. This way each point in the map can be projected into vehicle frame and the seed point is placed at frame's origin. The pseudo-code for the region-growing algorithm is shown in Algorithm 2.

### 5.2.2 Dealing with Noisy Data

While the road surface data can be extracted directly from a map, the observation collected is sparse that contains noisy data. To obtain a good road surface, the data is processed in a cascaded manner. The raw input data is first goes through a pre-filtering classifier to extract only the stable points.

**Algorithm 2:** Pseudo-code for region-growing algorithm

---

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

```

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

```

---

**Pre-filter**

Since road surface extraction relies on the curvature value, it is important to make sure that the scan lines along the  $z$  direction is well aligned. Recall in Section 3.2.4, height measurements are affected by roll and pitch value which is derived from IMU. This value is affected depending on how it varies in a short period of time. The error can be significant when a vehicle is traveling through a bumpy road or a speed bump, for example, where a sudden change in pitch angle results in a large variation on the measured values. This could render a noisy point as large variance in height being falsely segmented as a road boundary.

To actively recognize the noisy measurements, binary classifier is used. A Support Vector Machine (SVM) is trained with feature vectors  $\theta, \phi, \dot{\theta}, \dot{\phi}$  from IMU together with curvature values. The feature vectors from IMU provides a good temporal states of the vehicle to enable good classification. In the training process, a short segment of the map is used for labeling where a wrongly classified road boundary observation is labeled as the negative samples.

Fig. 5.10 shows an example of a pre-filter in road extraction process. The pink segments represent the detected noisy scan and it is then removed from the

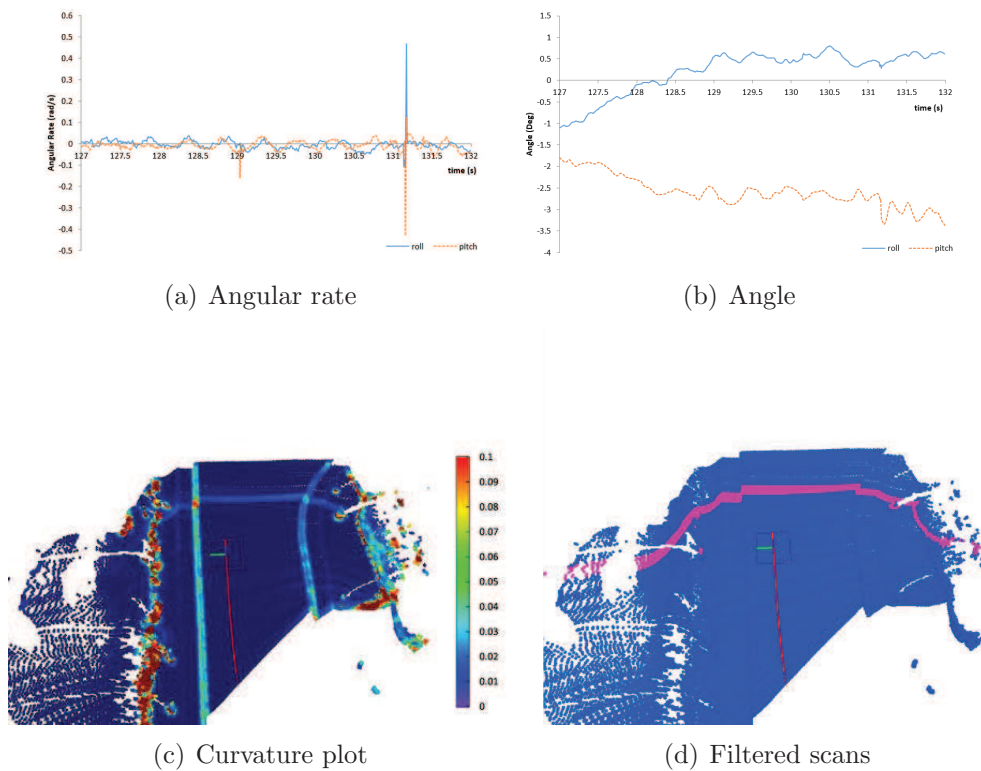


Figure 5.10: Noisy scan rolling window filtering. In the curvature plot, one light blue swathe of points appear in the middle of road are recognized as noise. These points are filtered through the pre-filtering process as shown in (d) visualized as a pink strip.

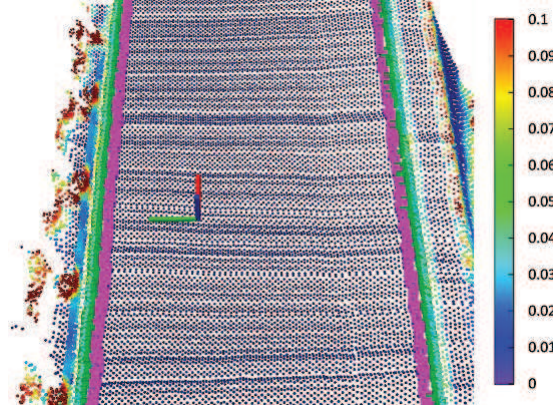


Figure 5.11: Boundary point adjustment

consideration of region growing process.

### 5.2.3 Road Boundary Retrieval

After the pre-filter, the region growing algorithm will proceed to extract road surface with its boundary. A conservative threshold value is chosen to reduce false positives where a road boundary is detected as road surface. However, due to the varying density of the observed point clouds, the road boundary always results in a smaller region of an actual road surface. This is due to the fact that fewer points being observed as scan angle is increased from the center of the LIDAR. This reduces the density of the point clouds and thus fewer supporting points can be used to perform curvature calculation. Consequently, there will be some region where surface is wrongly segmented as road boundary when a single threshold value is used. This can be alleviated by further examining the boundary of each observation, where a hill climbing search is performed until a local maxima is found. Fig. 5.11 shows the adjusted boundary point, where purple points are boundaries before the adjustment, and green points are the adjusted boundary points.

### 5.2.4 Post-Filtering

The road boundary points obtained through the steps described above in general contain a good estimate of the road surface. It is however still susceptible to road surface recognized as road boundary. Here another binary classifier is used to further confirm the boundary points are indeed correct. For the post filtering,

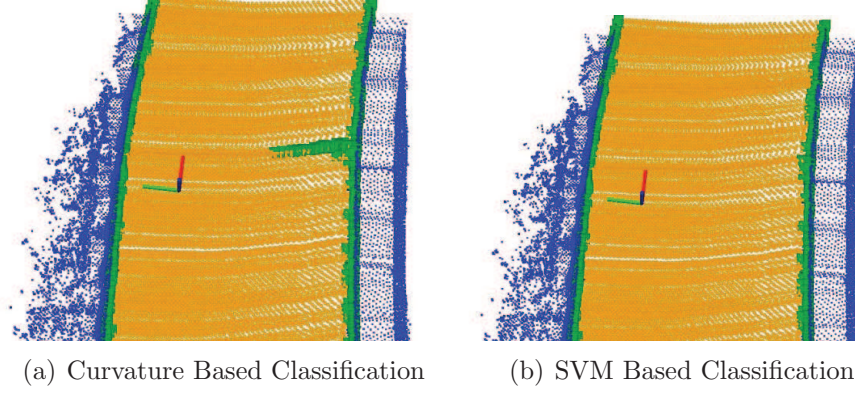


Figure 5.12: Increased classification accuracy through SVM

3 additional feature vectors are used to perform SVM training and classification. There are z-variance, point density and maximum height variance.

The training data is obtained from manually labeled boundary points extracted after boundary adjustment. This post-classification process improves the accuracy of the road boundary classification, and hence the road surface extraction. To train all the classifiers, LIBSVM is used. There are in total 14893 road boundary samples are used. In the samples, there are 2983 labeled as the false boundary points.

We trained 3 different classifiers and compare the post-filtering results. Table 5.2 shows the result of the classification. The “Original” column is the base benchmark with the result obtained directly from region growing. The first classifier (OneR) tries to perform boundary classification using only curve data as the most basic classification. The second classifier (SVM1) adds z-variance as another feature vector and the third (SVM2) uses the full array of feature vectors, i.e. curvature, z-variance, local density and maximum local height difference. The total accuracy is improved as more features are added and Fig. 5.12 shows one instance of improved road detection. The green points are the extracted road boundary and road surfaces represented in yellow color. More results can be found in Fig. 5.13.

Table 5.2: 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



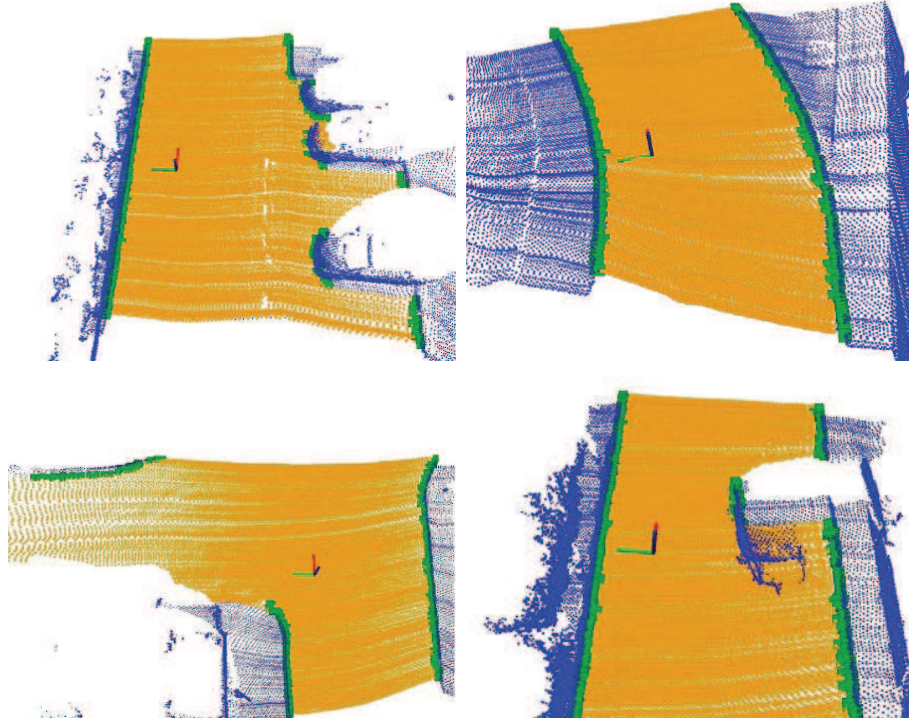


Figure 5.13: Results of the SVM classification of road boundary and surfaces

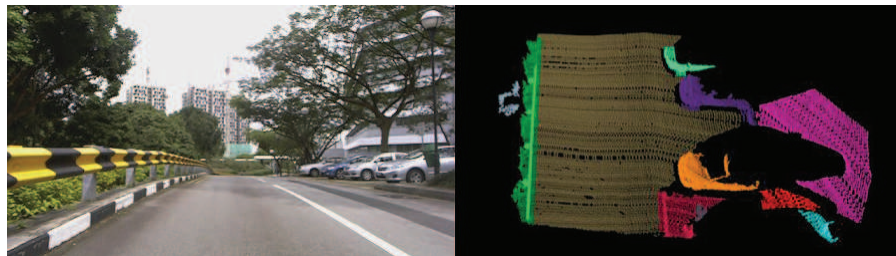


Figure 5.14: Point cloud segmentation example

### 5.2.5 Road Detection for Point Cloud Segmentation

Road surface is the most dominant feature in the urban road environment. The detection of road surface from 3D point cloud can be used for point cloud segmentation of road environment. Since the road surface serves as the supporting ground shared between objects, by detecting and removing the road surface from the 3D point cloud, points from different objects are naturally separated. Then the remaining point cloud can be segmented and clustered easily. Fig. 5.14 shows one example of point cloud segmentation, where different point cloud clusters are shown in different colors.

### 5.2.6 Probabilistic Road Mapping

The road surface detected is put together in a probabilistic fashion sharing a common global frame. The road map can be used for variety of purposes, vehicle path planning for example. In this case, the vehicle operates on a well-defined region, allowing a safe and efficient path planning. It can also provides contextual information for vehicle localization and for other perception purposes [53, 102].

Occupancy Grid Mapping (OGM) [47] is used for this mapping purpose. Since the vehicle poses are already known, the road mapping problem is simplified to an OGM problem, which is to estimate the posterior of the map given the data:  $p(m|z_{1:t}, x_{1:t})$ , where  $m$  is the map,  $z_{1:t}$  is the measurement from start time 1 to end time  $t$ , and  $x_{1:t}$  is the set of vehicle poses from time 1 to  $t$ . By assuming the independence between grids, 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 a map  $m$  [47]. An inverse sensor model  $p(m_i|z_t, x_t)$  is needed for the above estimation process.

The map provides optimized poses which gives an accurate sensor poses at each LIDAR scans. Given vehicle poses from the map, the road mapping problem can be solved by estimating  $p(m|z_{1:t}, x_{1:t})$ . In our application,  $m_i$  is a binary variable with two possible values, road surface and road boundary;  $z_t$  is the extracted road surface and boundary points in the vehicle attached coordinate at time  $t$ ;  $x_t$  is the vehicle pose in the global coordinate. The extracted points  $z_t$  can be transformed into map coordinate given the knowledge of  $x_t$ . Then,  $p(m_i|z_t, x_t) = p(m_i|z_{t_i})$  is obtained, where  $z_{t_i}$  is the point that falls into grid  $m_i$ . The inverse sensor model



is defined as:

$$p(m_i = road\_surface | z_{t_i} = surface\_point) = k_1, \quad (5.3)$$

$$p(m_i = road\_surface | z_{t_i} = boundary\_point) = k_2, \quad (5.4)$$

$$(5.5)$$

where  $k_1$ ,  $k_2$  are parameters to be selected in experiments. Generally  $k_1$  is larger than  $(1 - k_2)$ , reflecting the fact that there are more noise in boundary points than that 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  $p(m | z_{1:t}, x_{1:t})$ .

## 5.3 Graph Learning of Urban Road Network

The extracted road map can be used to perform automatic analysis on a road network. Earlier, we described a method where accurate metric mapping is performed to generate occupancy grid map of road surface and boundary. Based on this metric map, further analysis can be done to extract a topo-metric graph which captures both topological and metric information of an urban environment.

### 5.3.1 Topo-metric Graph

The topo-metric graph captures both topological and metric information of a road network. By encapsulating both topological and metric attributes in the nodes and edges, a topo-metric graph is established. From the topological aspect, each node maintains the IDs of its connected edges. This topological connectivity between nodes and edges captures the basic structure of the road network. From the metric aspect, the graph maintains its metric information via different attributes of nodes and edges. A node has attributes of its position and covering area, while an edge will have a spline representation for its shape, together with its road width.

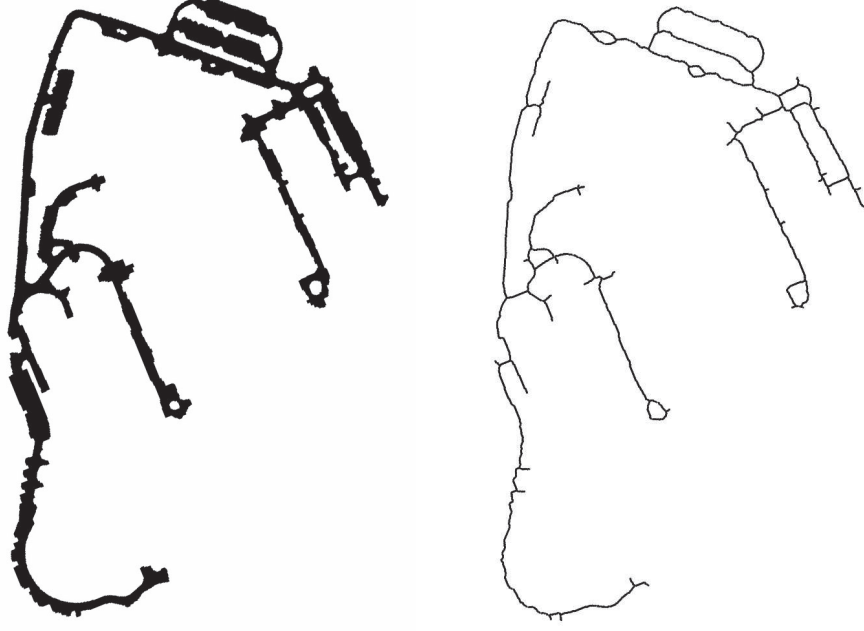


Figure 5.15: An example of road skeleton map after processed from a binary road surface map

### 5.3.2 Road Skeletonization

To construct a topo-metric graph, road skeleton is first extracted using skeletonization process. Skeletonization is a common pre-processing operation in raster-to-vector conversion used in a digital binary image. The digital binary image is obtained by applying global thresholding on the occupancy grid map. We obtained the required binary image (Fig. 5.15(a)) by thresholding the grayscale map, i.e. the white pixels of road surfaces will form the foreground object, while dark pixels of road boundary and gray pixels of unknown areas will become the background.

However, due to noise from the road detection process, the binary image may not be perfect, with defects appearing as a hole at the center of road, or protruding spurs at the sides. The holes appear where the region has low density of surface points, leaving their probability as road surface low. The spurs appear where road boundary points are wrongly classified as surface places, making related grids falsely categorized as surfaces. Usually the size of spurs are bigger than those of the holes. To remove these defects, morphological operations are applied. Since the holes are usually smaller than the spurs, we fill the holes using morphological closing with a smaller mask, and then remove the spurs using opening operation with a bigger mask.

After the binary image is prepared, we apply skeletonization to extract the road skeleton. Image thinning method is employed for the skeletonization purpose, which have several beneficial properties: it preserves the shape and topology of the original object, and forces the skeleton to be placed in the middle of the object. While there are many image thinning algorithms available in the computer vision literature, we choose two-subiteration thinning algorithm in [103], which produces a single-pixel width skeleton with few redundant pixels (right in Fig. 5.3.2).

### 5.3.3 Topological Structure Learning

The skeleton obtained previously captures the topological and shape information of the road network. We will use this to construct a topo-metric graph.

After skeletonization using image thinning, a skeleton with single-width pixel is obtained that can contain a few redundant pixels. A redundant pixel appears in the skeleton does not affect the skeleton of the road network. It is neither an endpoint nor a link that could break a road skeleton, i.e. deletion of a redundant pixel does not in any way alter the skeleton. In an ideal case where a skeleton has no redundant pixels, a pixel from a road link should not have more than two connected neighbors, while a pixel at one intersection should have three or more.

To remove the redundant pixels, we iterate through all the points in the skeleton except for the endpoints, and loop through a verification process. In the checking process, the pixel under examination is first turned off and a connected component analysis is performed on its 8-connected neighbors. If there is one connected-component formed by the 8 neighbors, then this pixel is a redundant pixel to be removed.

Given the minimal skeleton with no redundant pixels, it is trivial to build the topological structure of the road network. We first extract intersection pixels which have three or more neighbors, and store them as the nodes of our topo-metric graph. Then the intersection pixels are deleted from the skeleton, decomposing the skeleton into disconnected road links. These road links will serve as edges of the topo-metric graph. In the end, the connections between nodes and edges are constructed.

### 5.3.4 Metric Property Learning

An intersection can have metric properties that include its position and its covering area. For a road link, the metric properties include a spline representation of its shape, and the average road width. It should be clarified that these metric properties are learned in the image coordinate of the metric map, and can be easily transformed into real-world global coordinate given the map's origin and resolution.

Road links are the most fundamental components of a road network. While a road link is originally extracted as a string of connected pixels, a more compact representation can be used where each road lane is model after a spline curve [104]. Cubic spline is chosen to guarantee that a road link has continuous curvature. We use chord length parameterization to ensure that the parameterized length is proportional to the road length. Least square fitting technique is applied to obtain an optimal spline given the input pixels. The cubic spline incorporates the shape information of a road link, which is compact and precise, and represented by a continuous mathematical model. Besides a cubic spline, another metric property of a road link is its width. By assuming that a road link has uniform width, the road segment is represented with the average value of widths measured along the link.

An intersection is a junction with three or more road links. Its position and covering area are of vital information for vehicle navigation. We define the position of an intersection to be the actual position of the intersection pixel, and the covering area to be approximated as a radius centered on this position. Fig. 5.16 shows an example of a complete topo-metric graph map, where each edge is visualized as a thick spline with the thickness corresponds to the road's width. Each intersection is marked by a yellow circle with its radius represents covering area.

## 5.4 Summary

The map can be used for many purposes. The most direct application is localization. The contribution of the thesis was made through the notion of achieving highly accurate positioning estimation using minimalist approach. In particular, virtual sensors are constructed from curb and intersection points to perform lo-

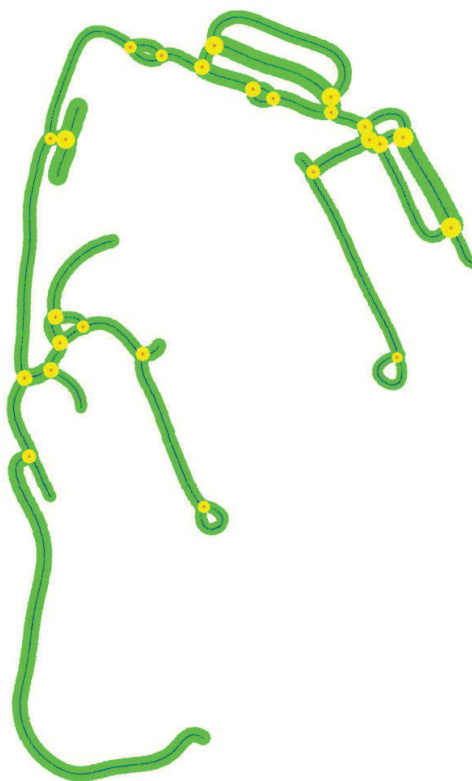


Figure 5.16: A topo-metric graph map of NUS engineering

calization. Using these 2 virtual sensors, we show that it is sufficient to perform autonomous navigation using information derived from a single LIDAR. This is further expanded through the use of synthetic LIDAR. We show how it can be used not only for mapping purposes but can also be used for localization. Through experiments, we show how a synthetic LIDAR is adaptable to many different kinds of environment while not restricted to road environment with curbs and intersections.

For a vehicle to travel in an urban environment safely, road surface information is an important metric as this is where the vehicles are allowed to perform navigation tasks. Although a raw map can be used directly for localization, we showed how the map can be interpreted through road mapping. Contribution was made on how to deal with noisy, real sensory information available to perform road network extraction. Through a multistage process that includes pre- and post-filtering, we performed road mapping probabilistically. With the road surface map extracted, urban road network can be learned to generate a road network graph.

We also proposed how a map can be further processed to obtain a topo-metric graph, where it captures both topological and metric information of road networks.

This is done by using the extracted road surface to obtain a skeleton map. Then, the road links and edges can be extracted from the single pixel road surface map. The topo-metric map is completed by inserting road width and intersection covering areas.

# Chapter 6

## Activity Mapping

While so far the work has been focused on static structure, this chapter discusses on how moving objects can be learned as part of the knowledge in the mapping framework. This is realized through an activity map, where it stores a summary on how objects move within a region of the map. The method that is presented here focuses on pedestrians as the moving objects, since they are not moving as fast relative to the speed of a golf cart. But the same concept we are proposing can also be extended to vehicles or other moving objects.

### 6.1 Gaussian Process for Pedestrian Modeling

The activity model learning is well established in computer vision community, where researchers have proposed various methods to learn pedestrian motion patterns. Some representative works can be found in [105, 106]. However, most of the works assumed a static observers where a stationary camera is used with the ability to observe complete trajectories of pedestrians. This is not a valid assumption for applications using mobile robots. This is because a moving robot will most likely only able to observe a small segment of a pedestrian's track. Given that most sensors have line-of-sight limit, it is susceptible to occlusions with limited visibility.

Lookingbill *et al.* in [107] uses a helicopter to identify moving objects on the ground to learn their motion patterns. It shows interesting results and enlightens us about representing motion pattern in the form of grid map. However, it only estimates the motion patterns in a grids where moving objects are observed, it

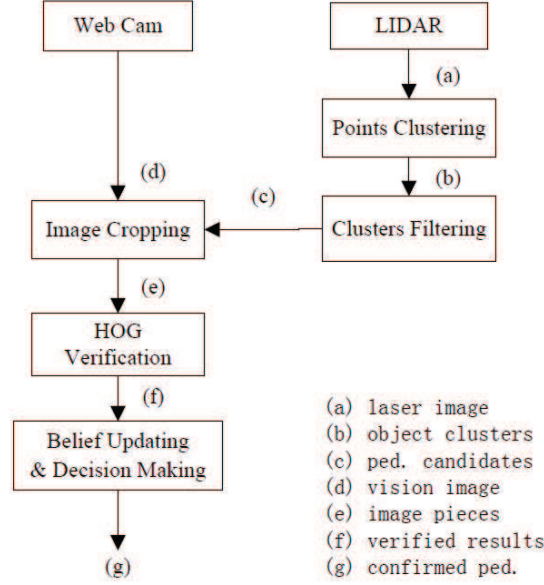


Figure 6.1: Pedestrian detection algorithm

also neglects the relationship between neighboring places. In our work, we use Gaussian Process to learn the activity model of the entire environment, which is able to overcome the above problems.

### 6.1.1 Pedestrian Detection and Tracking

Pedestrian detection and tracking is the basic requirement for pedestrian activity learning. In our work, on-board sensors are used. A laser range finder is used for moving object tracking, and a camera is used for pedestrian verification. Fig. 6.1 shows the overall flow of the algorithm and Fig. 6.2 shows the result of the detection algorithm for a single data frame. The algorithm runs in two phases: pedestrian candidate detector and pedestrian verification.

In pedestrian candidate detector phase, the LIDAR is segmented and clustered based on their position and relative velocity (Fig. 6.3b). Potential candidate clusters for pedestrians are filtered out based on their size and velocity where a simple linear velocity model is used in our implementation. For pedestrian verification part, a common webcam is used to verify whether extracted objects are pedestrians. Extrinsic calibration of webcam and LIDAR is done beforehand, then the candidates are projected as a region of interest in the webcam image correspondingly. The whole image is then cropped into several smaller sub-images. This is in



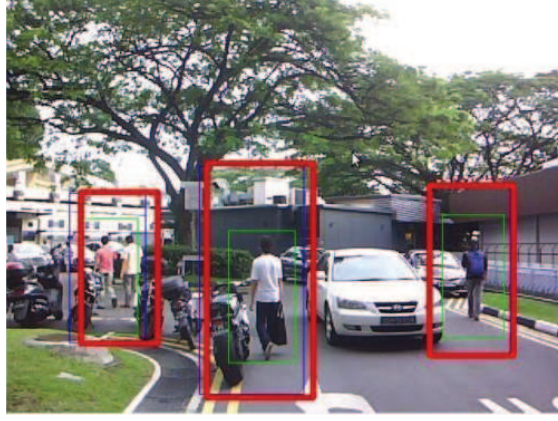


Figure 6.2: Snapshot of the onboard pedestrian detector

order to keep the computation as low as possible with a more focused region. Then, Histogram of Oriented Gradient object detection (HOG) is used. We found that the default trained people detector from OpenCV is sufficient to detect pedestrian. A faster verification is made possible through GPU accelerated HOG algorithm. By labeling the LIDAR track accordingly, we are able to recover a complete track of a pedestrian that comes within the view of the LIDAR. This reduces the computation load significantly allowing us to run such detectors in real time.

Fig. 6.3 shows how the system perform pedestrian detection and tracking. In Fig. 6.3f, the pedestrian first gets tracked by LIDAR, showing as white track. As the pedestrian enters the field of view of the camera, the object get verified as a pedestrian and the tracks turn green. This way we can recover the whole track by back-tracking the path.

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 coordinate of the vehicle, we transform the track information to the map frame. Thanks to the use of mapping and localization method discussed previously (Section 5.1), the transformation can take place in real time.

### 6.1.2 Track Classification and Clustering

After the pedestrian tracks are obtained, track clustering is performed. The purpose of the clustering is to cluster heterogeneous tracks into different homogeneous groups. While related works usually have pedestrian tracks that are clustered into

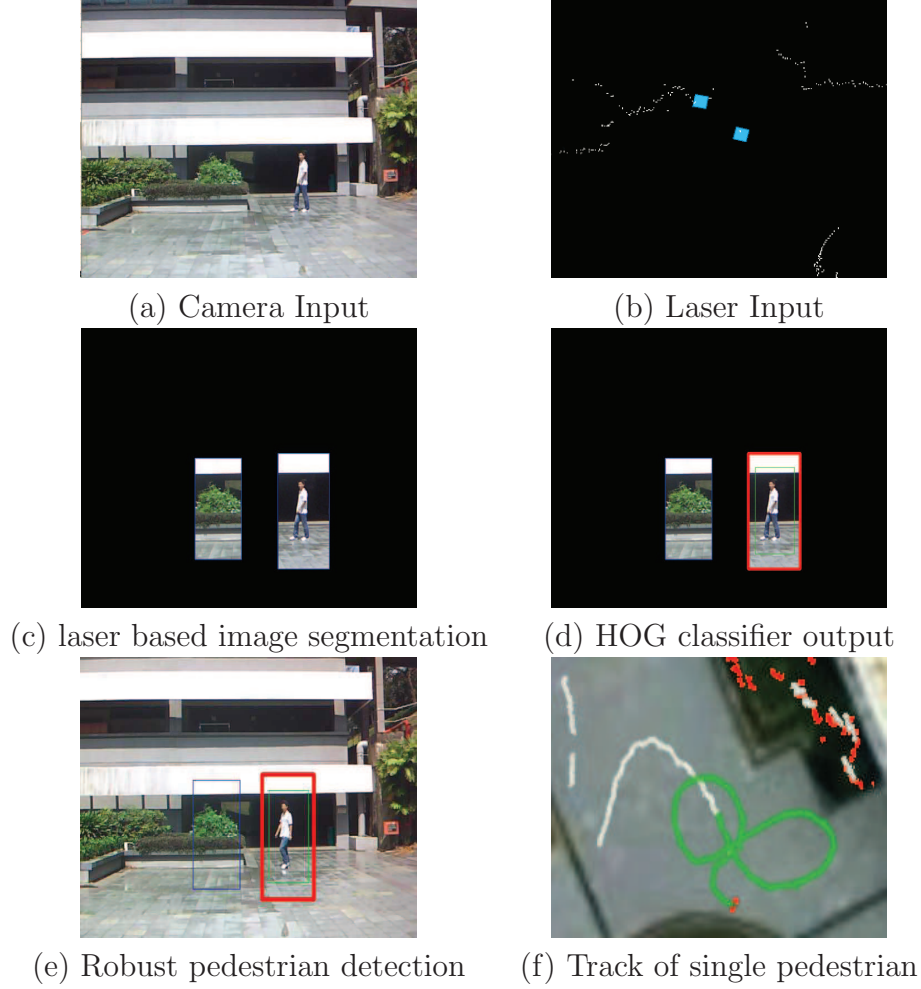


Figure 6.3: Pedestrian detection module

multiple groups of high similarity. In our work, however, data collection occur over a mobile platform works in a fairly large area. Hence it is going to collect pedestrians from many different heterogeneous motion types. This would result in a very involved computation if we were to try to perform clustering and learning the activity model for each of these types.

Instead, we simplify the problem by clustering moving tracks scattered on the map based on the directionality of the track. In fact, when viewing an urban environment microscopically, a place in an urban road environment can be described by two dominant motion patterns of pedestrians, where they share a similar speed but opposite directions. We call this a “bidirectional property” of pedestrian activity. This “bidirectional property” simplifies our clustering problem: we cluster the tracks into two groups, and only need to guarantee that the activity of each group is consistent at the microscopic grid level.

The clustering problem is formulated as follow: Given a set of pedestrian tracks, denoted as  $S = \{s_1, \dots, s_m\}$  where a single track  $s$  is a set of position-speed-angle tuples:

$$s = \{t_1, \dots, t_n\}, t_i = \langle x_i, y_i, v_i, \theta_i \rangle \quad (6.1)$$

where  $x_i, y_i$  are the pedestrian positions,  $v_i$  the speed, and  $\theta_i$  the moving direction. The similarity between two tuples,  $p, q$  are defined as:

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

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

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

The overall problem is to find two clusters of tracks A and B given input clusters  $S$ . 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  $\alpha$  and  $\beta$  respectively.

Alg. 3 shows the pseudo-code of the clustering algorithm. The clustering process starts with the longest track in the input tracks  $S$ . Then, the track that has the smallest similarity value is selected as track cluster B, with its tuple set  $\beta$ . Then the remaining tracks are perform similarity test recursively by comparing the tracks to cluster A and B and assigning them to the cluster that has the higher similarity value.

### 6.1.3 Activity Learning with Gaussian Process

Now that we have got 2 groups of tracks consist of moving clusters, each track share similar motion pattern on the grid-level that is to be learned. For the learning procedure, Gaussian Process (GP) is used. A GP is completely determined by its mean and covariance functions. Please refer to [108] for more details on Gaussian Process. In our work, activity learning can be model as a Gaussian Process Regression (GPR) problem. The set of position-speed-angle tuples for each cluster are used as the observation input, and we ought to find a predictor that outputs a pedestrian's speed  $v_{ij}$  and angle  $\theta_{ij}$ :  $\mathfrak{A}_{ij} = \{\bar{v}_{ij}, \sigma_{v_{ij}}^2, \bar{\theta}_{ij}, \sigma_{\theta_{ij}}^2\}^T$ .

It is worth noting that angle value is circular in nature, unlike a speed value

**Algorithm 3:** 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;

```

---

which is a linear variable spanning from  $(-\infty, +\infty)$ . A simple example is when subtracting between 2 angles, say  $1^\circ$  and  $360^\circ$ , the difference is  $1^0$ . This is overcome by modeling the angle distribution as a Projected Normal Distribution [109], where the angle can be inferred from the bivariate normal distribution of speed vector  $\vec{v} = (v_x, v_y)$ .

We used 3 separate GPR to obtain the posterior distribution of  $v, v_x, v_y$  for each point in a map  $X_{ij}$ . The tuple set of each cluster A,B are used as the training data that fed into the GPRs. In the training, zero mean function  $m$  and squared exponential covariance function  $K$  are used, which is given by

$$m(X) = 0, K(X, X') = \sigma_y^2 \exp \frac{-(X - X')^2}{2l^2} \quad (6.2)$$

The  $X$  here is the 2-dimension position vector in global coordinate,  $X \in R^2, X = (x, y)$ . By letting  $Y$  be the output of GPR, our GPR model is defined as follow:

$$Y = F(X) + \xi, F \sim GP(m, K), \xi \sim N(0, \sigma_n^2) \quad (6.3)$$

where  $F(X)$  is a function distributed as a GP with mean function  $m$  and covariance

function  $K$ . It is shown that the output of  $Y$  is also has a GP distribution with  $Y \sim GP(m, K + \sigma_n^2 \sigma_{ii'})$ , with  $\sigma_{ii'} = 1$  iff  $i = i'$ . This allows us to give an input set  $(X, Y)$  as training data, and obtain the posterior distribution for a set of test points  $X^*$  which has Gaussian distribution  $Y^*|Y \sim N(m(X^*) + K^T(X, X^*)K^{-1}(X, X)(Y - m(X)), K(X^*, X^*) - K^T(X, X^*)K^{-1}(X, X)K(X, X^*))$

By using log-likelihood maximization,  $(\sigma_n, \sigma_y, l)$  are learned from the training data. As explained earlier, a pedestrian moving angle's probability density function is model after Projected Normal Distribution (PND). Let  $\vec{x}$  be a two-dimensional random vector which has a normal distribution  $N_2(\mu, \Sigma)$ , then, the angle of  $x$  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 obtained as follow:

$$\frac{\vartheta(\mu; 0, \Sigma) + |\Sigma|^{-\frac{1}{2}} D(\theta) \Phi(D(\theta)) \phi(|\Sigma|^{-\frac{1}{2}} (x^T \Sigma^{-1} x)^{-\frac{1}{2}} \mu \wedge x)}{x^T \Sigma^{-1} x} \quad (6.4)$$

where  $\vartheta(\mu; 0, \Sigma)$  denotes the value of the probability density function for  $N_2(0, \Sigma)$  at point  $\mu$ ,  $\Phi$  and  $\phi$  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  $\mu = (\mu_1, \mu_2)^T$ .

Finally, to calculate the PND distribution, normal distribution of pedestrian speed vector  $\vec{v}$  is used. Assuming that the marginalized distribution of  $v_x$  and  $v_y$  is independent, where:  $\vec{v} \sim N(\text{diag}(\mu_{v_x}, \mu_{v_y}), \text{diag}(\sigma_{v_x}^2, \sigma_{v_y}^2))$

Calculation of the probabilistic density function of moving angle is calculated with this bivariate normal distribution. Mean of the distribution,  $\bar{\theta}_{ij}$  is the pedestrian moving angle with its variance representing the uncertainty of this moving angle  $\sigma_{\theta_{ij}}^2$ . Detailed definition and calculation of circular mean and variance can be found in [109].

#### 6.1.4 Bidirectional Property of Pedestrian Activity

Recall that in the earlier part we cluster pedestrian tracks into 2 groups according to a similarity measure, then the activity model is learned independently. As more data are collected, it was found that these 2 activity models have reciprocal properties, i.e. they are mirror of each other where the moving direction is opposite

to each other. Hence, it is reasonable to assume that pedestrian activity at one place is often bidirectional. This make sense, as pedestrian moving in between the buildings, and the doors act as both entry and exit points for the occupants. This assumption allows us to learn the activity model of single track cluster, by adding the angle values in the second activity tuples with  $180^\circ$ . While in a wide open space area, this assumption (walking in opposite directions) may not be valid, since the pedestrian tracks may have many different directions. But in our environment where the autonomous vehicle operates, this assumption holds.

### 6.1.5 Activity-based Semantic Mapping

Further activity related analysis can be done now that we have the activity map of a pedestrian track. One application is to use the pedestrian and road link direction as a feature to infer a place's semantics.

Here, we want to perform two layers of semantic map, the first layer is to identify “pedestrian path” (“PP”), and a more refined layer that recognize three different types of functional areas from the path, namely the “entrance/exit” (EE), “crossing” (CR), and “sidewalk” (SW) areas. These areas are not necessarily mutually exclusive, as the same area can share different purpose at any time. To capture the semantic properties at place  $m_{ij}$  in the map, a semantic vector of four binary variables is introduced as:  $\mathfrak{S}_{ij} = (p_{ij}, e_{ij}, c_{ij}, s_{ij})^T$ , 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”,  $\Lambda_s = \{\text{SW}, \text{non-SW}\}, s_{ij} \in \Lambda_s$ ;

Then, the given information are the activity information  $\mathfrak{A}_{ij}$ , and prior road network information.

### 6.1.6 Pedestrian Path Learning

#### Pedestrian Intensity

In an urban road environment, a pedestrian path usually located at the side road with a raised walkway. This can be inferred from the frequency of pedestrian appearance. In other words, when there are more pedestrians passing through one particular place, the more likelihood it is be part of a pedestrian path. Based on this, a measurement of “pedestrian intensity” as a feature for pedestrian path classification is used.

Let’s define that the intensity at place  $m_{ij}$  is denoted as  $I_{ij}$ , then the intensity is a function of pedestrian count

$$N_{ij} : I_{ij} = I_{local_{ij}} \times I_{global_{ij}} \quad (6.5)$$

where  $I_{local_{ij}} = N_{i,j} / \max_{a,b} (N_{i+a,j+b})$ ,  $I_{global_{ij}} = \frac{1}{1+\exp(-N_{ij}+N_{exp})} - \frac{1}{1+\exp(N_{exp})}$ ,  $a, b \in Z \cap [-l/2, l/2]$ .

The pedestrian intensity is the multiplication of these two factors, the local factor and the global factor, denoted by  $I_{local_{ij}}$  and  $I_{global_{ij}}$ . The local factor is used for pedestrian count normalization by making use the maximum counts in a  $l \times l$  local window. This normalization is important as the observations were made at different time using a mobile platform. Without this normalization factor, the difference in pedestrian count at different areas can become too large leading to an unbalance tracks. One can think of this as having a similar effects with adaptive threshold commonly used in image processing, in which it is used to recover details in an image that has varying brightness in the background. The global factor is namely a logistic function of  $N_{i,j}$ , which increases quickly when  $N_{i,j}$  is nearby  $N_{exp}$ , while changes slowly when far away.  $N_{exp}$  is a constant value chosen as the expected pedestrian count at a “path” place.

Based on pedestrian intensity calculated from the previous step, Markov Random Field (MRF) is used for path classification. MRF is a popular technique in image processing, which captures the dependency between neighboring pixels. It is widely used for image segmentation, restoration and other purposes. A seminal work by Kindermann *et al.* explains this in details [110].



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  $\sigma_{p_{ij}}$  is given, which can be obtained from the above learned model, the energy function is defined as  $U = \sum_{i,j} \left( \log(\sqrt{2\pi}\sigma_{p_{ij}}) + \frac{(I_{i,j} - \mu_{p_{ij}})^2}{2\sigma_{p_{ij}}^2} \right) + \sum_{i,j,h,k} \beta \delta(p_{ij}, p_{hk})$  where  $p_{ij}$  and  $p_{hk}$  are “path variables” of neighboring places  $m_{ij}$  and  $m_{hk}$ ,  $\beta$  is a weighting parameter,  $\beta \geq 0$ . Then, optimization can be done by minimizing this energy function, denoted as  $\hat{P} = \{\hat{p}_{ij}\}$ .

### 6.1.7 Refined Semantics Learning

After the coarse-level semantic learning for pedestrian path, a refined semantic is performed to search for “entrance/exit”, “crossing”, and “sidewalk” areas along the path with Naive Bayes Classifier (NBC). By assuming independence between features given the class variable, the probability model is:  $p(C|F) = p(C|F_1, \dots, F_n) = \frac{1}{Z} p(C) \prod_{i=1}^n p(F_i|C)$ , 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,  $p(F_i|C)$  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 include “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  $\hat{p}_{ij}$  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 pedestrian path, it is not likely to be some functional area.
- $F_{d_{ij}}$  is about the angle of pedestrian moving direction.  $\bar{\theta}_{ij}$  in  $\mathfrak{A}_{ij}$  is chosen as its value. This feature carries the typical motion information at each grid, 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 is the calculated



moving direction  $F_{d_{ij}}$ .

- $F_{p_{ij}}$  is about a place's relative position to the road network. This feature is introduced with the idea that the functional semantics of a certain place is actually related to its position on the road.

Using the maps generated previously for the road network, i.e. the binary grid map and topo-metric map, the maps are used to obtain the prerequisite positional information for the following semantic reasoning process. To give some examples, road boundary can be obtained based on the binary grid map and the direction of a road link can be obtained using a topo-metric map.

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

The knowledge of pedestrian entrances and exits in a road network is of vital importance and can help an autonomous vehicle's safe navigation. Pedestrians' entrance/exits are defined where pedestrians enter onto or depart from the road. Using bidirectional property, we can assume that the same pathway is used for entrance as well as exit in a single spatial region. Based on the feature set  $F_{ij}$ . The feature models are built as below.

i)  $p(F_{pp_{ij}}|e_{ij})$ : The entrance/exits (EE) are functional areas of pedestrians, which should only appear on pedestrian path. If an area is "EE", it should be "PP". The infinitesimal  $\epsilon$  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 two 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 moving direction estimation. This feature model is the same for other two NBCs.

$$\begin{aligned} p(F_{dv_{ij}}|e_{ij} = \text{EE}) &= \frac{2(\max_{i,j} F_{dv_{ij}} - F_{dv_{ij}})}{(\max_{i,j} F_{dv_{ij}} - \min_{i,j} F_{dv_{ij}})^2} \\ p(F_{dv_{ij}}|e_{ij} = \text{non-EE}) &= \frac{1.0}{\max_{i,j} F_{dv_{ij}} - \min_{i,j} F_{dv_{ij}}} \end{aligned}$$

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 road,  $\text{EE}_r$  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  $[0, \frac{\text{road\_width}_{ij}}{2.0}]$ .

$$\begin{aligned} p(F_{p_{ij}}|e_{ij} = \text{EE}) &= 1.0 - \frac{F_{p_{ij}}}{\text{EE}_r} \\ p(F_{p_{ij}}|e_{ij} = \text{non-EE}) &= \frac{2.0}{\text{road\_width}_{ij}} \end{aligned}$$

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 grid. However, these results are in the format of individual grids, we want to further cluster them into individual EE objects. Gaussian Mixture Model (GMM) is used for the clustering purpose, with which EE grids of places are clustered as EE objects. Each EE object corresponds to a 2D position in the map. Bayes Information Criteria is used to select the best cluster number. After the clustering, a set of EE objects  $\xi\xi = \{\text{EE}_1, \dots, \text{EE}_n\}$  is obtained.

**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 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 neighboring areas. In our application, we treat a place’s probability of “crossing” as a feature, and input 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, MRF is used to generate more homogeneous results for  $s_{ij}$ , as for  $c_{ij}$  discussed previously.

**6.1.8 Experiment Results**

Fig. 6.4 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.

Given the results from the track classification and clustering, we try to learn the activity model using Gaussian Process. As discussed in Section 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. Fig. 6.5 illustrates the pedestrian moving direction  $\bar{\theta}_{ij}$  of the learned activity model. The direction values are shown by red arrows, which is overlaid onto the satellite image for visualization. We can have a glance of the pedestrian motion flow in the environment from this figure.

Figure 6.4: Track clustering results

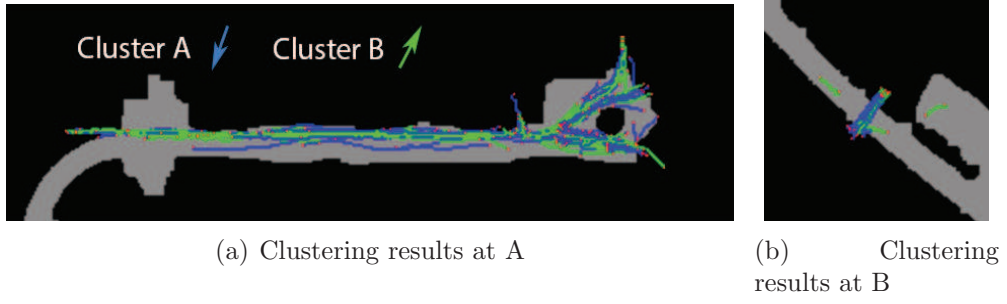
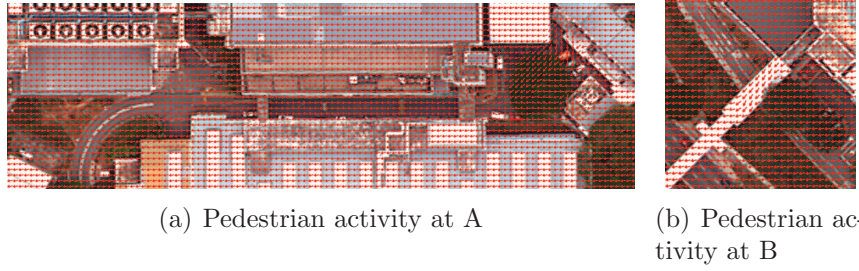


Figure 6.5: Moving direction of activity model



### Activity-based Semantic Mapping














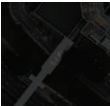

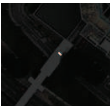
Together with the road network information, semantic mapping can be performed. Tab. 6.1 shows the mapping results for each of the 4 predefined semantic properties: pedestrian path, entrance/exit, crossing and sidewalk.

For the pedestrian path property, “pedestrian intensity” shows the normalized frequency of pedestrian appearance. The white line exhibit a smoothed gradient, this shows the effectiveness of the normalization factor although incomplete pedestrian tracks appear during data collection. The classification results from MRF successfully retrieve complete path with no false positives.

For the entrance/exit property, the output probability of NBC is shown in the row labeled as “EE probability”. Since the classification results obtained from NBC are made up of individual “EE grids”, GMM is used to cluster these grids. “EE objects” shows the clustering results labeled with different colors. The clusters are selected automatically with Bayes Information Criteria, which successfully extract 7 entrances/exits in both Area A and Area B, which corresponds exactly to the ground truth.

For the crossing property, we are able to recognize the crossing area in Area B. However, some grids in Area A where no crossing exists are misclassified as “cross-

Table 6.1: Mapping results for semantic properties of the four types

	Area A	Area B
Pedestrian intensity		
Path classification		
EE probabiliy		
EE objects		
CR probability		
CR classification		
SW probability		
SW classification		

ing”. The accuracy for the classification result is 80.3% which can be improved by further filtering out those small pieces of areas according to their size.

For the sidewalk property, we recognize a long sidewalk in Area A, which has several disconnected pieces at the right end, as shown in “SW classification”. According to our definition of pedestrian sidewalk, these disconnected pieces indeed have “sidewalk” property.

In summary, our activity-based semantic mapping corresponds well to real environment, where all of the predefined semantic properties are mapped well in our two survey areas.

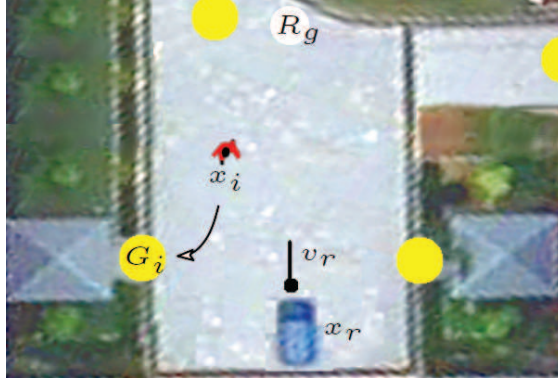


Figure 6.6: Pedestrian avoidance scenario

## 6.2 Motion Planning with POMDP

In previous section, we show that information about pedestrian tracks can be extracted as an activity map with its semantics information. In this section, an application where the semantic information obtained is used for pedestrian avoidance is presented.

In order to have a more principled way to handle pedestrian avoidance, an approach to integrate pedestrian intentions into motion planning for autonomous vehicle has been proposed [111]. We argue that in many cases, traditional approaches of proximity based reactive avoidance, or picking the most likely pedestrian intention failed to generate a safe and successful avoidance.

The solution in predicting the intention of the pedestrian is formulated as a Partial Observable Markov Decision Process (POMDP). In the formulation, the assumptions of finite number of pedestrian intentions, known environment and pedestrian models are used. More formally, for a known environment  $\omega$  and known possible pedestrian goal location  $\{g^k\}$  in  $\omega$ , find an optimal policy  $\Pi$ , to minimize the time taken for robot  $R$  to reach its goal  $R_g$  in the presence of multiple pedestrian  $p_i$  moving towards  $G_i \in \{g^k\}$  where  $G_i$  is unknown to the robot.

We formulate the avoidance of  $p_i$  by discrete Mixed Observability Markov Decision Process (MOMDP),  $M_i : (X, Y, A, O, Z, T_X, T_Y, R, \gamma)$ . Where  $X$  is the observed state variables (The robot's state: location  $x_r$ , velocity,  $v_r$  and pedestrians state:  $x_i$ ).  $Y$  is the unobserved state (The pedestrian's intention),  $A$  is the robot actions (cruise, accelerate, decelerate),  $O$  is the set of all possible observations,  $Z$  the observation function,  $R$  the reward function and  $\gamma$  the discount factor. The





(a) Avoidance for a single pedestrian (on-board camera picture merged at different times to show the evolution of the belief)



(b) Avoiding multiple pedestrians. (goals reordered and labeled differently than (a)). Each pedestrian generates a avoidance MOMDP and the beliefs are shown based on asynchronous set of observation history for each pedestrian.

Figure 6.7: Pedestrian crossing experiment

transition function  $T_x$  and  $T_y$  each gives the transition of the observed variables from current state to future state upon taking an action  $a$  and transition of the pedestrian intentions  $G_i$ .  $T_x$  incorporates the pedestrian and the robots motion models, in which we assume the motion of the pedestrian to follow a trajectory towards its intended goal in a shortest possible path, similar to [112]. Finally, policy is obtained by applying SARSOP [113], a leading point-based approximation algorithm into our model. To handle multiple pedestrian, we treat each pedestrian independently and combine the actions in a conservative manner.

We tested the algorithm on a real pedestrian crossing in NUS campus. To perform the experiment, the vehicle is required to navigate through a pedestrian

crossing autonomously while running the proposed motion planner. In this section of the road, the vehicle not only needs to detect the pedestrians and their intentions on the pedestrian crossing but also deal with jay walkers, a common phenomena in a campus environment. Fig. 6.7(a) shows the robot interacting with such a pedestrian. There are 3 sections in the figure. The section on the top left shows the on-board camera view, bottom left shows the evolution of belief over pedestrian's goal as the pedestrian moves towards its intended goal, with the color of each line encodes the goal index. Also shown in the same section is the commanded speed. The right section shows the simplified environment representation used for solving the policy. The green box represent the location of the robot and orange curve shows the pedestrian track.

When a pedestrian is detected, initial belief over the goals are equal. At this moment, the robot stops due to high initial uncertainty. As the robot continues to observe the pedestrian's intention, the belief values over G1 and G2 increase while G0 and G3 drop. This reflects the pedestrian wants to move towards the other side of the road. Slowly the belief over G1 grows stronger as the pedestrian starts moving diagonally. The vehicle starts moving as soon as the belief over G1 is sufficiently large. Fig. 6.7(b) shows the vehicle responding to multiple pedestrians.

The formulation of the problem allows us to elegantly handle multiple pedestrian without the need to encode the desired behavior of the robots explicitly. Given the suitable model of the pedestrian and the right assumption of observability, real life experiments shown that it can handle multiple pedestrians robustly. Even though the approach was presented for pedestrians on the road, such an approach could also be used to identify intention of other drivers on the road. With proper adaptation and generalization, the same approach can be implemented in an unstructured indoor environment.

## 6.3 Summary

Activity mapping tries to recognize motion patterns within a region of a known map. In this work, we used combination of LIDAR and camera mounted on a moving vehicles to model pedestrians activity. We assumed a bidirectional property



to solve the problem, where the recorded pedestrian tracks are first clustered into 2 main groups to obtain pedestrian activity distribution through a Gaussian Process Regression Model. We show that the activity map can be used to learn semantic properties of a map.

The semantic property is used to perform pedestrian avoidance with POMDP. In this work, intention of the pedestrian is formulated with known goals. Experiment shows that the approach can handle single and multiple pedestrian in real time.

# Chapter 7

## Conclusion

A map is the key enabler for autonomous navigation in an urban environment. In this thesis, we discussed how to build a map using minimal, self contained sensors using LIDAR and vehicle's odometry. We started the discussion showing how a single LIDAR can be augmented into curb and intersection virtual sensors. Later, the same approach is generalized by introducing the construction of a synthetic LIDAR. The synthetic LIDAR, with its full 3D point cloud modeling, has been shown to be able to produce expressive feature points under different urban environments. Using full 3D perception, one could perform road surface and boundary detection without using strong assumption. By processing 3D point clouds obtained from an optimized map, temporal relationship between adjacent scans are used implicitly. This way it is more robust to noise as features of each point will incorporate the information from their neighboring points too.

Using synthetic LIDAR, we show how a map can be constructed in a consistent manner at different places. To perform mapping we collected dataset that consist of raw sensor data from LIDAR, IMU and encoders. In all the dataset, an instrumented vehicle is driven manually through the region, then the collected data is used to create the map. The instrumented vehicle has to be driven around this new environment first to create a map. Then the map is used to perform navigation. The data structure of a map is designed in the same way on how the graph is constructed. Since each node in the graph correspond to a single LIDAR scan, a single node in the graph contains data of the raw LIDAR, the processed synthetic LIDAR, and the odometry measurement. The map also maintain a list of nodes that are

connected through loop closure. A pose graph, which represent the map is sensitive to a wrong loop closure. A single false closure will render a map useless with inconsistent representation of the real environment. We alleviate these limitations by avoiding a wrong loop closure in the first place. By performing scan matching verification, only matching results that are consistent is added to the graph. This ensures a consistent graph. Under the circumstances where wrong loop closure is added, large difference in the residual from the results of optimization is used to remove the loop closure, thereby recovering a consistent map.

With a map, localization can be performed. We show how virtual sensors, curb and intersection sensors can be used to perform autonomous navigation given a known map. The same can be said when performing localization using synthetic LIDAR. With this approach, we are able to achieve higher precision while using only one LIDAR. The map can also be used for other applications. We show how the map can be processed to extract accurate road surface information. This road surface information is then used to automatically produce a topo-metric graph. In this format, both the road network connectivity and its metric information (road width and intersection covering area) are captured automatically.

Lastly, semantic mapping method based on pedestrian activity patterns in the urban road environment is presented. By associating activity information with the road networks, different functional areas corresponding to certain types of activities can be inferred. This is an example of how 3 different dimensions of a map: metric, semantic and activity are jointly correlated. Hence, different kinds of information can be extracted that describe a specific function in one specific area in a map.

## 7.1 Contributions

In this thesis, mapping in urban environment for autonomous vehicle is discussed. The use of one LIDAR in a push-broom configuration shows the potential of using single planar sensor. The idea of using minimal sensor allows one to focus on what kind of information could be used to produce a robust solution. We thus propose using curbs and intersections as virtual sensors. Synthetic LIDAR make use of the full potential of a single LIDAR by using rolling window to reconstruct the

environment. Since we require a real time performance while at the same time workable in almost any kind of urban environment, we found that vertical surface as the feature descriptor works very well.

The mapping framework is tested in many different types of environment, e.g. campus, major road network near train stations, parks and gardens. To ensure a good match between 2 synthetic LIDAR, Extended Correlative Scan Matching is developed that performs matching using features available from a point measurement. Acceleration through GPU is worth noting that when it is used correctly, it can produce a tremendous boost on the performance, as evidenced from Section 4.1. For automatic graph construction, Monte Carlo based loop closure detection is designed that efficiently close the loop when building a pose graph.

By exploiting the unique geometrical structure of road boundary, localization based on curbs and intersections allowed vehicle to autonomously navigate through hilly part of the NUS engineering road. Synthetic LIDAR further expand the usage on allowing localization to be performed under different kinds of urban environment more accurately. The topo-metric graph mapping includes both topology and metric information of the map that is done using a ground vehicle. This allows automatic generation of the road topological from a map, which gives a compact description of the mapping region.

Activity mapping is shown to be effective with data collected from a moving vehicle. To obtain the semantic information of a map, activity and road network data are used. These extracted data can be used to aid autonomous navigation. We showed an example on how we can perform path planning using POMDP by knowing before hand a pedestrian crossing is presented in the map.

The list of publications resulted from this thesis's work is included in Appendix A.

## 7.2 Limitation and Future Work

**Map applications** Localization is the direct application to the mapped data. Thanks to the consistent map, localization in a known environment can be done precisely. For the moment we rely on the geometrical features of an environment

which works very well. Part of the future work is to include texture information available from the intensity value of a LIDAR. This can be further extended by combining RGB color pixels.

The reliance on odometry while building the rolling window can be a problem. Although most of the errors are compensated through loop closure, the failure to recognize a closed loop or errors in localization can occur with a very poor odometry estimates. This can be avoided through online calibration that always maintain accurate extrinsic calibration on all sensors mounted on a moving vehicle.

### **Long term mapping**

The mechanics of map updates can be further developed to always ensure an always up-to-date map. This should include all types of the map discussed in the thesis: static, topo-metric, activity and semantic map. One specific improvement, for example, is the map should learn about places where changes occur consistently. Parking lots for instance, changes throughout a day where an empty lot can become occupied and vice versa.

The topo-metric map can be expanded to include detailed path that automatically provides specific lane information that can inform an autonomous vehicle exactly where a nominal path is. Other semantic information, for example a traffic light equipped intersection, an upcoming tight bend should be used collectively to define an autonomous vehicle's navigation behavior.

# Bibliography

- [1] R. O'Toole, *Gridlock: Why We're Stuck in Traffic and What to Do About It*. Cato Institute, 2009. [Online]. Available: <http://books.google.com.sg/books?id=1I8Wuv7P13AC>
- [2] J. Hooper, "From darpa grand challenge 2004 darpa's debacle in the desert," <http://www.popsci.com/scitech/article/2004-06/darpa-grand-challenge-2004darpas-debacle-desert>, 2004.
- [3] S. Russell, "Darpa grand challenge winner: Stanley the robot!" <http://www.popularmechanics.com/technology/engineering/robots/2169012>, 2006.
- [4] J. Voelcker, "Autonomous vehicles complete darpa urban challenge," <http://spectrum.ieee.org/green-tech/advanced-cars/autonomous-vehicles-complete-darpa-urban-challenge>, 2007.
- [5] J. Markoff, "Google cars drive themselves, in traffic," <http://www.nytimes.com/2010/10/10/science/10google.html?pagewanted=all>, October 9, 2010.
- [6] A. Knapp, "Nevada passes law authorizing driverless cars," <http://www.forbes.com/sites/alexknapp/2011/06/22/nevada-passes-law-authorizing-driverless-cars/>, 2011.
- [7] M. T. Review, "Bmw cdc," <http://www.technologyreview.com/business/39410/?p1=BI>.
- [8] "2010 Autonomous Audi TTS Pikes Peak — car review @ Top Speed," <http://www.topspeed.com/cars/audi/2010-autonomous-audi-tts-pikes-peak-ar92542.html>, 2010.

- 
- [9] T. Vanderbilt, “Let the robot drive: The autonomous car of the future is here,” <http://www.nytimes.com/2010/10/10/science/10google.html?pagewanted=all>, January 20, 2012.
- [10] “NISSAN — Japan’s Prime Minister Rides in Nissan LEAF for first Autonomous Drive Public Road Test,” [http://www.nissan-global.com/EN/NEWS/2013/\\_STORY/131109-01-e.html](http://www.nissan-global.com/EN/NEWS/2013/_STORY/131109-01-e.html), 2013.
- [11] “UK to allow driverless cars on public roads in January,” <http://www.bbc.com/news/technology-28551069>, 2014.
- [12] W. J. Mitchell, C. E. Borroni-Bird, and L. D. Burns., *Reinventing the Automobile: Personal Urban Mobility for the 21st Century*. Cambridge, MA: The MIT Press, 2010.
- [13] C.-C. Chuang, “Taipei mobility : gone in less than 300 seconds : mixed modality transportation system in dense mixed-use urban fabric, take taipei city for example,” 2008. [Online]. Available: <http://hdl.handle.net/1721.1/45965>
- [14] S. A. Shaheen, S. Guzman, and H. Zhang, “Bikesharing in Europe, the Americas, and Asia,” *Transportation Research Record: Journal of the Transportation Research Board*, pp. 159–167, 2010.
- [15] M. Pavone, S. L. Smith, E. Frazzoli, and D. Rus, “Load Balancing for Mobility-on-Demand Systems,” in *Robotics: Science and Systems*, 2011.
- [16] G. Berbeglia, J.-F. Cordeau, and G. Laporte, “Dynamic pickup and delivery problems,” *European Journal of Operational Research*, vol. 202, no. 1, pp. 8–15, 2010.
- [17] M. Pavone, K. Treleaven, and E. Frazzoli, “Fundamental Performance Limits and Efficient Policies for Transportation-On-Demand Systems,” in *IEEE Conf. on Decision and Control*, 2010.

- 
- [18] T. Cowen, “Can i see your license, registration and c.p.u.?” <http://www.nytimes.com/2011/05/29/business/economy/29view.html>, May 28, 2011.
  - [19] D. Ferguson, T. Howard, and M. Likhachev, “Motion planning in urban environments: Part i,” in *Proceedings of the IEEE/RSJ 2008 International Conference on Intelligent Robots and Systems*, September 2008.
  - [20] —, “Motion planning in urban environments: Part ii,” in *Proceedings of the IEEE/RSJ 2008 International Conference on Intelligent Robots and Systems*, September 2008.
  - [21] Y. Kuwata, G. A. Fiore, J. Teo, E. Frazzoli, and J. P. How, “Motion planning for urban driving using RRT,” in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, Nice, France, September 2008, pp. 1681–1686. [Online]. Available: <http://acl.mit.edu/papers/KuwataIROS08.pdf>
  - [22] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How. Real-Time Motion Planning With Applications to Autonomous Urban Driving. <http://dspace.mit.edu/openaccess-disseminate/1721.1/52527>.
  - [23] Z. Chong, B. Qin, T. Bandyopadhyay, T. Wongpiromsarn, E. Rankin, M. Ang, E. Frazzoli, D. Rus, D. Hsu, and K. Low, “Autonomous personal vehicle for the first- and last-mile transportation services,” in *Cybernetics and Intelligent Systems (CIS), 2011 IEEE 5th International Conference on*, sept. 2011, pp. 253 –260.
  - [24] N. P. L. J. S. M. F. W. T. S. Bosse, M., “An atlas framework for scalable mapping,” vol. 2, 2003, pp. 1899–1906, cited By (since 1996) 84. [Online]. Available: <http://www.scopus.com/inward/record.url?eid=2-s2.0-0344014239&partnerID=40&md5=68b9dc4dd27e436cc90dd512406dcfc9>
  - [25] S. C. B. W. Grisetti, G., “Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling,” vol.



- 2005, 2005, pp. 2432–2437, cited By (since 1996) 114. [Online]. Available: <http://www.scopus.com/inward/record.url?eid=2-s2.0-33846122974&partnerID=40&md5=1a57f56f708a3ee244a8affce1fb1d5d>
- [26] R. Smith, M. Self, and P. Cheeseman, “Estimating uncertain spatial relationships in robotics,” in *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, vol. 4, mar 1987, p. 850.
- [27] E. R. L. J. Walter, M.R., “Exactly sparse extended information filters for feature-based slam,” *International Journal of Robotics Research*, vol. 26, no. 4, pp. 335–359, 2007, cited By (since 1996) 45. [Online]. Available: <http://www.scopus.com/inward/record.url?eid=2-s2.0-34047263802&partnerID=40&md5=9efb10b9b86769e9ff4207fa6a53981a>
- [28] K. Murphy, “Bayesian map learning in dynamic environments,” in *In Neural Info. Proc. Systems (NIPS)*. MIT Press, pp. 1015–1021.
- [29] G. Grisetti, C. Stachniss, and W. Burgard, “Improved techniques for grid mapping with rao-blackwellized particle filters,” *IEEE Transactions on Robotics*, vol. 23, p. 2007, 2007.
- [30] P. J. Besl and N. D. McKay, “A method for registration of 3-d shapes,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 14, no. 2, pp. 239–256, 1992.
- [31] B. Steder, G. Grisetti, and W. Burgard, “Robust place recognition for 3d range data based on point features,” in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 1400–1405.
- [32] M. Magnusson, H. Andreasson, A. Nuchter, and A. J. Lilienthal, “Appearance-based loop detection from 3d laser data using the normal distributions transform,” in *Robotics and Automation, 2009. ICRA’09. IEEE International Conference on*. IEEE, 2009, pp. 23–28.
- [33] D. F. Huber and M. Hebert, “Fully automatic registration of multiple 3d data sets,” *Image and Vision Computing*, vol. 21, no. 7, pp. 637–650, 2003.

- 
- [34] F. Lu and E. Milios, “Globally consistent range scan alignment for environment mapping,” *Autonomous robots*, vol. 4, no. 4, pp. 333–349, 1997.
  - [35] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, “A tutorial on graph-based slam,” *Intelligent Transportation Systems Magazine, IEEE*, vol. 2, no. 4, pp. 31–43, 2010.
  - [36] K. Granstrom, J. Callmer, F. Ramos, and J. Nieto, “Learning to detect loop closure from range data,” in *Robotics and Automation, 2009. ICRA’09. IEEE International Conference on*. IEEE, 2009, pp. 15–22.
  - [37] 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.
  - [38] U. Frese and L. Schroder, “Closing a million-landmarks loop,” in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. IEEE, 2006, pp. 5032–5039.
  - [39] G. Grisetti, C. Stachniss, and W. Burgard, “Non-linear constraint network optimization for efficient map learning,” *IEEE Transactions on Intelligent Transportation Systems*, p. 2009.
  - [40] M. Kaess, A. Ranganathan, and F. Dellaert, “isam: Incremental smoothing and mapping,” *Robotics, IEEE Transactions on*, vol. 24, no. 6, pp. 1365–1378, 2008.
  - [41] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert, “isam2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering,” in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 3281–3288.
  - [42] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, “g2o: A general framework for graph optimization,” in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, Shanghai, China, May 2011.

- 
- [43] N. Sunderhauf and P. Protzel, “Towards a robust back-end for pose graph slam,” in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 1254–1261.
  - [44] E. Olson and P. Agarwal, “Inference on networks of mixtures for robust robot mapping,” *Proceedings of Robotics: Science and Systems (RSS), Sydney, Australia*, 2012.
  - [45] Y. Latif, C. Cadena, and J. Neira, “Robust loop closing over time,” *Proceedings of Robotics: Science and Systems (RSS), Sydney, Australia*, 2012.
  - [46] N. Sünderhauf and P. Protzel, “Switchable constraints vs. max-mixture models vs. rrr—a comparison of three approaches to robust pose graph slam,” in *IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany, May 2013.
  - [47] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT Press, 2005.
  - [48] A. Mohamed and K. Schwarz, “Adaptive Kalman filtering for INS/GPS,” *Journal of Geodesy*, vol. 73, no. 4, pp. 193–203, 1999.
  - [49] 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.
  - [50] 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.
  - [51] T. Kos, I. Markezic, and J. Pokrajcic, “Effects of multipath reception on GPS positioning performance,” in *ELMAR, 2010 PROCEEDINGS*. IEEE, 2010, pp. 399–402.
  - [52] 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.

- 
- [53] 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.
- [54] M. Jabbour and P. Bonnifait, “Global localization robust to GPS outages using a vertical lidar,” *2006 9th International Conference on Control, Automation, Robotics and Vision, Vols 1- 5*, pp. 1013–1018, 2006.
- [55] 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.
- [56] M. Hentschel and B. Wagner, “Autonomous robot navigation based on open-streetmap geodata,” in *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, sept. 2010, pp. 1645 –1650.
- [57] O. Wulf, K. Arras, H. 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, 26-may 1, 2004, pp. 4204 – 4209 Vol.4.
- [58] 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.
- [59] J. Levinson and S. Thrun, “Robust vehicle localization in urban environments using probabilistic maps,” in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, may 2010, pp. 4372 –4378.
- [60] 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.
- [61] O. Wulf, D. Lecking, and B. Wagner, “Robust self-localization in industrial environments based on 3d ceiling structures,” in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. IEEE, 2006, pp. 1530–1534.

- 
- [62] D. Lecking, O. Wulf, and B. Wagner, “Localization in a wide range of industrial environments using relative 3d ceiling features,” in *Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on*. IEEE, 2008, pp. 333–337.
- [63] P. Checchin, F. Gérossier, C. Blanc, R. Chapuis, and L. Trassoudaine, “Radar scan matching slam using the fourier-mellin transform,” in *Field and Service Robotics*. Springer, 2010, pp. 151–161.
- [64] M. Aly, “Real time detection of lane markers in urban streets,” in *Intelligent Vehicles Symposium, 2008 IEEE*. IEEE, 2008, pp. 7–12.
- [65] 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.
- [66] 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.
- [67] C. Wang, S. Huang, and L. Fu, “Driver assistance system for lane detection and vehicle recognition with night vision,” in *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*. IEEE, 2005, pp. 3530–3535.
- [68] S. Kammel and B. Pitzer, “Lidar-based lane marker detection and mapping,” in *Intelligent Vehicles Symposium, 2008 IEEE*. IEEE, 2008, pp. 1137–1142.
- [69] P. Lindner, E. Richter, G. Wanielik, K. Takagi, and A. Isogai, “Multi-channel lidar processing for lane detection and estimation,” in *Intelligent Transportation Systems, 2009. ITSC’09. 12th International IEEE Conference on*. IEEE, 2009, pp. 1–6.
- [70] S. Thrun, M. Montemerlo, and A. Aron, “Probabilistic terrain analysis for high-speed desert driving,” *Proc. Robotics Science and Systems, Philadelphia, PA, USA*, 2006.

- 
- [71] W. Zhang, “LIDAR-based road and road-edge detection,” in *Intelligent Vehicles Symposium (IV)*, 2010 IEEE, pp. 845–848.
  - [72] 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.
  - [73] W. S. Wijesoma, K. R. S. Kodagoda, and A. P. Balasuriya, “Road-boundary detection and tracking using ladar sensing,” *IEEE Transactions on Robotics and Automation*, vol. 20, no. 3, pp. 456–464, 2004.
  - [74] 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.
  - [75] O. Mozoš, 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*.
  - [76] O. Martinez Mozoš, A. Rottmann, R. Triebel, P. Jensfelt, W. Burgard *et al.*, “Semantic labeling of places using information extracted from laser and vision sensor data,” 2006.
  - [77] 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.
  - [78] 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.
  - [79] A. Nüchter and J. Hertzberg, “Towards semantic maps for mobile robots,” *Robotics and Autonomous Systems*, 2008.
  - [80] 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.
- [81] 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.
- [82] D. F. Wolf and G. S. Sukhatme, “Semantic mapping using mobile robots,” *Robotics, IEEE Transactions on*, vol. 24, no. 2, pp. 245–258, 2008.
- [83] 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.
- [84] B. Qin, Z. J. Chong, T. Bandyopadhyay, M. Ang, E. Frazzoli, and D. Rus, “Curb-intersection feature based monte carlo localization on urban roads,” in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, May 2012, pp. 2640–2646.
- [85] 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.
- [86] 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.
- [87] 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.
- [88] R. Rusu, “Semantic 3d object maps for everyday manipulation in human living environments,” *KI-Künstliche Intelligenz*, vol. 24, no. 4, pp. 345–348, 2010.
- [89] 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.
- [90] 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.
  - [91] E. B. Olson, “Real-time correlative scan matching,” in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 4387–4393.
  - [92] G. Rong and T.-S. Tan, “Jump flooding in gpu with applications to voronoi diagram and distance transform,” in *Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games*, ser. I3D '06. New York, NY, USA: ACM, 2006, pp. 109–116. [Online]. Available: <http://doi.acm.org/10.1145/1111411.1111431>
  - [93] A. Okabe, B. Boots, and K. Sugihara, *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. New York, NY, USA: John Wiley & Sons, Inc., 1992.
  - [94] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, “Monte carlo localization for mobile robots,” in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 2. IEEE, 1999, pp. 1322–1328.
  - [95] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, “Monte carlo localization: Efficient position estimation for mobile robots,” in *Proceedings of the National Conference on Artificial Intelligence*. JOHN WILEY & SONS LTD, 1999, pp. 343–349.
  - [96] 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.
  - [97] Y. Liu and H. Zhang, “Visual loop closure detection with a compact image descriptor.”



- 
- [98] B. Kim, M. Kaess, L. Fletcher, J. Leonard, A. Bachrach, N. Roy, and S. Teller, “Multiple relative pose graphs for robust cooperative mapping,” in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, May 2010, pp. 3185–3192.
- [99] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, “OctoMap: An efficient probabilistic 3D mapping framework based on octrees,” *Autonomous Robots*, 2013, software available at <http://octomap.github.com>. [Online]. Available: <http://octomap.github.com>
- [100] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.
- [101] 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.
- [102] 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.
- [103] Z. Guo and R. W. Hall, “Parallel thinning with two-subiteration algorithms,” *Commun. ACM*, vol. 32, no. 3, pp. 359–373, Mar. 1989. [Online]. Available: <http://doi.acm.org/10.1145/62065.62074>
- [104] C. Hasberg and S. Hensel, “Online-estimation of road map elements using spline curves,” in *Information Fusion, 2008 11th International Conference on*, June 2008, pp. 1–7.
- [105] 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.

- 
- [106] 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.
  - [107] 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.
  - [108] C. E. Rasmussen, “Gaussian processes for machine learning,” 2006.
  - [109] K. V. Mardia and P. E. Jupp, *Directional statistics*. Wiley. com, 2009, vol. 494.
  - [110] R. Kindermann, J. L. Snell *et al.*, *Markov random fields and their applications*. American Mathematical Society Providence, RI, 1980, vol. 1.
  - [111] T. Bandyopadhyay, Z. J. Chong, D. Hsu, M. H. A. Jr., D. Rus, and E. Frazzoli, “Intention-aware pedestrian avoidance.” in *ISER*, 2012, pp. 963–977.
  - [112] D. Helbing, L. Buzna, A. Johansson, and T. Werner, “Self-organized pedestrian crowd dynamics: Experiments, simulations, and design solutions,” *Transportation Science*, vol. 39, no. 1, pp. 1–24, Feb. 2005. [Online]. Available: <http://dx.doi.org/10.1287/trsc.1040.0108>
  - [113] H. Kurniawati, D. Hsu, and W. S. Lee, “Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces,” in *In Proc. Robotics: Science and Systems*, 2008.
  - [114] “Kairos Autonomi: Universal Agnostic Autonomy Systems for Existing Vehicles and Vessels,” <http://www.kairosautonomi.com/>, 2014.
  - [115] 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, submitted.”
  - [116] Z. Chong, B. Qin, T. Bandyopadhyay, M. Ang, E. Frazzoli, and D. Rus, “Mapping with synthetic 2d lidar in 3d urban environment,” in *Intelligent*

*Robots and Systems (IROS)*, 2013 IEEE/RSJ International Conference on, Nov 2013, pp. 4715–4720.

- [117] S. Karaman and E. Frazzoli, “Sampling-based algorithms for optimal motion planning,” *Int. J. Rob. Res.*, vol. 30, no. 7, pp. 846–894, Jun. 2011. [Online]. Available: <http://dx.doi.org/10.1177/0278364911406761>
- [118] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, “Ros: an open-source robot operating system,” *ICRA workshop on open source software*, vol. 3, no. 3.2, p. 5, 2009.

# Appendices

# Appendix A

## Vehicle Platform

Our autonomous vehicle fleet consist of Rudolph, Driverless Jockey (DJ), and Shared Computer-Operated Transport (SCOT). To evaluate different scenario in urban environment, different classes of vehicles are used. Rudolph and DJ can traverse alongside pedestrian with its relative low speed, making it suitable for use in indoor pedestrian environments. This type of vehicles have been used in airports to help passengers who needed an urgent transfer to make connection flights. It is also being used in amusement parks, where it is used to transport visitors to different areas of a park. On the other hand, a roadworthy car is suitable for use to travel road networks at a higher speed, covering a larger area in shorter time.

### A.1 Golf Carts

The instrumented golf carts, Rudolph (Fig. A.1) and DJ (Fig. A.2) are based on a Yamaha G-Max 48 Volt Golf Car G22E. It has a seating capacity of 2 persons with maximum forward speed of 24 km/h. To enable drive by wire for computer control, a servo is attached to the golf cart's steering column. Similarly, a motor is fitted near the brake pedal to actuate the brake mechanically. On the other hand, the throttle signal is voltage regulated that completes the control of the vehicle's speed and directional controls.

Both rear wheels of the golf carts are mounted with encoders that provide an estimate of the distance traveled. This is evident on Rudolph, where the encoders are mounted externally. It is later redesigned to place the encoders directly to



Figure A.1: First instrumented golf cart - Rudolph



Figure A.2: Second generation golf cart - DJ



Figure A.3: SCOT

the wheel shaft, resulting a cleaner look. An Inertial Measurement Unit (IMU) is mounted at the center of the rear axle to provide attitude and heading of the vehicle.

For external sensing, 2 LIDARs are mounted at the frontal part of the golf cart at different heights. The LIDARs, a Sick LMS151 is a single plane LIDAR with 50 *m* range measurements with a 270 degree field of view. Both LIDARs are connected using on-board Local Area Networks, which enable high speed connection to the LIDARs. The LIDAR is able to provide measurements at 0.5 degree of resolution running at 50 Hz. The top LIDAR is mounted in push broom configuration and the second LIDAR is mounted looking forward horizontally.

There are 2 regular desktop PCs fitted with Intel i7 quad-core CPUs and interface cards, along with supporting electronic circuits, this includes DC-DC converters and microcontrollers.

## A.2 SCOT

SCOT (Fig. A.3) is based on an electric car (the Mitsubishi's iMiev). The iMiev is a five door hatchback electric car. The car is fitted with a strap on kit from Kairos Autonomi [114] to allow computer control on steering, brake, throttle and gears.

The sensor configuration is similar to a golf cart. 2 LIDARs are mounted at the



car's front. Similarly, an IMU is positioned at the center of the rear axles, mounted underneath the back seat with 2 mini-ITX PCs located at the trunk space of the car. There are no external encoder mounted, as the same information of distance traveled is available through the CAN-bus of iMiev.

## A.3 Coordinate Systems

East-North-Up (*ENU*) coordinate system is adopted on all vehicles. Its origin and axes are given as follows:

1. The origin is located at the center of rear axle attached to the ground
2. The X-axis points forward, along the symmetry plane of the vehicle
3. The Y-axis points to the left side of the vehicle
4. The Z-axis completes the right-hand rule points upwards from the ground

## A.4 Euler Angles

A relative orientation between any two Cartesian frames can be described by Euler Angles. In this thesis, Z-Y-X rotation sequence is adopted to move a child frame to a parent frame. These three rotations are known as yaw, pitch and roll angles, which is defined as the following:

1. Yaw Angle, denoted by  $\psi$ , is the right-handed rotation about the Z-axis. It is the projection of the X-axis from one frame to another on the X-Y plane. It is sometime referred to as the heading of the vehicle.
2. Pitch Angle, denoted by  $\theta$ , is the projection of the Y-axis to the referred frame on the Z-X plane. It is the right-handed rotation about the Y-axis.
3. Roll Angle, denoted by  $\phi$ , is the right-handed rotation about the X-axis. It is the projection of the Z-axis on the Y-Z plane.



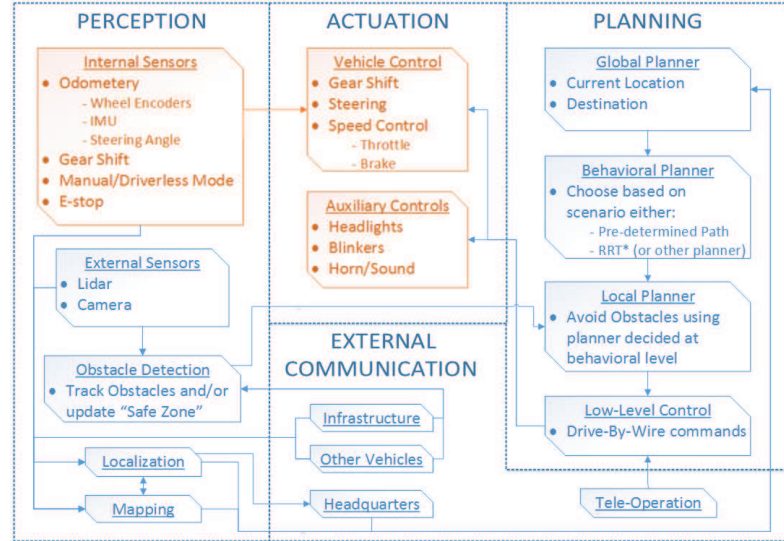


Figure A.4: System architecture for our autonomous vehicles. Only the orange section may vary for each vehicle type.

## A.5 System Architecture

The system architecture is common to all vehicles in the fleet, shown in Fig. A.4. Note that only some portion of the internal sensors and vehicle controls (shown in orange) may vary between vehicles, thus those portions of the software is unique to different type of vehicles. The external sensors and bulk of the software would be maintained; this includes all high-level algorithms, such sensor data fusion and localization [115], mapping [116], and motion planning with RRT\* [117] in our case. The Robot Operating System (ROS) is employed to standardize communication across modules [118].

## A.6 Odometry

Odometry gives an estimate on the state of the vehicle. In the platform, dead reckoning derived from the encoders and IMU is used. Using a unicycle model, the 2D state estimation is given by

$$x_{k+1} = x_k + d \cos \Psi$$

$$y_{k+1} = y_k + d \sin \Psi$$

$$\psi_{k+1} = \psi_k + \Psi$$

Where  $d$  is the distance travel and  $\Psi$  is the difference in yaw angle. Usually, this 2D information is enough to handle state estimation along a relatively flat ground. However, it is not sufficient when traveling on a hilly surface. In this case, another measurement  $\theta$  is included to perform estimation in 3D. As such, equation above can be expanded into pseudo 3D state estimation, given by

$$x_{k+1} = x_k + d \cos \Psi \cos \theta$$

$$y_{k+1} = y_k + d \sin \Psi \sin \theta$$

$$z_{k+1} = z - d \sin \theta$$

$$\psi_{k+1} = \psi_k + \Psi$$

where  $\theta$  is the pitch angle as measured by the IMU. The estimation, although accurate within a short distance, the error on the estimation increases as the vehicle traveling in longer distance. This can be attributed by many factors, for example unequal wheel diameters, tire slippage and the error increments of the yaw estimate from IMU.

# Appendix B

## Author's Publications

1. 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.
2. 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.
3. 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.
4. **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.
5. 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.
6. **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
7. 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 Mechantronics (CIS & RAM), 2013
  8. T. Bandyopadhyay, **Z. J. Chong**, D. Hsu, M. H. Ang Jr., D. Rus, E. Frazzoli, "Intention-Aware Pedestrian Avoidance," in International Symposium on Experimental Robotics (ISER), 2012.
  9. 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
  10. **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
  11. **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
  12. 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
  13. **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

- 
14. 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.