MapLite: Autonomous Navigation in Rural Environments without Detailed Prior Maps

by

Teddy Ort

Submitted to the Department of
Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2020

Signature redacted

Author .....
Department of Electrical Engineering and Computer Science
Jan 30, 2020

Signature redacted

Certified by .....
Daniela Rus
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Signature redacted

Accepted by .....
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

MapLite: Autonomous Navigation in Rural Environments without Detailed Prior Maps

by

Teddy Ort

Submitted to the Department of Electrical Engineering and Computer Science
on Jan 30, 2020, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science and Engineering

Abstract

Most autonomous vehicle systems currently rely on high-definition prior maps in order to localize and navigate in their environment. In this work, we present MapLite: a one-click autonomous navigation system capable of piloting a vehicle to an arbitrary desired destination point given only a sparse publicly available topometric map (from OpenStreetMap). The onboard sensors are used to segment the road region and register the topometric map in order to fuse the high-level navigation goals with a variational path planner in the vehicle frame. This enables the system to plan trajectories that correctly navigate road intersections without the use of an external localization system such as GPS or a detailed prior map. Since the topometric maps already exist for the vast majority of roads, this solution greatly increases the geographical scope for autonomous mobility solutions. We implement MapLite on a full-scale autonomous vehicle and exhaustively test it on over 15km of real-world driving including over 100 autonomous intersection traversals. We further extend these results through simulated testing to validate the system on complex road junction topologies such as traffic circles.

Thesis Supervisor: Daniela Rus
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

I would like to express my heartfelt gratitude to the many people who have helped make this project possible. First, I want to thank my mother and all of my siblings for always supporting me along my journey to reach this goal. Whether it is sending care packages when I'm sick, life advice when I'm confused, or encouraging words when I'm feeling down, I'm so lucky to have all of you in my back pocket. I'm so grateful to my girlfriend Emily for always being there for me even when, at times, this project left little room for anything else. Your love, constant encouragement, and boundless optimism is wonderful and it's more than I hoped for.

Next, I would like to thank my Advisor, Professor Daniela Rus, for all of her help, support, and guidance throughout this work. From choosing the right problems to effectively communicating the results, she has taught me so much about this field in the last few years. I certainly look forward to keeping the gradient!

I can't express how grateful I am to Professor Liam Paull, who originally conceived of this project, served as a continuous source of advice and inspiration throughout, and without whom none of this could have succeeded.

Many thanks to my other coauthors, Igor Gilitschenski, Rohan Banerjee, Krishna Murthy, Sai Krishna Gottipati, and Dhaivat Bhatt for all of their assistance both with bringing this project to fruition and writing the associated papers. I'd also like to acknowledge the Toyota Research Institute (TRI)[1] for their generous support of this work.

Finally, I want to thank all of my friends at the Distributed Robotics Lab and throughout MIT for welcoming me and always making me feel at home, first as a transfer student and then as a graduate student. There are so many reasons I feel lucky to come to lab every day, but the biggest one is all of you. I Have Truly Found Paradise.

[1]Toyota Research Institute (TRI) provided funds to assist the author with his research. However, this thesis solely reflects the opinions and conclusions of the author and not TRI or any other Toyota entity.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Autonomous driving technology has recently achieved rapid advances leading to large increases in investment, expanded research interest, and dozens of fielded trials by private industry and institutions around the world. The potential benefits if these goals are brought to fruition are indeed substantial. The United States Department of Transportation reports over 30,000 motor vehicle fatalities each year [35] with millions more injured [34]. More than 90% of fatal accident were attributable to human errors such as impaired or distracted driving. The Toyota-CSAIL Joint Research Institute [18] - through which this work was partially funded - aims to reduce or even eliminate traffic fatalities by developing a vehicle that is incapable of causing a traffic accident. Furthermore, autonomous vehicles may also offer improvements in a range of areas including environmental benefits through efficient vehicle allocation and ridesharing, increased mobility for people with disabilities, reduction in traffic congestion, and many others.

Those who live in rural areas stand to derive the most benefit from autonomous mobility systems. While the typical city offers a myriad transportation options including buses, subways, taxis, and rideshare services, these options are not typically available to the rural population. For the 60 million Americans who live in these areas, the only transportation option is often owning and operating a personal vehicle.

This can be a challenge for people struggling financially, the elderly, or anyone with a disability that prevents them from safely driving a car. However, the vast majority of autonomous driving pilots currently fielded have taken place in densely populated urban environments. Autonomous navigation in rural environments is much more challenging because the unstructured nature of these environments make building the required *high definition* 3D maps much more difficult. Furthermore, the low population density in these areas makes it difficult to justify the cost of maintaining these maps and the lack of fixed urban structures also makes these maps less reliable.

Currently, most autonomous navigation approaches focus on one of two methods:

1. Building and localizing in detailed "High Definition" maps

2. Lane following relying on lane markings or road boundaries

The first approach works well for trials in small urban areas. However, building detailed maps is expensive and time consuming. This approach does not scale well to vast rural areas where the traffic density is low and cost of creating the maps becomes prohibitive. Furthermore, maintaining the maps in an unstructured rural environment is more difficult because these environments are typically characterized by trees and vegetation, which change frequently and lack the fixed structures such as building and infrastructure that make urban areas more amenable to high definition map creation.

The second approach is appealing because it doesn't require building and maintaining any maps. However, many roads don't contain the required road markings and, in fact, only 2/3 of the roads in the United States are even paved [7]. Moreover, lane following approaches typically only allow for following roads without junctions (i.e. freeways). This does not enable decision making at intersections, which is necessary for autonomous navigation from the start point all the way to the destination. For these reasons while current approaches have demonstrated promising results in small urban field tests, they do not easily scale to significant portions of the road network.

1.2 Method Overview

In this work, we present MapLite, a "one-click" autonomous navigation system capable of piloting a vehicle to an arbitrary desired destination point given only a sparse publicly available topometric map. The main benefit of this approach is that it doesn't require the burdensome task of creating high definition maps. Furthermore, we also don't rely on lane or road infrastructure which allows the system to work even on unmarked rural roads. This greatly expands the size of the areas in which point-to-point autonomous navigation could be deployed and the population it could reach.

Our proposed system has three main steps: First, we segment the scene using onboard sensors to detect the road ahead. The input is a 3D pointcloud and the segmentation step produces a label for each point to indicate whether it lies on or off the road. Next, we perform topometric registration to best fit the OpenStreetMap (OSM) to the observed scene. Here, the input is the topometric map from OSM and the registration step deforms the input map to better match the output from the road segmentation without changing the topology of the map. Finally, we solve the navigation problem at the topological level and the path planning problem at the metric level to generate a local goal path. This entails first solving the shortest-path graph search problem on the topometric map using A^* and then using the resulting path as a reference for a variational planner that applies safety constraints such as remaining on the road. The result is a reference path that incorporates both the high-level navigation information and the local planning constraints. Finally, this path is followed using a pure-pursuit controller that generates the correct steering angles to follow the reference path toward the goal. A dynamic speed controller governs the speed of the vehicle to ensure it doesn't violate maximum speed or acceleration constraints.

The resulting system can take as input a lightweight topometric map and a requested goal point from a passenger, and autonomously drive the car to the destination. To the best of our knowledge, this is the first deployment of an autonomous

navigation pipeline capable of autonomously planning and executing navigation tasks between arbitrary points without requiring a precise geo-referenced localization estimate either from high-precision Global Positioning System (GPS) or high definition detailed prior maps.

MapLite includes the following technical contributions:

1. A framework for autonomous driving that relies on a topometric rather than a detailed prior map;

2. A novel topometric map registration algorithm that approximates the maximum a posteriori estimate of the registration between the vehicle and the map

3. A route planning method we call "smart replanning" that allows for efficiently solving the topological navigation problem on a frequently updating map

4. Extensive evaluations on a full-scale autonomous car in a rural driving setting where the roads lack structure and through simulations in CARLA [21] on more complex road topologies such as traffic circles

This system could be utilized across the spectrum of autonomous mobility applications. Firstly, it could provide a fully autonomous vehicle for mobility-as-a-service in rural areas to those who need it most. Additionally, in places where *high definition* 3D maps are available, this system could still serve as a robust fallback mechanism to guide an autonomous vehicle to safety in the event that it loses localization in the HD map. Finally, even in manually driven vehicles, advanced driver-assistance systems (ADAS), would particularly benefit from the ability to function in areas without HD maps because they cannot be easily geo-fenced the way fully autonomous systems can.

1.3 Thesis Outline

The remainder of this thesis is organized as follows: In Chap. 2 we discuss the related work and similar approaches. In Chap. 3 we detail the MapLite system including a

base case for single edge traversal we first described in [37] as well as the complete system [38] which allows for point-to-point navigation. Next, in Chap. 4 we describe our real-world experiments on a full-scale autonomous Toyota Prius and report the evaluation results. Finally, in Chap. 5 we end with some concluding remarks and lessons learned. For a complete listing of all the sections, please see the Table of Contents (page 8).

Chapter 2

Related Work

This work is inspired by previous work in the areas of *hybrid metric-topological mapping*, as well as *local perception and localization for autonomous driving*. In this chapter, we will highlight some of the most closely related works. We discuss the more common approach using a detailed prior map in Sec. 2.1, Map-less systems in Sec. 2.2, Hybrid Metric/Topological methods in Sec. 2.3, and we end with a discussion on the rich literature relating to road-segmentation - a key component of the MapLite system - in Sec. 2.4.

2.1 Map-based Autonomous Driving

Most of the existing approaches to autonomous driving rely on detailed prior maps of the environment, often constructed using precision laser scanners. Typical driving strategies on pre-built maps involve some form of localization followed by local trajectory and motion planning. For example, [29] uses a LiDAR sensor to create maps of the road for autonomous navigation. In an extension of that work, [28] change the map from directly storing the road reflectance, to a probability distribution for greater robustness. Both these approaches rely heavily on the strong reflectance disparity of painted lane markings and thus are only suitable in urban areas where these markings are consistently found. To overcome this [49] create Gaussian mixture maps over the road geometry for later localization. Similar to our approach, this method

doesn't rely on lane markings and is suitable for rural environments. However, it still requires manually driving over the road for the map construction, which can be time consuming. Other methods include vision-based localization [48], which benefits from the low cost of cameras compared with LiDAR sensors. However, these approaches often suffer from ambient lighting changes since cameras can be blinded by bright lights or darkness.

Evidently, all such approaches need extremely precise prior maps, and cannot easily adapt to environment changes (e.g., repaved roads, detours, etc.) [44]. For autonomous mobility-on-demand systems that are severely restricted in their operational area, maintaining a highly accurate globally referenced metric map may be possible, but this approach does not scale well either temporally or spatially.

2.2 Map-less Autonomous Driving

A different approach, more closely related to what we are proposing here, aims to overcome some of the scalability issues with map-based systems by perceiving the local environment for autonomous vehicle navigation. There have been several attempts to go *mapless* (at least in part). More than two decades ago, [10] presented an early example of such a method using a model-based algorithm to control a vehicle with just a camera on clearly-marked empty roads. More recently, [50] solved some of these issues with a camera-based road detection system capable of piloting a vehicle even on roads with obstacles or missing or occluded lane markings. [1] made progress on the issue of ambient lighting changes and shadows interrupting road detection by projecting imagery into a shadowless feature space. Here we utilize a LiDAR sensor, which actively illuminates the scene with invisible laser radiation to completely avoid issues with ambient lighting. [30, 2] use a particle filter and RANSAC model respectively to detect lane markings in front facing camera imagery. To deal with unmarked roads, [45] detect the road boundaries in camera images using a Bayes Filter.

While all these approaches rely on color/grayscale images to detect structures on

the road, these suffer the common pitfalls of any image-based segmentation algorithm: poor generalization across varying illumination, scene dynamics, and infrastructural changes. Moreover, most such approaches rely heavily on road markings [30, 2, 31], the very predictable geometry of the road curbs [11, 40], or both [27]. Furthermore, all such approaches only allow for navigation on road segments without junctions or intersections where the lane markings/boundaries reliably determine the reference trajectory. Here, we utilize a road detection algorithm that makes minimal assumptions on the road structure (only that the road is relatively flat), along with a topometric prior map, to enable autonomous navigation on unmarked rural roads even through road intersections to reach a desired goal point.

Another popular set of approaches take an *end-to-end* learning approach to mapless driving. [8] use imitation learning to train a neural network to imitate human driving steering angles from front facing camera imagery. An extension of this work [16] introduced a high-level planning cue to the network to enable traversal through intersections. Similarly, [3] use the imitation learning network to predict a probability distribution of steering angles to allow for integration of high-level planning. While such approaches alleviate the need for precision maps, they are still laborious to implement, as they require enormous amounts of data/demonstrations from a (human) expert. Here we focus on allowing truly one-click autonomous navigation using already constructed and freely available lightweight maps.

2.3 Hybrid Metric / Topological Mapping

The notion of hybrid metric/topological estimation has been an active field of research in the simultaneous localization and mapping (SLAM) community for two decades or more. For example, seminal works such as [15] and [25] laid the foundation in this field. The fundamental idea here is to use *local* perception for small-scale localization and topological perception for global localization and navigation. In some sense, these works are faced with a larger problem than what we are attempting to tackle because they attempt to simultaneously build and localize within both the small-scale and

the topological maps in an online fashion. In this work, we exploit the fact that, in the driving context, there already exists a curated and open topological map that is actually much more expressive than the topological representations used in these works (such as the Generalized Voronoi Graph) since the edges in OSM [36] are also labeled with important semantic information.

In a series of works, Ballardini et al. [4, 5, 6] leverage the OSM for localization and intersection detection. In a similar vein, OpenStreetSLAM [22] aligns OSM data to local maps as an additional cue in a Monte-Carlo localization framework. However, all these approaches demonstrate the use of OSM in urban areas, where the precision of OSM is significantly higher than for rural areas. In particular, our approach can efficiently deal with missing roads, incorrect intersection annotations, and large deviations in road shapes.

Many other related works in the SLAM field embrace a topological representation as well. [43] presents an algorithm for building a topological environment representation from sensor data. Conversely, [9] demonstrates localization in very large dense maps by associating each submap with an edge in the topological graph and focus on solving the relative transformation between the sub-graphs. Here however, we focus not on building the topological maps, or using them to augment a SLAM method, but instead as the sole means of specifying and executing the high-level navigation goal while using only onboard sensors for the local planning problem.

We embrace a fully relative representation of the local map. In other words, the frame used for small-scale localization is always attached to the robot (and is updated each time the robot moves). Others have embraced this fully relative view [46] for SLAM by parameterizing the robot trajectory in continuous time.

In this work, our goal is not to generate maps (in relative or global frame) but rather to generate feasible trajectories within the map. By explicitly connecting the local relative perception problem with the task of navigating through the environment at that instant, we can derive a much more lightweight requirement for the prior map.

2.4 Road Segmentation

In contrast to mapped approaches, where the drivable road surface is assumed available from the map, real-time road surface segmentation is a crucial requirement for mapless navigation. The basic problem of road segmentation is simple: for each element in the incoming sensor data, label it as either *road* or *off-road*. In Fig. 3-6 an example pointcloud has been segmented tow show the points lying on the road surface in black and the off-road points in blue. This problem has been studied extensively in the literature including model-based approaches such as [1] and those that rely on camera images [50, 51], LiDAR [12], and combined vision and LiDAR [14, 13] imposes some unique constraints. First, since a major benefit of our independence from prior maps is the ability to operate in large-scale rural regions where roads are missing structures such as lane markings or curbs, we abstain from utilizing these features in our approach. Furthermore, deployment on a real-world vehicle necessitates fast processing time. Finally, rural roads are often unlit, and have highly varying illumination even during the daytime due to seasonal vegetation. Therefore, our method relies solely on LiDAR since these sensors are immune to changes in ambient lighting.

Here, we implement and compare three different choices for our road segmentation module including a linear Support Vector Machine (SVM) and two convolutional networks: SparseConvNet (SCN) [23] and PointNet [42]. In Sec. 3.4, we detail each of these methods and compare their implementations and relative performance metrics.

Chapter 3

MapLite System

3.1 Framework Overview

An overview of our proposed system is shown in Fig. 3-1. Note that our approach requires only a topometric map, which does not contain the detail necessary for precise localization. However, we demonstrate that it suffices for specifying the goal location and onboard sensors can be used to plan safe trajectories on the fly. In the next section, we describe a base implementation for traversing a single edge directly from the sensor input. The following sections describe the five main components - Topometric Map, Road Segmentation, Map Registration, Route Planner, and Motion Planner - that enable this system to safely navigate to any desired location (including intersection traversals) without the need for a detailed prior map.

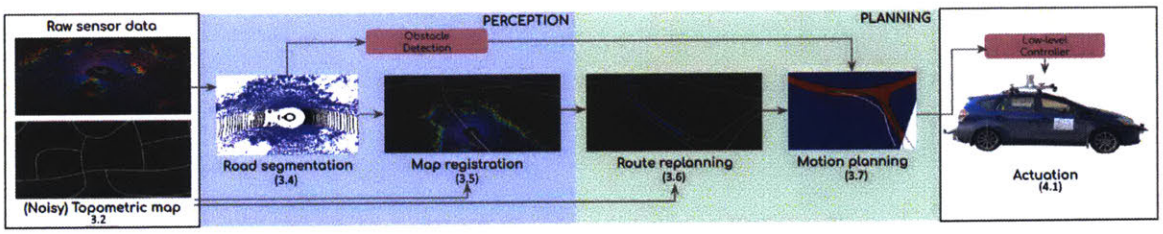

Figure 3-1: The MapLite system differs from a typical autonomous navigation pipeline in five components: Topological Map, Road Segmentation, Map Registration, Route Replanning, and Motion Planning

3.2 Topometric Maps

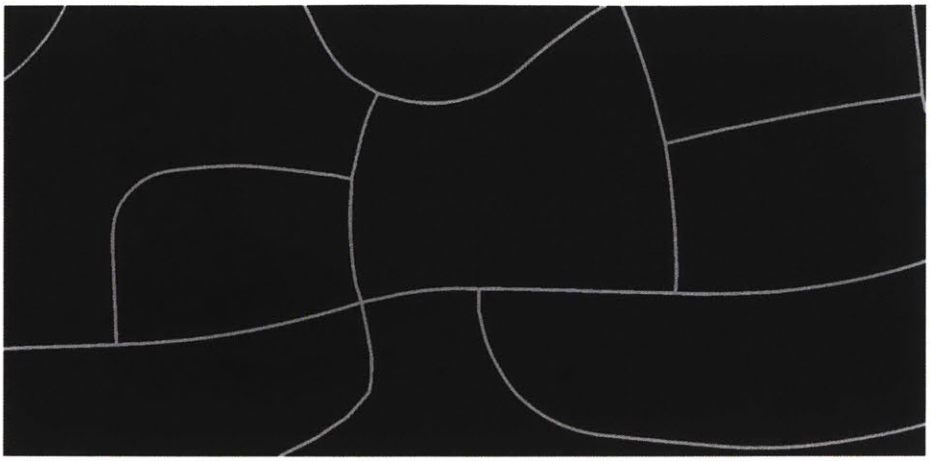

Figure 3-2: The topological maps used in MapLite are sparse structures containing the positions and connectivity of the road network. While we assume they are topologically accurate, they are metrically noisy and the edges may not always directly overlay the actual road center lines.

The topometric map plays a key role in the MapLite system. Here, we briefly describe its structure and how it differs from the high definition maps typically used. The topometric maps are simple graph-like data structures with the extension that each node is associated with a two dimensional point on the surface of the earth described by its longitude and latitude. We utilize the flat-earth approximation with the UTM transform [26], which places the vertices in a plane. Thus for each map M we have

$$M = \{V, E\}$$

where each vertex $v_i \in \mathbb{R}^2$ describes a waypoint and each edge $e_i \in E \subset |V| \times |V|$ describes a road segment. However, while the connectivity of the network can be assumed to be relatively correct, the same cannot be said for either the relative or global accuracy of the specific waypoints.

Topometric maps of the roads throughout the United States and much of the world are readily available from a number of sources. For this work, we used maps downloaded from OpenStreetMap.org[36] which provides open access for the public to both download and upload mapping data. Fig. 3-2 shows an example of the

topometric map used.

These topometric maps differ from the detailed maps typically used for localization in a number of important ways. First, detailed maps are usually recorded with a human driver prior to deployment and often require further labor-intensive manual editing to ensure the maps are accurate [29]. For this reason, detailed maps are currently available for only a handful of cities and even those are often only useful for a particular autonomous vehicle sensor setup. Topometric maps, on the other hand, are already freely available for most roads throughout the world.

Another difference between topometric maps and detailed maps lies in the storage size. While a detailed map used to localize over the 20,000 miles of roads in a small city takes about 200GB [29], a similar topometric map could be stored in about 3.5GB. Considering the US alone contains over 4,000,000 miles of roadways, this can have a large impact on the storage and data transmission required for autonomous navigation.

Finally, since detailed maps include many transient surface features, any changes to the road surface, seasonal vegetation, or building construction can render a detailed map obsolete often in just a few months. Topometric maps, on the other hand, only need updating when the road network itself changes which means these maps are typically accurate for many years.

3.3 Base Case: Single Edge Traversal

As a base case for the framework shown in Fig. 3-1, we first designed and implemented an edge traversal component that was used to navigate autonomously on rural road segments. In this application, the entirety of the PERCEPTION and PLANNING blocks in Fig. 3-1 is replaced with a single component that maps directly from sensor input and OSM to the low-level controllers. We achieve this by detecting the road boundaries and using them to directly estimate the necessary control commands as described shortly. Note that while this method is considerably simpler than the full system as described in the following sections, it is limited in that it can only be

used to follow the road edges of a single segment without intersections. However, this method served as a useful base case for demonstrating the overall feasibility of navigating without a detailed map and for proving the rest of the autonomy pipeline. For the full point-to-point navigation task however, the complete pipeline shown in Fig. 3-1 and described in sections 3.4-3.7 is required. An overview of the base case procedure is given in Algorithm 1.

Algorithm 1 Edge Traversal Algorithm: Local Road Perception, Tracking and Following

Inputs:

- next global waypoint
- stream of local road detection measurements
- stream of odometry measurements

while Current state not equal to global waypoint **do**
  while New sensor input not ready **do**
    Update reference trajectory estimate $\mathbf{f}_{k+1}(\alpha, \bar{\Theta}_{k+1})$ where $\bar{\Theta}_{t+1} = \mathbf{T}^{-1}\Theta_t$
    Generate control commands with trajectory following controller
  end while
  Generate a new reference trajectory based on the current sensor input $\mathbf{f}_{k+1}(\alpha, \hat{\Theta}_{k+1})$
  Probabilistically fuse reference trajectory estimates
end while

The sensor used for planning the edge traversal in the local frame is a Velodyne HDL-64 laser scanner. Each revolution of the sensor generates a pointcloud. An algorithm was developed to obtain the road boundary points in the sensor swath of the vehicle. These points were then fit using a RANSAC/least-squares approach to obtain an optimal trajectory within the road boundary points. Importantly, the quality of the fit was also obtained from the residuals output from the least-squares minimization. This allows the trajectory estimates to be probabilistically fused during the next iteration. The vehicle is also equipped with odometry sensors used to propagate the previous trajectory estimates to the local vehicle frame after the vehicle has moved. Thus, the reference trajectory is always in the vehicle local frame. Since the odometry

is typically updated much faster than the sensor measurements arrive (in this case 100Hz), the vehicle can continue to follow the reference trajectory for some time even while no new measurements have been received. Finally, the reference trajectory is used to steer the vehicle towards the local goal for edge traversal.

3.3.1 Preliminaries

As is common in autonomous vehicle navigation, we represent the reference trajectory as a clothoid [20].

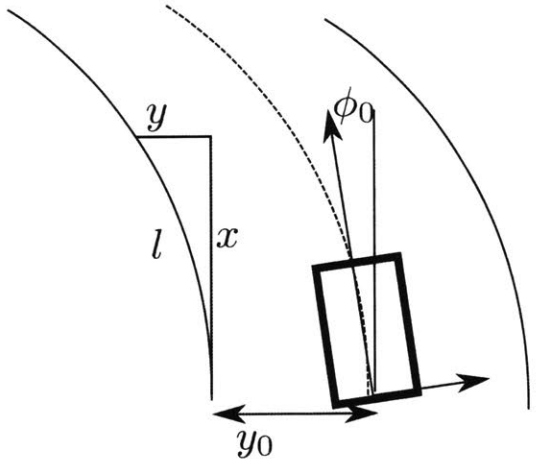

Figure 3-3: Clothoid geometry

Specifically, we choose to model the road as a clothoid of degree one whereby the curvature of the road varies linearly. We can recover the heading direction along the path by integrating the curvature along the path:

$$\begin{aligned} \phi(l) &= \phi_0 + \int_0^l C(t)dt \\ &= \phi_0 + lC^0 + \frac{l^2}{2}C^1 \end{aligned} \tag{3.1}$$

Referring to Fig. 3-3, based on differential geometry we can reparameterize the

clothoid in Cartesian coordinates [20]:

$$\begin{aligned} x =& x_0 + \int_0^l \cos\phi(t)dt \\ y =& y_0 + \int_0^l \sin\phi(t)dt \end{aligned} \tag{3.2}$$

By assuming that: (a) the origin of the curve in Cartesian coordinates is in line with the vehicle ($x_0 = 0$) and (b) that the change in heading over the course of the curve is relatively small ($\cos\phi(t) \approx 1, \sin\phi(t) \approx \phi(t) \forall t$[1]), we can arrive at the following spline representation in Cartesian coordinates:

$$y = y_0 + \phi_0 x + \frac{C^0}{2}x^2 + \frac{C^1}{6}x^3, \tag{3.3}$$

3.3.2 Edge Traversal

First, define the local coordinate frame of the robot at time t to be X_t, the set of valid configurations that adhere to the rules of the road on edge e_{ij} in the topometric map to be $\mathcal{X}_{ij}$, and the sensor swath at time t to be $\mathcal{S}_t$. Furthermore, for brevity of notation, we will assume in this section that $\mathcal{X}_{ij}$ is 2D, that is, that the road segment is approximately flat. However, this approach could easily be extended to 3D if the elevation change in the edge traversal were significant.

Next, we find the local goal as the point in X_t that is as close as possible (in the Euclidean sense) to v_j, the next waypoint in the list of graph vertices:

$$(x_{goal}, y_{goal}) = \underset{(x_t, y_t) \in \mathcal{S}_t \bigcap \mathcal{X}_{ij}}{\arg\min} ||^{X_t}v_j - [x_t, y_t]^T||_2 \tag{3.4}$$

where $^{X_t}v_j$ is the Cartesian location of vertex v_j in the X_t frame.

This formulation accounts for two important possibilities that result from the fact that we are planning in the local frame: 1) The GPS coordinates in the OSM map may be sufficiently inaccurate that navigating precisely to those coordinates would

[1] as is noted in [20] this second assumption can be explicitly enforced by truncating the curve at such a point as this approximation error grows too large - typically $\phi(t) > 15^o$ is used as a threshold

be dangerous and 2) The next vertex to be attained in the graph may be sufficiently far away that it is outside of the sensor swath of the vehicle. In such cases, it is impossible to generate a feasible reference path from the current vehicle location all the way to the next waypoint. We solve both of these problems by projecting the goal waypoint onto the feasible space in the sensor swath.

The reference trajectory therefore satisfies:

$$\tau(\alpha) : [0,1] \mapsto \mathcal{X}_{ij} \tag{3.5}$$

with $\tau(0) = [0,0]^T$ and $\tau(1) = [x_{goal}, y_{goal}]^T$.

In general, this trajectory can be internally parameterized by arc length or curvature; however, we will assume that there is some mapping $\mathcal{M}$ to Cartesian coordinates:

$$\tau(\alpha) \overset{\mathcal{M}}{\longmapsto} \mathbf{f}_t(\alpha, \Theta_t) = \begin{bmatrix} f_x(\alpha, \Theta_t) \\ f_y(\alpha, \Theta_t) \end{bmatrix} \tag{3.6}$$

where $\mathbf{f}(\alpha, \Theta)$ defines the model that is being used to represent the trajectory and Θ are the model parameters that are recursively estimated. Furthermore, since the reference trajectory is generated using noisy sensor data, we also obtain $\sigma_\tau(\alpha)$, which contains the associated uncertainty for each point along $\alpha = [0,1]$.

Collectively (3.1) and (3.2) combine to define the mapping $\mathcal{M}$ in (3.6) using the spline parameterization in (3.3) by selecting $\mathbf{f}(\alpha, \Theta)$ according to

$$\mathbf{f}(\alpha, \Theta) = \begin{bmatrix} 0 & x_{goal} & 0 & 0 \\ y_0 & \phi_0 x_{goal} & \frac{C^0}{2} x_{goal}^2 & \frac{C^1}{6} x_{goal}^3 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ \alpha \\ \alpha^2 \\ \alpha^3 \end{bmatrix} \tag{3.7}$$

.

The four-dimensional state that we will use to estimate the allowable configurations within the sensor swath will then be: $\mathcal{S}_t \bigcap \mathcal{X}_{ij} = \{y_0, \phi_0, C^0, C^1\}$. Note that this region is the maximum drivable region in the current sensor swath and rules of

the road constraints in $\mathcal{X}_{ij}$ could be used to further restrict the allowable trajectories to a particular portion of the road. In this application, since the roads are in a test facility without oncoming traffic, the trajectory was chosen to follow the center of the road.

3.3.3 Trajectory Propagation from Odometry

For modeling the motion of the vehicle we use a zero-slip kinematic odometry motion model and assume that we have access to a noisy measurement of linear forward change in position, $\Delta_x + \epsilon_x$, and a noisy measurement of the change in heading, $\Delta_\phi + \epsilon_\phi$. The kinematic model is then given by:

$$\begin{aligned} x^r_{t+1} &= x^r_t + (\Delta_x + \epsilon_x)\cos\phi^r_t \\ y^r_{t+1} &= y^r_t + (\Delta_x + \epsilon_x)\sin\phi^r_t \\ \phi^r_{t+1} &= \phi^r_t + (\Delta_\phi + \epsilon^r_\phi) \end{aligned} \tag{3.8}$$

where $X^r = [x^r_t, y^r_t, \phi^r_t]^T$ is the pose of the vehicle at time t. However, in contrast to the standard approach where this motion model is used to estimate the position of the robot in some global reference frame, in this case, we are re-aligning the "global frame" with the local frame at every timestep. In other words, $x^r_t = y^r_t = \phi^r_t = 0$, and the motion model can be significantly simplified:

$$\begin{aligned} x^r_{t+1} &= (\Delta_x + \epsilon_x) \\ y^r_{t+1} &= 0 \\ \phi^r_{t+1} &= (\Delta_\phi + \epsilon_\phi) \end{aligned} \tag{3.9}$$

We update the local reference frame to coincide with the vehicle frame. Define the measurement as (Δ_x, Δ_ϕ), with associated uncertainty σ_Δ. This is achieved through a standard homogeneous transformation matrix, $[\bar{x}_{t+1}, \bar{y}_{t+1}, 1]^T = \mathbf{T}^{-1}[x_t, y_t, 1]^T$: where

$$\mathbf{T} = \begin{bmatrix} \cos(\Delta_\phi) & -\sin(\Delta_\phi) & \Delta_x \\ \sin(\Delta_\phi) & \cos(\Delta_\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.10}$$

Then if we restrict the mapping $\mathcal{M}$ to a linear function on the model parameters of the form $\mathbf{f}(\alpha, \Theta) = \Theta\mathbf{A}(\alpha)$ then the model parameters in the new coordinate frame are given by:

$$\begin{aligned} \overline{\boldsymbol{x}}_{t+1} &= \boldsymbol{T}^{-1}\boldsymbol{x}_t \\ &= \boldsymbol{T}^{-1}\boldsymbol{f}\left(\alpha, \Theta_t\right) \\ &= T^{-1}\Theta_t\boldsymbol{A}\left(\alpha\right) \\ &= \bar{\Theta}_{t+1}\boldsymbol{A}(\alpha) \end{aligned} \tag{3.11}$$

where $\bar{\Theta}_{t+1} = \mathbf{T}^{-1}\Theta_t$ and the modifications induced by the change of reference frame are propagated to the parameters in order to maintain the fixed model form of $\mathbf{f}$.

In practice, the frequency of this odometry loop was much faster than the measurement loop described in the next subsection.

When new vehicle odometry arrives, the spline estimated is updated. The process model is given by

$$X^s_{t+1} = f(X^s_t, X^r_{t+1}, \epsilon_t) \tag{3.12}$$

where $\epsilon_t = [\epsilon^x, \epsilon^\phi, \epsilon^{C^0}, \epsilon^{C^1}]^T$ are all the additive zero-mean Gaussian sources of noise in the process model, and f is derived by substituting the updated vehicle estimate from odometry (3.9) into the spline function (3.3), and, differently from [45], we also incorporate the change in heading resulting from the odometry, which results in an additional offset added to the ϕ_0 parameter:

$$y = y_0 + (\phi_0 + \Delta_\phi + \epsilon_\phi)(x + \Delta_x + \epsilon_x) + \frac{C^0}{2}(x + \Delta_x + \epsilon_x)^2 + \frac{C^1}{6}(x + \Delta_x + \epsilon_x)^3$$

which is solved and then the coefficients are collected relative to the appropriate terms

to yield the noiseless process model:

$$X_{t+1}^s = \begin{bmatrix} 1 & \Delta_x & \frac{1}{2}\Delta_x^2 & \frac{1}{6}\Delta_x^3 \\ 0 & 1 & \Delta_x & \frac{1}{2}\Delta_x^2 \\ 0 & 0 & 1 & \Delta_x \\ 0 & 0 & 0 & 1 \end{bmatrix} X_t^s + \begin{bmatrix} \Delta_\phi \Delta_x \\ \Delta_\phi \\ 0 \\ 0 \end{bmatrix} \tag{3.13}$$

Note that this process model is linear with respect to the state but not with respect to the noise parameters. Therefore the noise Jacobians, $W = \frac{\partial f}{\partial \epsilon_t}$ are calculated at each time step in the prediction phase:

$$W = \begin{bmatrix} \phi_t^0 + \Delta_x C_t^0 + \frac{1}{2}\Delta_x^2 C_t^1 + \Delta_\phi & \Delta_x & 0 & 0 \\ C_t^0 & \Delta_x C_t^1 & 0 & 0 \\ C_t^1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.14}$$

and the final additive noise term in the filter update is given by $W^T Q W$, where Q is a diagonal matrix of noise covariances for each of the sources of noise.

3.3.4 Single Scan Road Boundary Detection

Here, we describe our method for robustly detecting road boundaries in rural areas from LiDAR data. Each new scan from the LiDAR sensor is fed through a perception system to generate "feature" detections in the sensor frame (i.e. no registration is necessary).

The road boundary detection process begins with a single scan from a Velodyne HDL-64 Lidar sensor. The velodyne sensor data is divided into rings, where each laser completes one revolution in a single scan. The top of Fig. 3-5 shows a single scan from the sensor where each point is colored based on intensity. The blue spheres highlight the points detected as lying at the road boundary.

The key insight in this method is that we can detect the road boundaries in each individual laser scan by thresholding in the *frequency domain.* The road has a much smoother texture relative to surrounding areas, the points in each ring exhibit less

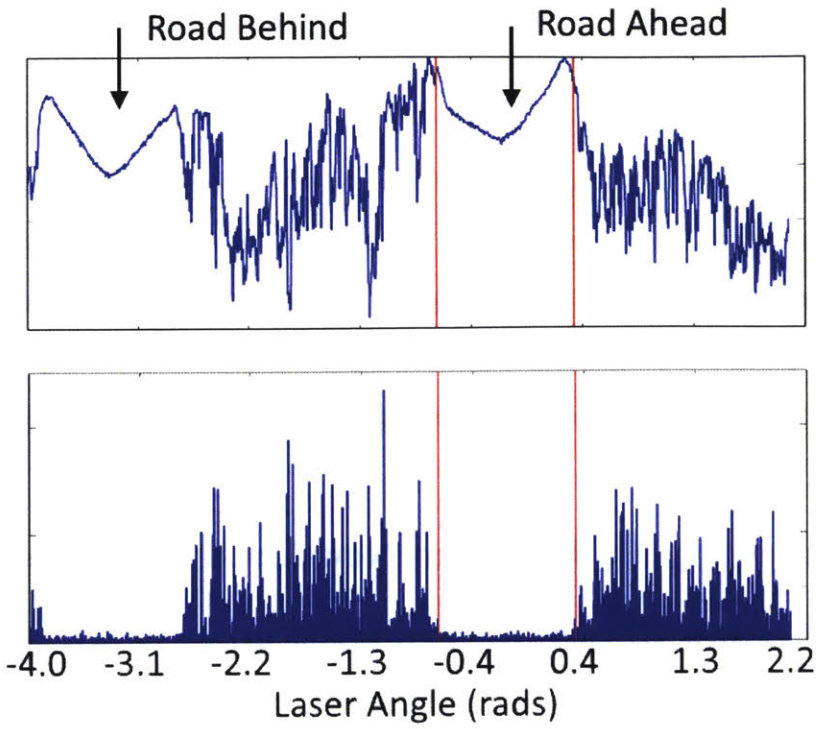

Figure 3-4: A single ring from the pointcloud viewed as a 1 dimensional signal the road texture is clearly distinguishable from the surroundings. The top graph shows the raw signal and the bottom shows the signal after post-processing. The vertical red lines indicate the location marked as the road boundary in front of the vehicle.

variation in their position compared to the points lying in the grass or trees on the side as shown in Fig. 3-5.

To characterize the texture of a region, the distance from the vehicle at each point is calculated where for the i^{th} point x_i the distance is $d_i = \|x_i\|$. Next, the distance signal is processed by taking the absolute value of the first derivative of the signal in order to remove the mean and amplify the noise. The resulting signal is a measure of the surface texture where larger values indicate rougher surfaces. The top graph in Fig. 3-4 shows the raw signal representing a single ring in a laser scan while the processed signal is shown in the bottom. The vertical red lines show the location of the actual road edges for the road in front of the vehicle.

Finally, the processed signal is searched for the road edges by first finding the standard deviation of the signal in the road area and then returning the first point

that is larger than n standard deviations where n is a tunable parameter used to choose the sensitivity of the edge detection. In practice, this value could remain constant over a variety of roads and weather conditions.

Once the edges in a single ring are found, the algorithm moves on to the next ring. In each case, the initial guess for the location of the edge is chosen as the point that has the shortest Euclidean distance to the one in the prior ring. In this way, the edge detection moves progressively along the edges of the road without the need to explore the regions of the point cloud that are far from the road. Fig. 3-5 shows the result of running the edgepoint detection algorithm on the pointclouds.

3.3.5 Trajectory Update from New Local Estimate

We use RANSAC to fit the boundary points to a spline (one for each road boundary). The reference spline trajectory is then generated as the average between the two boundary splines as shown in blue in Fig. 3-5. The result of each single scan curve fit is treated as a new incoming measurement. After performing RANSAC, the inliers are fit to the spline using a least-squares fitting procedure, and the *residual* of this fitting procedure is used to scale the covariance of the measurement update.

$$z_t = HX_t^s + \eta_t$$

where $H = [x, \frac{1}{2}x^2, \frac{1}{6}x^3]^T$ and each data point used to fit the curve can be treated as an independent measurement. For point i:

$$\eta_t^i = ||z_t^i - H^i X_t^s||^2$$

Thus, the detected road is used to generate a feasible driving trajectory along with a measure for the certainty that the trajectory is correct.

Since the perception system is generally imperfect, the uncertainty generated by perception can be propagated onto the trajectory parameters. Consequently, we arrive at a new local reference trajectory, $\mathbf{f}_{k+1}(\alpha, \hat{\Theta}_{k+1})$ where once again the predefined

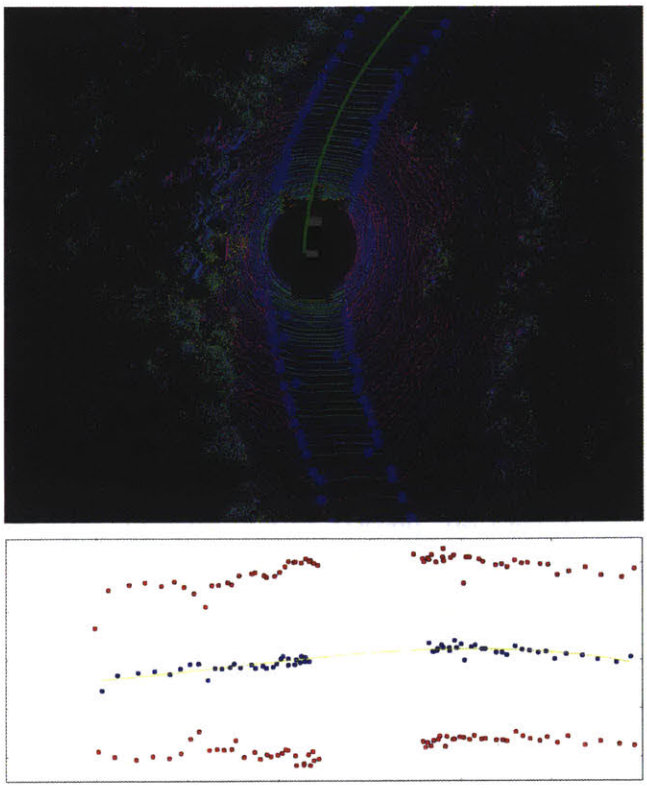

Figure 3-5: Top: We use a texture-based approach to detect the road boundaries (blue spheres) from single laser scans. Bottom: The result from a RANSAC implementation used to fit a spline to the edge points detected at the road boundaries. Note that the empty area in the center is the location of the vehicle, which occludes the sensor.

model has been fit and the trajectory uncertainty has been propagated to the parameters. Finally, since the trajectory from odometry propagation $\mathbf{f}_{k+1}(\alpha, \bar{\Theta}_{k+1})$ and the new trajectory generated from the sensor measurement $\mathbf{f}_{k+1}(\alpha, \hat{\Theta}_{k+1})$ are in the same local reference frame, they can be probabilistically fused.

3.3.6 Trajectory Following Controller

The final step to enable autonomous edge traversal is the trajectory following controller. This generates low-level control setpoints from the estimated road boundaries. The trajectory follower takes as input a parameterized trajectory spline along the road and determines the steering angle the vehicle should take to follow that trajectory. Since the trajectory generation is happening in a frame always coincident with the vehicle, we use a moving goal-point approach where the vehicle is always following a

point at a fixed distance, δ, on the input trajectory. To calculate the position and orientation errors at δ, first the spline model in (3.3) is evaluated to get the position error $\Delta y_0 = y(\delta)$, then the derivative of that model is used to calculate the angle error.

$$\begin{aligned} \frac{dy}{dt} &= \phi_0 + C^0 x + \frac{C^1}{2} x^2 \\ \Delta\phi_0 &= \tan^{-1}\left(\frac{\mathrm{dy}}{dt}\bigg|_{x=\delta}\right) \end{aligned} \tag{3.15}$$

Finally, these angle errors are passed to the position and orientation PID controllers, which were tuned to drive these errors to 0.

3.4 Road Surface Segmentation

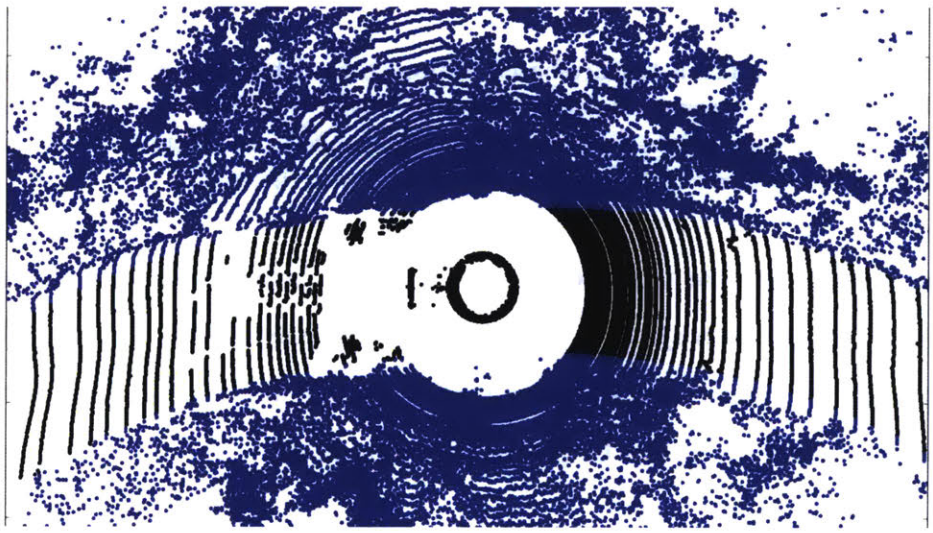

Figure 3-6: The Road Segmentation problem is solved for each time a sensor measurement is received. Every point in the pointcloud is labeled as either *road* (black) or *not road* (blue).

The boundary point detection method used in the previous section sufficed for the simple task of edge traversal. However, it could not be used for complete navigation task because the edge points at intersections do not directly map to the required trajectory. In this section, we describe a general road surface segmentation algorithm for directly segmenting all points in the pointcloud, even at road junctions. We use

a linear SVM model for road surface segmentation since its extremely fast runtime is needed for real-time operation at speed, and its accuracy was close to that of the much more performance-heavy CNNs (see section 4.5 for a comparison). Our linear SVM relies on five features extracted from the LiDAR data. The first feature is the elevation z measured directly by the sensor. Next, a measure of the surface texture is calculated using an approach similar to [39] using a sliding window for all points $\boldsymbol{p}_i$ in the neighborhood N_p. Then the local variance v_p for each point is calculated using

$$v_p = \frac{1}{|N_p|} \sum_{i \in N_p} |\boldsymbol{p}_i - \overline{\boldsymbol{p}}|^2$$

where $\overline{\boldsymbol{p}}$ is the mean over all points in N_p. This feature yields a measure of the local surface texture that is larger for rough surfaces (e.g. grass and trees) and smaller on drivable surfaces. Next, the intensity of each return is also included to account for the difference in reflectivity of road surfaces. The fourth feature is a unique laser ID accounting for physical differences in each of the 64 transmitters in the HDL-64 sensor. The laser ID is included in the training to account for systemic bias between the transmitters in a single device. Thus, new training data would need to be collected if switching to a new sensor. Finally, an indicator feature in $\{0, 1\}$ is used to indicate any points that did not return. This is important as the presence of an out-of-range value (either too far or too close) contains valuable information for segmentation. These five features are used together to classify each point as either *road* or *not road*.

3.5 Topometric Registration

In the map registration step, we aim to obtain an estimate of the robot location on the map using only odometry measurements and LiDAR scans. The problem is formulated as a maximum a posteriori (MAP) estimation problem and we describe the computation and approximation of each of its individual components. Formally,

our goal is to obtain the estimate

$$\hat{\boldsymbol{x}}_t = \arg\min_{\boldsymbol{x}_t} \left(-\ln P\left(\boldsymbol{x}_t | Z^o_{1:t}, Z^L_{1:t}\right)\right) \tag{3.16}$$

where $\boldsymbol{x}_t = [x_t, y_t, \theta_t]$ is the robot's pose in the topometric map frame at time t while $Z^o_{1:t}$ and $Z^L_{1:t}$ are the sets of all odometry and LiDAR measurements up to time t respectively. An approximation of this likelihood is required to allow for real-time operation.

3.5.1 Likelihood Approximation

Using Bayes' rule, the probability in (3.16) can be represented in terms of its prior, the LiDAR observation likelihood, and the odometry-based state prediction as

$$\begin{aligned}
P\left(\boldsymbol{x}_t | Z^o_{1:t}, Z^L_{1:t}\right) &= P\left(\boldsymbol{x}_t | Z^L_t, Z^o_{1:t}, Z^L_{1:t-1}\right) \\
&= P\left(Z^L_t | \boldsymbol{x}_t, Z^o_{1:t}, Z^L_{1:t-1}\right) \frac{P\left(\boldsymbol{x}_t | Z^o_{1:t}, Z^L_{1:t-1}\right)}{P\left(Z^L_t | Z^o_{1:t}, Z^L_{1:t-1}\right)} \\
&\propto_x P\left(Z^L_t | \boldsymbol{x}_t, Z^o_{1:t}, Z^L_{1:t-1}\right) P\left(\boldsymbol{x}_t | Z^o_{1:t}, Z^L_{1:t-1}\right) \\
&\propto_x P\left(Z^L_t | \boldsymbol{x}_t\right) P\left(\boldsymbol{x}_t | Z^o_{1:t}, Z^L_{1:t-1}\right) \\
&\propto_x P\left(Z^L_t | \boldsymbol{x}_t\right) \int P\left(\boldsymbol{x}_t | \boldsymbol{x}_{t-1}, Z^o_t\right) \cdot P\left(\boldsymbol{x}_{t-1} | Z^o_{1:t-1}, Z^L_{1:t-1}\right) \, \mathrm{d}\boldsymbol{x}_{t-1} \, .
\end{aligned}$$

We approximate the prior as a discrete distribution concentrated at the last estimate $\delta(\boldsymbol{x}_{t-1} - \hat{\boldsymbol{x}}_{t-1})$ which results in

$$P\left(\boldsymbol{x}_t | Z^o_{1:t}, Z^L_{1:t}\right) \propto_x P\left(Z^L_t | \boldsymbol{x}_t\right) \cdot P\left(\boldsymbol{x}_t | \hat{\boldsymbol{x}}_{t-1}, Z^o_t\right)$$

where the first factor represents the likelihood, and the second the prior.

3.5.2 LiDAR Observation Likelihood $P\left(Z_t^L|\boldsymbol{x}_t\right)$

Each LiDAR scan Z_t^L is represented as a set of n 3-tuples $\boldsymbol{z}_i = (x_i^L, y_i^L, l_i)$ where $x_i^L, y_i^L \in \mathbb{R}$ give the position of each measured LiDAR point in the sensor frame and l_i is a binary classification label obtained in the previously described segmentation module. We define the signed distance function $f_D\left(\boldsymbol{z}_i,\, \boldsymbol{x}_t,\, M\right)$ representing the distance from the point $\boldsymbol{z}_i$ to the nearest point on any edge in the topological map M (assuming the vehicle is in location $\boldsymbol{x}_t$). We model the probability of observing each point $\boldsymbol{z}_i$ at location $\boldsymbol{x}_t$ using a sigmoid function

$$P\left(\boldsymbol{z}_i|\boldsymbol{x}_t\right) = \begin{cases} \frac{1}{1+\exp(f_D(\boldsymbol{z}_i,\boldsymbol{x}_t,M)-r_w)} & l_i = 1 \\ 1 - \frac{1}{1+\exp(f_D(\boldsymbol{z}_i,\boldsymbol{x}_t,M)-r_w)} & l_i = 0 \end{cases} \tag{3.17}$$

where r_w is a tunable parameter that represents the likelihood of finding points labeled road, far from the map M which contains road centerlines. Notice that at the road center where $f_D\left(z_i, M\right) = 0$, $P\left(z_i\right) \approx 1$ if $l = 1$ $P\left(z_i\right) \approx 0$ if $l = 0$ while far from the road, the converse is true. Also, in this work, a constant value was sufficient for r_w since all the roads traversed were of similar width $(5 - 7\text{m})$ However, given that the lanecount and roadway class is available for most roads on OpenStreetMap, this parameter could easily be extended to depend on the road type. Finally, the overall probability of an entire LiDAR scan is computed as the product over the probabilities of the individual measurement tuples $P\left(Z_t^L|\boldsymbol{x}_t\right) = \prod_i P\left(z_i|\,\boldsymbol{x}_t\right)$.

3.5.3 Odometry-based prediction $P\left(\boldsymbol{x}_t|\hat{\boldsymbol{x}}_{t-1},\, Z_t^o\right)$

Each odometry measurement $Z_t^o = [\Delta x_t,\, \Delta y_t,\, \Delta\theta_t]$ with respect to the previous pose is obtained by fusing both the wheel encoder information and the IMU measurements using an Extended Kalman Filter [32]. The odometry-based likelihood of the future state is then modeled as

$$P\left(\boldsymbol{x}_t|\hat{\boldsymbol{x}}_{t-1},\, Z_t^o\right) \propto_x \exp\left(-\frac{||\boldsymbol{x}_t - \hat{\boldsymbol{x}}_{t-1} + Z_t^o||}{b}\right)$$

with scale factor b that is obtained, for computational reasons, through parameter tuning rather than by estimating it along with the prior.

3.5.4 Implementation for Real-Time Operation

The main remaining driver of computational cost is evaluating $P\left(Z_t^L|\boldsymbol{x}_t\right)$. This is due to the fact that $f_D\left({}^o z_i, M\right) \in O(|E|)$ (with $|E|$ being the number of edges in M) and thus $P\left(Z_t^L|\boldsymbol{x}_t\right) \in O(n \cdot |E|)$. To achieve a speed-up we employ a number of further approximations. First, we discretize the space containing the map M into cells of size r_c (0.5m for our experiments) and precompute the distance transform. This computation only needs to happen once for each map, and in practice takes <1s to compute for a map of size 1km^2. This approximation turns distance computations into a simple lookup with runtime independent of number of edges or scanned points. Next, we randomly subsample the input LiDAR scans using two tunable parameters $s \in \mathbb{N}_+$, $b \in [0, 1]$ where s is the number of points to sample, and b is the desired share of points from the road (i.e. where $l_i = 1$) to account for high label-imbalance. Finally, we simplify (3.17) by approximating

$$\left(1 + \exp\left(f_D\left(\boldsymbol{z}_i, \boldsymbol{x}_t, M\right) - r_w\right)\right)^{-1} \approx 1 - \min\left(\frac{f_D\left(\boldsymbol{z}_i, \boldsymbol{x}_t, M\right)}{r_w}, 1\right)$$

In practice, we found these approximations to have a negligible effect on system performance, while decreasing the time required to solve the problem from $10s$ to $20ms$ on a standard laptop with an Intel i7 processor.

3.6 Route Planner

The route planner is responsible for choosing the shortest path from the start location of the vehicle to the goal. It takes the registered topometric map as input, converts it into a graph structure and calculates Euclidean distances as edge weights. Next A* [24] is used to plan the shortest path from the start to the goal. Given the heavy computation required by the other modules, we cannot afford to recalculate the entire

route at a high frequency. However, since the topometric registration frequently updates the map we also cannot simply reuse the original route plan. To account for this, we use a *fast_update* method that updates the positions of the waypoints in the path in the metric space without replanning the route in the topological space. Furthermore, since it is possible the vehicle could deviate from the planned path (e.g. if the map registration hasn't updated recently) we also check for deviations from the plan and only replan with the A* algorithm as necessary. Algorithm 2 describes how the fast replanner determines when a replan of the route plan is necessary or simply a *fast_update* of the existing path to the new map. In our testing, we found that this reduced the average latency of the route planning module from $100ms$ to $1.5ms$. In section 4.7 we show that this optimization has a substantial impact on real-world system performance.

Algorithm 2 Fast Route Replanning

```
1: Inputs: (goal, map, pose, route_plan)
2: while pose not equal to goal do
3:   if route_plan is empty then
4:     Store a new route_plan from A*(pose, goal, map)
5:     Set path using route_plan
6:   else
7:     Set path using fast_update(route_plan, map)
8:     if distance(pose, path) > recalculate_threshold then
9:       Recalculate a new route_plan with A*
10:    end if
11:  end if
12:  return path
13: end while
```

Additionally, we utilize a Finite State Machine to break the entire route plan into sub-tasks (e.g. drive to the next intersection) which enables the vehicle to wait for a goal, drive, make the correct navigation choice at each intersection, and finally stop at the goal destination.

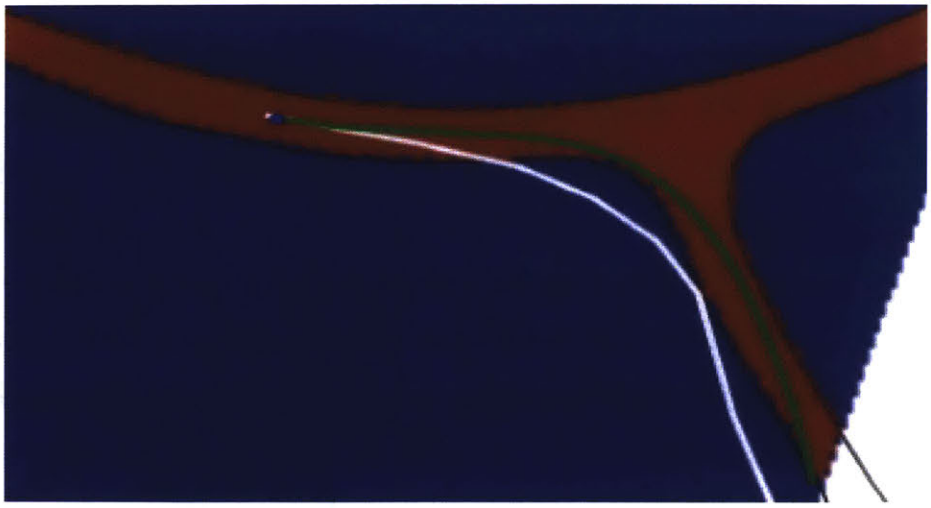

Figure 3-7: An example result from the trajectory planner. The vehicle is located at the blue dot, and received the white path from the Route Planner to indicate a right turn. The red (road) and blue (off-road) road segmentation is used to plan the green trajectory which follows the high-level navigation goal while safely remaining on the road.

3.7 Trajectory Planner

The trajectory planner generates the trajectory the vehicle must follow to reach its destination. As seen in Fig. 3-1 it takes as input a segmented LiDAR scan as well as the high-level navigation reference path from the Route Planner. We utilize a variational trajectory planner to generate safe trajectories. However, unlike typical [52] implementations, we do not obtain road boundaries from a detailed map. Instead, the road boundaries are obtained from the road segmentation module (Section 3.4) and this planner integrates the high-level navigation information obtained from the route planner to plan a locally safe path in the sensor frame. Fig. 3-7 shows an example of the typical problem this planner solves. The blue point is the current vehicle location, and the red and blue represents the road/not-road segmentation result. The white path, is the result from the topometric registration and route planning modules, which, while reliably providing navigation information (e.g. in this case "turn right") we cannot assume it to be precise at a metric level, since the topometric map does not contain such detail. Instead, the planner chose the green route, as it safely remains on the road, while also achieving the high-level navigation requirement.

The variational trajectory planner utilizes B-splines [19] to parameterize smooth trajectories. For a given reference trajectory r, the planner completes a two-stage optimization process. The first stage *local goal-point selection* consists of determining the closest point to the high-level navigation goal that is both in the vehicle field of view and also lies on a road surface. The second stage *trajectory optimization* seeks an "optimal" path from the current vehicle location to the goal-point based on a number factors described shortly.

Local Goal Point Selection

The reference path is first clipped to the sensor range $\mathcal{X} \in \mathbb{R}^2$. Then, for a reference path r consisting of n waypoints, we denote the final point r_n. Note, although by construction r_n is in the sensor range, it may not lie on the road. To obtain the local goal point we define a cost function $J(x) = [d(x), d_e(x, r_n)]$ for $x \in \mathcal{X}$ where $d(x)$ is the signed distance function from x to the free space in $\mathcal{X}$ and $d_e(x, r_n)$ is the Euclidean distance from x to r_n. Then the goal-point is found by

$$x_g = \arg\min_{x \in \mathcal{X}} J(x) W_g^T$$

where W_g is a weight vector for tuning the relative importance of the cost function terms.

Trajectory Optimization

The second stage of the variation planner creates a B-spline trajectory that begins at the vehicle's location and ends at the goal-point x_g. This optimization utilizes a three-part cost function composed of a road-distance cost

$$J_d(\boldsymbol{q}) = \frac{1}{k} \sum_{i=1}^{k} d(q_i)^2$$

where $\boldsymbol{q}$ is a candidate B-spline with k waypoints and $d(q_i)$ is, once again, the signed distance function to the road points. Next, in order to ensure that shorter paths are

preferred to equally feasible longer ones, a relative-path-length cost is computed as

$$J_l(\boldsymbol{q}) = \frac{\sum_{i=2}^{k} d_e(q_i, q_{i-1})}{d_e(x_g, x_0)}$$

where x_0 is the current vehicle location. Finally, we also seek to minimize the maximum curvature of trajectories for ride comfort. Since B-splines allow for analytic curvature calculation, we impose a maximum curvature cost as

$$J_\kappa(\boldsymbol{q}) = \max \frac{\|\sigma' \times \sigma''\|}{\|\sigma'\|^3}$$

where σ is the B-spline representation of $\boldsymbol{q}$ and σ', σ'' are the first and second derivatives. Finally, the optimal trajectory is obtained using a numerical non-linear optimization routine based on [41] to solve

$$\boldsymbol{q}_{opt} = \arg\min_{\boldsymbol{q} \in \mathcal{X}} \left[J_d(\boldsymbol{q}), \quad J_l(\boldsymbol{q}), \quad J_\kappa(\boldsymbol{q}) \right] W_q^T$$

where W_q is a weight vector for tuning the relative importance of the cost function terms. Note that the roads utilized for testing in our rural testing environment are unmarked and often single-width. Therefore, the cost function prioritized driving down the center of the road. For use on roads with multiple lanes, we could easily extend this cost function to include a cost term based on lane boundaries instead.

3.8 Summary

In this section, we have described a framework for an autonomous vehicle to navigate from an arbitrary start location to a goal point on a road network without the need for either a high definition prior map or a high precision GPS sensor. We described the major components needed to achieve this including the road segmentation of the raw sensor data, the registration of the topometric map to the segmented road, a novel route planning algorithm that is optimized for the topometric planning problem, and finally, a variational planner for generating safe trajectories toward the goal.

In the next chapters, we describe our experimental setup to test this pipeline on a full-scale autonomous vehicle in the real world. Importantly, we also perform an ablation study during which we disable various parts of the pipeline described here to see how each component contributes to the overall robustness of the entire system.

Chapter 4

Experiments

4.1 System Integration

The vehicle used for testing and data collection was a 2015 Toyota Prius V retrofitted for autonomous driving (see [33] for details). A Velodyne HDL-64S3 LiDAR sensor was used for perception, while a Microstrain 3DM-GX4 inertial measurement unit (IMU) and external wheel encoders provided onboard odometry measurements. The entire PERCEPTION and PLANNING sections of the pipeline shown in Fig. 3-1 was carried out on a standard laptop PC with a quad-core Intel-i7 processor. A separate laptop (with similar specifications) was used to run the low-level control and sensor drivers. This separation ensured that latency in the high-level perception and planning threads did not lead to delays in the low-level controllers, which need to run at a much higher frequency. These two computers communicated over an onboard Ethernet network.

Since the sensor is able to perceive the road up to 35m ahead, and the vehicle reached maximum speeds of $10\frac{\text{m}}{\text{s}}$, at 5hz, the frequency of the road segmentation was fast enough to ensure that the vehicle traveled at most 2m or $< 5\%$ along the known road spline before it was ready to incorporate new sensor data.

We integrate the MapLite system described in Chapter 3 into a pipeline for fully autonomous navigation capable of operating the vehicle from an arbitrary starting point to a user specified goal location. To execute motion plans generated by the

Variational Planner, we implement a pure pursuit controller [17], which enables us to choose a lookahead distance $d = 8m$ that was large enough to ensure rider comfort, while small enough to closely follow the reference path.

We set the vehicle speed using a dynamic speed controller such that it conforms to the following constraints:

1. Maximum Speed

2. Maximum Linear Acceleration

3. Maximum Linear Deceleration

4. Maximum Centripetal Acceleration

. The speed controller first generates a predicted path by simulating the pure pursuit controller forward a fixed distance, and then analytically solves for the maximum speed that will remain within the constraints over the length of the predicted path.

The output of the Pure Pursuit and Dynamic Speed controllers are a command steering angle and velocity. In order to achieve those commands using robust closed-loop control, a pair of PID controllers was implemented. Each of these PID controllers was tuned to obtain fast responses to speed and steering commands while minimizing overshoot and steady state error.

4.2 Test Site

The system was evaluated at a test site in Devens, MA. The roads were single-lane, mostly unmarked roads, which do not have curbs or barriers at the boundaries (see Fig. 4-1). The roads are surrounded by forest and vegetation. These features change seasonally, which makes building a high definition 3D map of this rural environment particularly challenging.

Figure 4-1: Snapshots of the car operating at the test site in Devens, MA. It is a rural area where single-lane, mostly unmarked roads, without curbs, are surrounded by forest.

4.3 Ground Truth

In order to evaluate the proposed methods, we utilized a Real-Time-Kinematic GPS Inertial Navigation System (Model: OXTS-RT3003) with a base-station transmitting Differential GPS corrections (Fig. 4-2). The base-station provides groundtruth position and orientation with accuracy of $2cm$ with a range of $10km$.

Figure 4-2: The Real-Time-Kinematic (RTK) GPS Inertial Navigation System (OXTS-RT3003) used to obtain groundtruth position of the vehicle for the evaluation experiments.

Note that the RTK-GPS system was utilized only to obtain ground truth, and to provide an initial position when loading the map before the vehicle begins moving. While navigating, the vehicle only had access to odometry measurements, and corrected for drift entirely using the MapLite system. Fig. 4-3 shows an example trajectory that was driven autonomously. The blue path is the one chosen by MapLite, while the green is the path estimated by the odometry. Clearly, the drift in the odometry would quickly cause the vehicle to deviate from the road without the MapLite corrections.

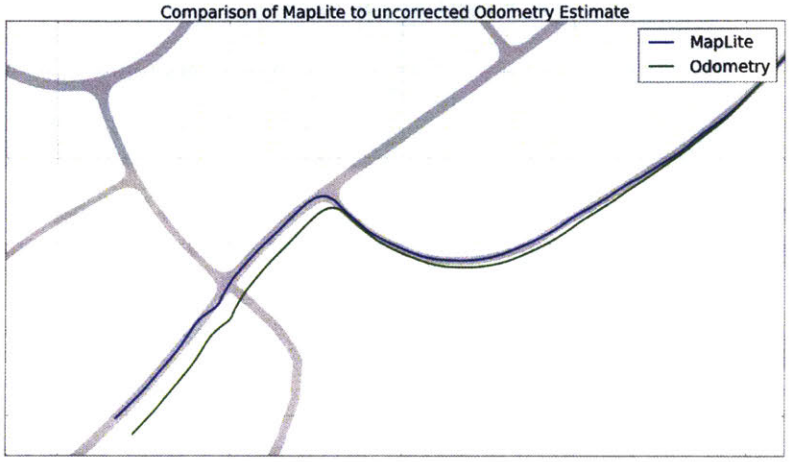

Figure 4-3: A comparison between the trajectory autonomously driven by MapLite (blue) with the path estimated by odometry (green). The shaded area is the groundtruth road surface.

4.4 Single Edge Traversal Results

In order to evaluate the accuracy of the local trajectories generated by the edge traversal base case (Sec. 3.3), the parameters $\hat{X}_t^s$ and the uncertainty P_t at each time step t were recorded. Furthermore, the raw parameters estimated directly from the model-fitting step without temporal filtering were also saved. Finally, the vehicle was manually driven down the center of the road and the path taken was recorded using the odometry measurements. These measurements served as a ground truth allowing us to compare the path predicted by the segmentation process with the path the

vehicle actually took. The root mean squared distance (RMSD) was evaluated for each point along the vehicle path, as compared to the true road center to determine how well the predicted path matched. These measurements allowed us to compare both the unfiltered and filtered road parameters as estimated by the segmentation system for evaluation. In the top of Fig. 4-4, we show these road spline estimates overlaid on a map of the test site and the corresponding deviation from the road center is shown in the bottom. Notice that while the unfiltered parameters occasionally show spikes in the deviation, the filtered estimate shown in red tends to attenuate those spikes due to the robustness gained by fusing the new measurements with the prior estimate.

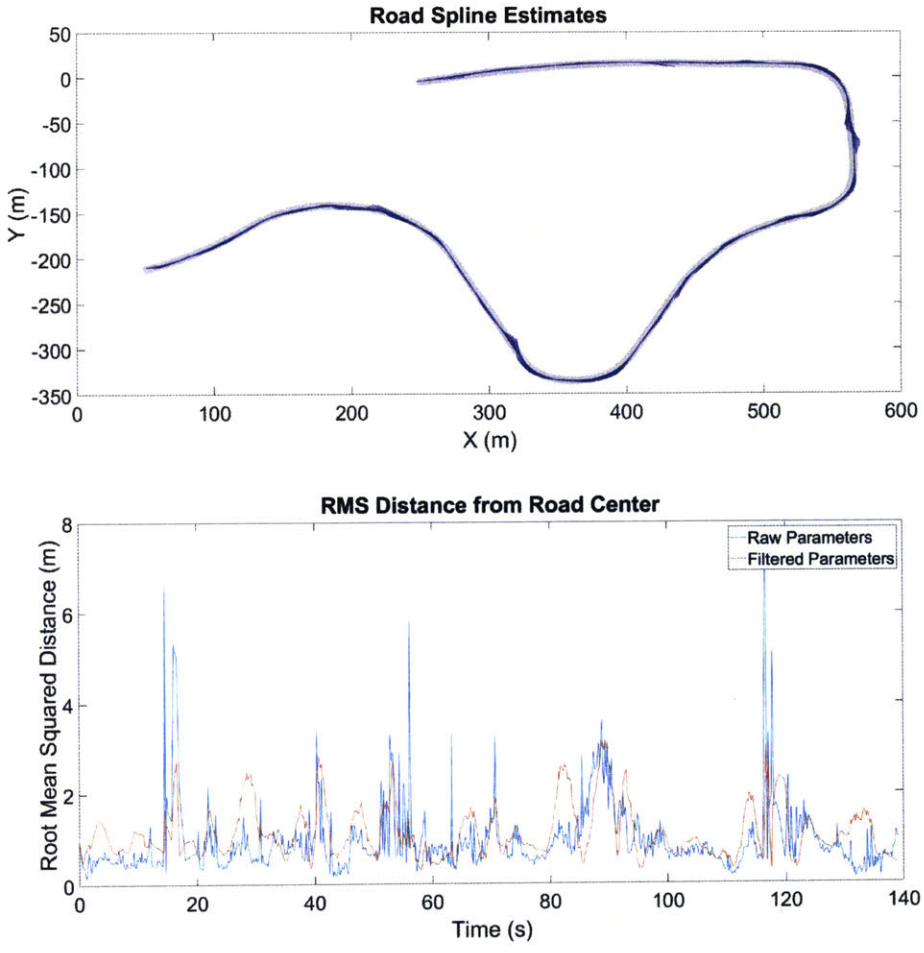

Figure 4-4: Evaluation of 700 laser scans taken over a kilometer of driving in Devens, MA. Top: Each of the road splines estimated by the segmentation process at each time step. Bottom: The Root Mean Squared Distance evaluated along each estimated spline compared to the true road center.

To better quantify the overall quality of the results and the effect of the filter, refer to Fig. 4-5. The histogram shows that for both methods more than 80% of the predicted trajectories were within 1m of the road center and more than 97% within the road boundaries at 3 meters. While we do see a drawback due to utilizing the filter in a shift of the histogram of the filtered parameters toward the right side, this is expected because the filtered parameters do not respond as quickly to new measurements giving them a larger deviation overall. As seen previously in Fig. 4-4 however, this is critical in order to robustly reject the outliers in the raw estimate caused by the highly unstructured environment.

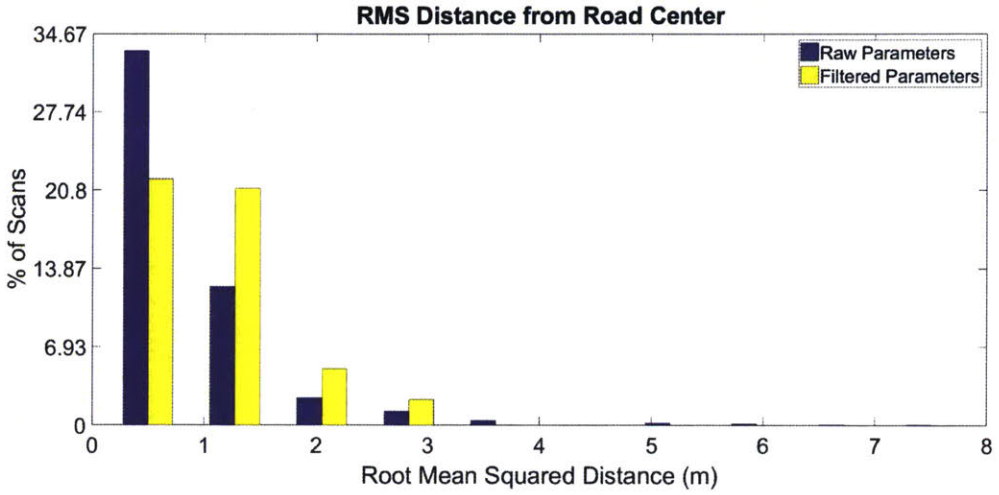

Figure 4-5: Histogram of the evaluation of the RMS Distance from the Road center for each of the 700 scans evaluated. For a half-road-width of 3 meters, more than 97% of the predicted trajectories lie within the road boundaries.

	Raw Estimate	Filtered Estimate
Mean RMS Distance from Center (m)	0.75	0.90
% of Scans Within Road Boundary	97.6	99.3
Mean RMS Distance beyond Road Boundary	1.39	0.10

Table 4.1: The RMS distances over $n = 700$ segmented scans. The first row gives the mean RMS Distance from the road center over all the scans. The second gives the percentage of the estimates that did not cross over the road boundary, while the last row shows the mean RMS distance beyond the road boundary amongst those trajectories that did go beyond the boundary.

Table 4.1 shows a summary of the results from processing $n = 700$ scans. The first row indicates that while, on average, the error was < 1m, there was a slight increase in the average deviation when using the filter. The next row shows that the vast majority of the road trajectories estimated were within the road boundaries both with and without the filter. The last column captures the benefit of the filtered estimates. Amongst the 2.4% of the raw estimates that were *not* within the road boundary in the raw estimate, the average deviation beyond the road boundary was 1.39m. On the other hand, for the filtered parameters, $< 1\%$ of estimates crossed the road boundary, and those that did had on average, 0.1m distance. It is precisely for outliers such as these that necessitate a temporal filter to ensure the system can run continuously, and robustly, for an extended period of time.

4.5 Road Segmentation Results

We trained and evaluated three segmentation approaches: 1) the linear SVM from Sec. 3.4, 2) A CNN based on SparseConvNet [23], and 3) another CNN based on PointNet [42]. Performing dense convolutions over sparse data is inefficient and computationally expensive due to fill-in. Both SparseConvNet and PointNet are specifically designed to achieve accuracy typical of CNN's on the sparse pointcloud data structure of LiDAR sensors.

Table 4.2: Evaluation of Road Surface Segmentation Methods

	Precision	**Recall**	**F1**	**Accuracy**	**Runtime (s)**
SparseConvNet	0.92	0.84	0.88	0.97	0.27
Linear SVM	0.91	0.84	0.87	0.96	0.19
PointNet	0.45	0.79	0.57	0.83	2.32

To utilize the groundtruth system to evaluate the road segmentation methods, the vehicle was first driven manually along the border of each road included in the test set. This included approximately $10km$ of roads in rural Massachusetts. Next, the road boundary GPS traces were collected and used to create a georeferenced polygon of the road boundaries. Finally, at test time, the location of the vehicle at the time of

each scan was used to project each scan into the georeferenced map. In this manner, the groundtruth class of each point was calculated to enable evaluation of each road segmentation method. Table 4.2 shows the results of evaluating the road segmentation methods on a set of 500 representative laser scans comprising over $55M$ segmented points. Notice that as expected, the SVM model is by far the fastest to segment a scan, although it does not perform quite as well as the SparseConvNet. The vastly larger number of parameters in the SparseConvNet network is also a consideration as that potentially allows it to learn a more complex variety of environments. However, that also makes it susceptible to overfitting. For our systems level evaluations presented in section 4.7, we chose to use the Linear SVM as its low runtime is paramount for enabling real-time operation at speed.

4.6 Topometric Registration Results

We evaluate the topometric registration by comparing the location of the road centerline in the registered topometric map, to the actual road center as measured by the groundtruth. We compute the root-mean-square error (RMSE) along the portion of the map that is in the sensor range (as the topometric registration only considers that portion of the map. As a baseline, we also compute the RMSE using the prior to map registration (e.g. based solely on the odometry estimate).

Fig. 4-6 shows RMSE before (blue) and after (green) topometric registration computed during $380s$ of driving. As expected, the error based solely on odometry starts out quite low. However, due to the inherent drift in dead-reckoning measurements, it quickly drifts to $> 5m$ RMSE, which is much too large to use for autonomous operation. The maps corrected by topometric registration on the other hand, have a consistently smaller RMSE, which reduces by 85.7% on average. Furthermore, it reduces the maximum RMSE by 79.7%. The corrected maps based on the registration alone are not sufficient for blindly following the topometric map because the topometric map does not contain the detailed lane-level metric information needed for autonomous driving. However, these corrected maps are sufficient to generate a

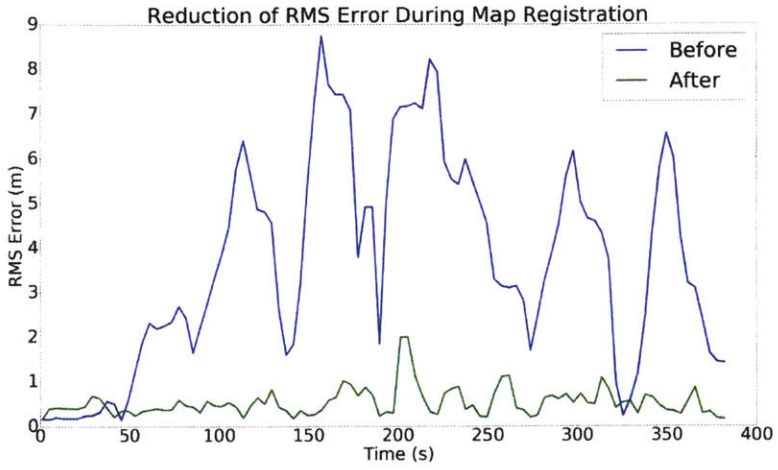

Figure 4-6: A comparison between the root-mean-square error (RMSE) of the estimated trajectory before (Blue) and after (green) topometric registration. The estimate shown in blue displays the expected odometry drift while the green estimates maintain consistently lower error using map registration corrections.

reference path that incorporates the high-level navigation information into the motion planner which generates the actual control trajectory. This incorporation of the reference path from the topometric registration is described in section 3.7.

4.7 System Evaluation

We compare the MapLite system performance against that of an expert human driver to quantify performance metrics. First, we chose a standardized test route of $1.2km$ in length requiring the traversal of a total of ten intersections (3 right-turns, 4 left-turns, and 3 straight). The test route was in a rural area near Devens, Massachusetts with overgrown, mostly unmarked, single-lane roads. (See Fig. 4-1)

Next, we utilize the GPS groundtruth to measure a baseline traversal of the test route. Finally, both the human driver and MapLite steer the vehicle along the test route twice. For these runs, we compute two metrics: 1) Accuracy and 2) Precision. The accuracy metric is the RMSE of the test run compared to the baseline. The precision metric is the root-mean-square deviation (RMSD) of the second run compared to the first. This gives a measure of the repeatability of the system independent of

Table 4.3: MapLite Performance Evaluation and Ablation Study

	Precision(m)	Accuracy(m)	Intervention
Human Expert	0.31	0.43	0
MapLite	0.17	0.49	0
1) Map Registration	0.34	0.81	0
2) Variational Planner	0.28	0.81	1.3
3) 1 & 2	-	-	>10
4) Smart Replanning	0.55	0.57	0

how closely it navigates to the baseline.

We found that the expert human driver could navigate the test route with an Accuracy of $RMSE = 43cm$ while the MapLite system obtained $RMSE = 49cm$. While this was not quite as close to the baseline as the human driver, it is quite close considering the $6cm$ is quite a small difference with respect to the size of typical lanes. Perhaps surprisingly, when it came to precision, the MapLite system outperformed the human driver with $RMSD$ of only $17cm$ while the human had $31cm$. We attribute this to the unique ability of the autonomous system to precisely reproduce the same trajectory, while human drivers typically have more variability even when navigating safely.

Next, to exhaustively test the system, we also randomly selected GPS destinations and for each one, MapLite autonomously planned a route and piloted the vehicle from its starting position to the destination. Once the vehicle came to a stop, a new destination was input, and the process repeated. We executed this test for a total of 16.1km consisting of 107 intersection traversals. At no point throughout this test was safety driver intervention required, and the vehicle navigated safely to every one of the destination points.

Finally, to determine the value of each of the components described previously, we performed an ablation test wherein we removed a single component and computed the accuracy and precision measurements described previously by traversing the test route twice more. We did this for each of the following four ablation scenarios:

1. We remove the Topometric Registration component leaving the map fixed

2. We remove the Variational Planner, instead directly following the reference from

the topometric map

3. We combine both 1) and 2)

4. We remove the Smart Replanning, instead recomputing a new route plan each time the map is updated

For each test, we also recorded the required interventions/km by the safety driver to prevent road departure. Table 4.3 compares the performance of the Human driver, MapLite, and each of the ablation tests. For 1) Removal of Topometric Registration, no interventions were needed. However, the performance of the system was worse in both accuracy and precision. While it might seem surprising that the system can operate at all with a fixed topometric map, it should be noted that in this system, the map is used mostly as a source of high-level navigation information, while the local trajectory plan is computed by the Variational Planner.

Next, for 2) removal of the Variational Planner the driver was required to intervene $1.3 interventions/km$. Furthermore, both the accuracy and precision metrics increased. Next, for 3) both the Topometric Registration and Variational Planner were removed. In this case, however, the vehicle could not safely navigate autonomously, and the test was aborted after 10 interventions. This is interesting because it provides evidence that while the Topometric Registration and Variational Planner are supplementary (the vehicle performed best when they were both used) they are also complementary (each of them alone could enable at least some measure of autonomous navigation).

Finally, in 4) we remove the Smart Replanning feature of the route planner. In that test again, there were no interventions required. However, both the precision and accuracy were dramatically worse. This can be explained by the increased processing time required to recompute the route plan each time the map was updated, which caused a lag between when new sensor data arrived and when the vehicle was able to respond.

4.8 Simulations

While the testing location in Massachusetts provided a range of intersection topologies in which to test MapLite, we leveraged autonomous driving simulation to assess the performance of MapLite in more complex traffic topologies that were not represented at the testing site. We chose to use the CARLA [21] autonomous driving simulator for our analysis because it includes realistic environments with a larger diversity of road topologies. In particular, we selected the built-in Town03 urban environment in CARLA for testing, shown in Fig. 4-7, which includes a multi-lane traffic circle with multi-lane incoming roads.

Figure 4-7: A traffic circle in the CARLA simulator. We used CARLA to test more complex road topologies than were available at our test site. A traffic circle in the CARLA simulator. We used CARLA to test more complex road topologies than were available at our test site. Here the vehicle must enter, traverse, and exit the traffic circle to reach its destination without access to the GPS groundtruth provided by the simulator.

We assessed the performance of MapLite in this environment in the same manner as we did the real-world vehicle. In this case, our test route included both entering and exiting the traffic circle, and the groundtruth GPS measurement came from the simulator itself. Using this measurement as the baseline, we obtained a precision $RMSD = 10cm$ and accuracy $RMSE = 38cm$. Both of these values are actually lower than the corresponding values for the real-world vehicle (see Table 4.3). However, while the simulation does include random noise, and, in fact, when the vehicle navigates solely based on odometry it quickly drifts from the road, the simulation

does not accurately model all of the various noise sources in a real vehicle, and thus its better precision and accuracy are not unexpected.

Chapter 5

Conclusion

5.1 Conclusion

In this work, we have demonstrated a novel autonomous navigation system capable of piloting a vehicle on rural roads without a detailed prior map. Instead, the system uses publicly available, lightweight, topometric maps from OpenStreetMap to obtain the information needed for high-level planning. A road segmentation algorithm is used to segment the road using the onboard LiDAR sensor. Then a topometric registration fits map information to the scene observed by the onboard sensors. Motion planning is then executed in the local frame to enable the vehicle to navigate safely toward its high-level navigation goal without the need for a detailed prior map of the region.

Prior work mostly relies on detailed prior maps for navigation, but these maps can be very costly to build, difficult to scale, and hard to maintain in rapidly changing rural environments. By using instead the existing topometric maps, we hope this technique can greatly expand the scale and scope of autonomous mobility solutions.

To evaluate our work, we performed extensive testing on over 15km of real world driving including >100 intersection traversals. We have found that the system could successfully navigate to an arbitrary goal location requested by the passenger without the need for a detailed prior map or high-precision GPS. We further extended this testing using simulation to more complex intersection topologies including traffic circles. Finally, we also completed an ablation study to determine the effects of each

of the various components in our navigation pipeline. This study demonstrates that the components are not only supplementary (the system performed best when all components were used) but also complementary (each of the components alone could enable at least some level of autonomous navigation).

We envision that this system could be used to enable autonomous vehicles to leave the small well-mapped urban areas they have been confined to, and utilize the wide availability of topometric maps to more easily deploy throughout the world. It is only through such widespread adoption that perhaps the vision of the Toyota-CSAIL Joint Research Institute can someday be realized: to build a car that cannot be the cause of a motor vehicle accident.

5.2 Lessons Learned

Over the course of this work, we have learned some key lessons that provide valuable insight for anyone tackling projects of this nature. Firstly, while implementing the algorithms necessary to run MapLite, we found the performance to be a key factor. In particular, the road segmentation (Sec. 3.4), topometric map registration (Sec. 3.5), and variational planner (Sec. 3.7) all presented a challenge to enable real-time performance. Especially in the context of autonomous driving, it is critical to keep lag and latency to a minimum for the entire system to function. In the road segmentation component, we chose to use the SVM solution for performance reasons even though it was not quite as accurate as some of our other solutions. Similarly, in Sec. 3.5.4 we describe in detail a few approximations and techniques that were necessary to be able to perform the map registration fast enough. Throughout this work, we found that tools such as *ROSPROFILER* [47] that allow careful tracking of the frequency of each algorithm as well as the overall lag in the system were invaluable for meeting our latency budget.

Additionally, we found that as with any system with many interconnected parts, the complexity grows exponentially with the number of components. This made debugging issues difficult as an error in a single algorithm could propagate throughout

the system. To deal with these issues efficiently, we found a few techniques especially helpful. Firstly, whenever possible we tested system components in simulation and on stored data collections. This allows for analyzing components in isolation, replaying problematic events, and carefully tracing inputs and outputs to determine root cause without the large overhead of field-testing a full-size autonomous vehicle. We also found it very helpful to build simpler versions of the complete system to enable testing single components or smaller subsets. For example, we could test the entire control pipeline by setting up a groundtruth perception component to separate the challenging perception task from the rest of the control pipeline.

Not only was this project complex due to the number of components, it was also difficult due to the many challenges associated with field-testing a full-size autonomous car. While cars are simple low degree-of-freedom robots in theoretical models, they can be quite complex machines in the real world. For example, changes in tire pressure can affect the wheel odometry, the torque-speed curve for the steering column is highly non-linear, road conditions can drastically change coefficients of friction that affect the path travelled, and even internal systems in the vehicle safety computer can sometimes detect autonomous interference and trigger safety responses such as ABS braking or shutdown of the power steering system. These are just a few examples of the type of issues that can crop up when testing a theoretical algorithm on a complex physical machine. To deal with these we found it helpful to ensure that whenever the vehicle was undergoing testing, all data was carefully logged, along with detailed descriptions of the experiment setup, ambient conditions, and parameters and configuration values. This helped to ensure the data gleaned was as useful as possible for future iterations.

Finally, we initially envisioned this project as a substitute for the typical approach of constructing high-definition detailed prior maps before attempting autonomous driving. However, over the course of this work, we have also heard from researchers who have expressed interest in this approach for other applications as well. For example, a similar system could be used to partially or fully automate the building and annotation of detailed maps using existing topometric maps. Additionally, advanced

driver-assistance systems (ADAS) could use such a system to ensure safety even for human drivers when they are driving in areas without HD maps.

5.3 Limitations and Future Work

While the present work has focused on the driving task, in the future we would like to investigate how autonomous vehicles can be used to automatically create and update topometric maps.

Furthermore, in this work, we focus only on the navigation task, which typically relies on a detailed prior map. The problem of planning in the presence of dynamic obstacles is the same as that faced by typical systems that do utilize detailed prior maps and is out of the scope of this work. However, high definition prior maps are sometimes also used as a prior to enable higher quality detection of such dynamic actors. For example, a neural network trained as a classifier should be more likely to detect a pedestrian in a region where pedestrians are commonly found (e.g. crosswalks) and less likely in areas where they are seldom present (e.g. freeways). Such priors can allow for detection of critical obstacles even under occlusion, challenging illumination, or inclement weather while mitigating the risk of spurious detections. While this work investigated the use of topometric maps for navigation information, we have not yet considered them as a prior for other areas of perception like obstacle detection. In future work, we look forward to investigating to what extent a detailed map prior is necessary for other perception tasks, and whether we can extend this approach to glean similar information from a topometric map as well.

Finally, in this work we assumed the topometric map does not perfectly model the actual road centerlines, but rather only serves as an approximation and, we used the publicly available OSM to demonstrate that this is sufficient for navigation. However, we have not yet quantified how inaccurate the topometric map can be before navigation becomes impossible. In the future, we hope to analyze exactly how much error can exist in the topometric map before navigation errors become likely.

Bibliography

[1] José M Alvarez, A Lopez, and Ramon Baldrich. Illuminant-invariant model-based road segmentation. In *2008 IEEE Intelligent Vehicles Symposium*, pages 1175–1180. IEEE, 2008.

[2] Mohamed Aly. Real time detection of lane markers in urban streets. In *2008 IEEE Intelligent Vehicles Symposium*, pages 7–12. IEEE, 2008.

[3] Alexander Amini, Guy Rosman, Sertac Karaman, and Daniela Rus. Variational end-to-end navigation and localization. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8958–8964. IEEE, 2019.

[4] Augusto Luis Ballardini, Simone Fontana, Axel Furlan, Dario Limongi, and Domenico Giorgio Sorrenti. A framework for outdoor urban environment estimation. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pages 2721–2727. IEEE, 2015.

[5] Augusto Luis Ballardini, Daniele Cattaneo, Simone Fontana, and Domenico Giorgio Sorrenti. Leveraging the osm building data to enhance the localization of an urban vehicle. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 622–628. IEEE, 2016.

[6] Augusto Luis Ballardini, Daniele Cattaneo, Simone Fontana, and Domenico Giorgio Sorrenti. An online probabilistic road intersection detector. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 239–246. IEEE, 2017.

[7] World Bank. Trading economics: Paved roads in the united states, 2008. URL `http://www.tradingeconomics.com/united-states/roads-paved-percent-of-total-roads-wb-data.html`. Accessed: 2019-09-10.

[8] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[9] Michael Bosse, Paul Newman, John Leonard, and Seth Teller. Simultaneous localization and map building in large-scale cyclic environments using the atlas

framework. *The International Journal of Robotics Research*, 23(12):1113–1139, 2004.

[10] Alberto Broggi and Simona Berte. Vision-based road detection in automotive systems: A real-time expectation-driven approach. *Journal of Artificial Intelligence Research*, 3:325–348, 1995.

[11] Jaemin Byun, Ki-in Na, Beom-su Seo, and Myungchan Roh. Drivable road detection with 3d point clouds based on the mrf for intelligent vehicle. In *Field and Service Robotics*, pages 49–60. Springer, 2015.

[12] Luca Caltagirone, Samuel Scheidegger, Lennart Svensson, and Mattias Wahde. Fast lidar-based road detection using fully convolutional neural networks. In *2017 ieee intelligent vehicles symposium (iv)*, pages 1019–1024. IEEE, 2017.

[13] Luca Caltagirone, Mauro Bellone, Lennart Svensson, and Mattias Wahde. Lidar–camera fusion for road detection using fully convolutional neural networks. *Robotics and Autonomous Systems*, 111:125–131, 2019.

[14] Zhe Chen, Jing Zhang, and Dacheng Tao. Progressive lidar adaptation for road detection. *IEEE/CAA Journal of Automatica Sinica*, 6(3):693–702, 2019.

[15] Howie Choset and Keiji Nagatani. Topological simultaneous localization and mapping (slam): toward exact localization without explicit localization. *IEEE Transactions on robotics and automation*, 17(2):125–137, 2001.

[16] Felipe Codevilla, Matthias Miiller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9. IEEE, 2018.

[17] R Craig Coulter. Implementation of the pure pursuit path tracking algorithm. Technical report, Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, 1992.

[18] MIT CSAIL. Announcing the CSAIL / TOYOTA Research Center, 2015. URL `https://toyota.csail.mit.edu/`.

[19] Carl De Boor. On calculating with b-splines. *Journal of Approximation theory*, 6(1):50–62, 1972.

[20] Ernst D. Dickmanns and Birger D. Mysliwetz. Recursive 3-d road and relative ego-state recognition. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 2:199–213, 1992.

[21] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. *arXiv preprint arXiv:1711.03938*, 2017.

[22] Georgios Floros, Benito Van Der Zander, and Bastian Leibe. Openstreetslam: Global vehicle localization using openstreetmaps. In *2013 IEEE International Conference on Robotics and Automation*, pages 1054–1059. IEEE, 2013.

[23] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9224–9232, 2018.

[24] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[25] Benjamin Kuipers, Joseph Modayil, Patrick Beeson, Matt MacMahon, and Francesco Savelli. Local metrical and global topological maps in the hybrid spatial semantic hierarchy. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, volume 5, pages 4845–4851. IEEE, 2004.

[26] Richard B Langley. The utm grid system. *GPS world*, 9(2):46–50, 1998.

[27] John Leonard, Jonathan How, Seth Teller, Mitch Berger, Stefan Campbell, Gaston Fiore, Luke Fletcher, Emilio Frazzoli, Albert Huang, Sertac Karaman, et al. A perception-driven autonomous urban vehicle. *Journal of Field Robotics*, 25 (10):727–774, 2008.

[28] Jesse Levinson and Sebastian Thrun. Robust vehicle localization in urban environments using probabilistic maps. In *2010 IEEE International Conference on Robotics and Automation*, pages 4372–4378. IEEE, 2010.

[29] Jesse Levinson, Michael Montemerlo, and Sebastian Thrun. Map-based precision vehicle localization in urban environments. In *Robotics: science and systems*, volume 4, page 1. Citeseer, 2007.

[30] Wei Liu, Hongliang Zhang, Bobo Duan, Huai Yuan, and Hong Zhao. Vision-based real-time lane marking detection and tracking. In *2008 11th International IEEE Conference on Intelligent Transportation Systems*, pages 49–54. IEEE, 2008.

[31] Joel C McCall and Mohan M Trivedi. An integrated, robust approach to lane marking detection and lane tracking. In *IEEE Intelligent Vehicles Symposium, 2004*, pages 533–537. IEEE, 2004.

[32] Thomas Moore and Daniel Stouch. A generalized extended kalman filter implementation for the robot operating system. In *Intelligent autonomous systems 13*, pages 335–348. Springer, 2016.

[33] Felix Naser, David Dorhout, Stephen Proulx, Scott Drew Pendleton, Hans Andersen, Wilko Schwarting, Liam Paull, Javier Alonso-Mora, Marcelo H Ang, Sertac Karaman, et al. A parallel autonomy research platform. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 933–940. IEEE, 2017.

[34] Department of Transportation. Summary of motor vehicle crashes, 2016. URL `https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812580`.

[35] Department of Transportation. 2016 fatal motor vehicle crashes: Overview, 2016. URL `https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812456`.

[36] OpenStreetMap contributors. OpenStreetMap, 2017. URL `https://www.openstreetmap.org`.

[37] Teddy Ort, Liam Paull, and Daniela Rus. Autonomous vehicle navigation in rural environments without detailed prior maps. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2040–2047. IEEE, 2018.

[38] Teddy Ort, Krishna Jatavallabhula, Rohan Banerjee, Sai Krishna Gottipati, Dhaivat Bhatt, Igor Gilitschenski, Liam Paull, and Daniela Rus. Maplite: Autonomous intersection navigation without a detailed prior map. *IEEE Robotics and Automation Letters*, 2019.

[39] Mark Pauly, Markus Gross, and Leif P Kobbelt. Efficient simplification of point-sampled surfaces. In *IEEE Visualization, 2002. VIS 2002.*, pages 163–170. IEEE, 2002.

[40] Kevin Peterson, Jason Ziglar, and Paul E Rybski. Fast feature detection and stochastic parameter estimation of road shape using multiple lidar. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 612–619. IEEE, 2008.

[41] Michael JD Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The computer journal*, 7(2): 155–162, 1964.

[42] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[43] Ananth Ranganathan and Frank Dellaert. Online probabilistic topological mapping. *The International Journal of Robotics Research*, 30(6):755–771, 2011.

[44] Heiko G Seif and Xiaolong Hu. Autonomous driving in the icity—hd maps as a key challenge of the automotive industry. *Engineering*, 2(2):159–162, 2016.

[45] Young-Woo Seo and Ragunathan Raj Rajkumar. Detection and tracking of boundary of unmarked roads. In *17th International Conference on Information Fusion (FUSION)*, pages 1–6. IEEE, 2014.

[46] Gabe Sibley, Christopher Mei, Ian Reid, and Paul Newman. Vast-scale outdoor navigation using adaptive relative bundle adjustment. *The International Journal of Robotics Research*, 29(8):958–980, 2010.

[47] Robot Operating System. Ros.org - rosprofiler, 2020. URL `http://wiki.ros.org/rosprofiler`. Accessed: 2020-01-27.

[48] Ryan W Wolcott and Ryan M Eustice. Visual localization within lidar maps for automated urban driving. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 176–183. IEEE, 2014.

[49] Ryan W Wolcott and Ryan M Eustice. Fast lidar localization using multiresolution gaussian mixture maps. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 2814–2821. IEEE, 2015.

[50] Geng Zhang, Nanning Zheng, Chao Cui, Yuzhen Yan, and Zejian Yuan. An efficient road detection method in noisy urban environment. In *2009 IEEE Intelligent Vehicles Symposium*, pages 556–561. IEEE, 2009.

[51] Jinyou Zhang and H-H Nagel. Texture-based segmentation of road images. In *Proceedings of the Intelligent Vehicles' 94 Symposium*, pages 260–265. IEEE, 1994.

[52] Julius Ziegler, Philipp Bender, Markus Schreiber, Henning Lategahn, Tobias Strauss, Christoph Stiller, Thao Dang, Uwe Franke, Nils Appenrodt, Christoph G Keller, et al. Making bertha drive—an autonomous journey on a historic route. *IEEE Intelligent transportation systems magazine*, 6(2):8–20, 2014.