AUTOMATIC LASER CALIBRATION, MAPPING, AND LOCALIZATION FOR AUTONOMOUS VEHICLES

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Jesse Sol Levinson
August 2011

This dissertation is online at: http://purl.stanford.edu/zx701jr9713

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Sebastian Thrun, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Daphne Koller

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Marc Levoy

Approved for the Stanford University Committee on Graduate Studies.

Patricia J. Gumport, Vice Provost Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.

Abstract

This dissertation presents several related algorithms that enable important capabilities for self-driving vehicles.

Using a rotating multi-beam laser rangefinder to sense the world, our vehicle scans millions of 3D points every second. Calibrating these sensors plays a crucial role in accurate perception, but manual calibration is unreasonably tedious, and generally inaccurate. As an alternative, we present an unsupervised algorithm for automatically calibrating both the intrinsics and extrinsics of the laser unit from only seconds of driving in an arbitrary and unknown environment. We show that the results are not only vastly easier to obtain than traditional calibration techniques, they are also more accurate.

A second key challenge in autonomous navigation is reliable localization in the face of uncertainty. Using our calibrated sensors, we obtain high resolution infrared reflectivity readings of the world. From these, we build large-scale self-consistent probabilistic laser maps of urban scenes, and show that we can reliably localize a vehicle against these maps to within centimeters, even in dynamic environments, by fusing noisy GPS and IMU readings with the laser in realtime. We also present a localization algorithm that was used in the DARPA Urban Challenge, which operated without a prerecorded laser map, and allowed our vehicle to complete the entire six-hour course without a single localization failure.

Finally, we present a collection of algorithms for the mapping and detection of traffic lights in realtime. These methods use a combination of computer-vision techniques and probabilistic approaches to incorporating uncertainty in order to allow our vehicle to reliably ascertain the state of traffic-light-controlled intersections.

Acknowledgements

As members of the Stanford Driving Team, Mike Montemerlo, Dirk Haehnel, Hendrik Dahlkamp, David Stavens, Alex Teichman, Michael Sokolsky, Soeren Kammel, Charles DuHadway, David Jackson, David Held, Ganymed Stanek, Jake Askeland, Jan Becker, Jennifer Dolson, J. Zico Kolter, Dirk Langer, Oliver Pink, Christian Plagemann, and Moritz Werling contributed to various aspects of this work and to the supporting hardware and software infrastructure used in our autonomous vehicle.

As fellow students of Professor Thrun, Varun Ganapathi and James Diebel provided valuable insight in conversations about our research.

Portions of this research were funded by DARPA, Volkswagen, Google, Intel, Qualcomm, and Boeing. I received additional financial support as a National Science Foundation Graduate Research Fellow from 2005 to 2008 and from the Qualcomm Innovation Fellowship in 2010.

I extend deep thanks to my dissertation advisor, Sebastian Thrun, for his invaluable insight and guidance during this process. Much thanks goes out to the rest of my faculty committee: Daphne Koller (Examiner/Reader), Marc Levoy (Examiner/Reader), Vaughan Pratt (Reader), and Clifford Nass (Examination Chair).

I would also like to recognize the support and encouragement of my parents, Arthur and Rita Levinson, and my sister, Anya Levinson.

Contents

Abstract **iv**

Acknowledgements **v**

1 Introduction **1**

2 Mapping and Localization **5**

2.1 Introduction 5

2.2 Road Mapping with GraphSLAM 8

2.2.1 Modeling Motion 8

2.2.2 Map Representation 9

2.2.3 Latent Variable Extension for GPS 10

2.2.4 The Extended GraphSLAM Objective Function 11

2.2.5 Integrating Out the Map 11

2.2.6 Computing the Map 13

2.3 Online Localization 14

2.3.1 Localization with Particle Filters 14

2.3.2 Data Management 16

2.4 Experimental Results 18

2.4.1 Mapping 18

2.4.2 Localization 20

2.4.3 Autonomous Driving 23

2.5 Conclusion 23

3 Extension to Probabilistic Maps **25**
3.1 Introduction . . . 25
3.2 Probabilistic Maps . . . 27
3.2.1 Map alignment using GraphSLAM . . . 28
3.2.2 Laser calibration . . . 28
3.2.3 Map creation . . . 29
3.3 Online Localization . . . 31
3.3.1 Motion update . . . 31
3.3.2 Measurement update . . . 32
3.3.3 Most likely estimate . . . 35
3.4 Experimental Results . . . 35
3.4.1 Quantitative Results . . . 36
3.4.2 Autonomous Success . . . 38
3.5 Conclusion . . . 39

4 Localization Using a Vector Road Map **42**
4.1 Introduction . . . 42
4.2 Lane Marker Matching Constraint . . . 46
4.2.1 RNDF lane marker response prior . . . 47
4.2.2 Laser lane marker response filter . . . 47
4.2.3 Computation of RNDF alignment strength . . . 48
4.3 Curb Avoidance Constraint . . . 49
4.3.1 RNDF lane corridor prior . . . 50
4.3.2 Curb response filter . . . 50
4.3.3 Computation of RNDF alignment strength . . . 52
4.4 GPS Constraint . . . 53
4.5 Lateral Localization . . . 54
4.6 Results . . . 56
4.6.1 Hardware requirements . . . 56
4.6.2 Qualitative performance . . . 56
4.6.3 Quantitative performance . . . 58

4.7 Conclusion . . . 59

5 Unsupervised Calibration for Multi-beam Lasers 61

5.1 Introduction . . . 61
5.2 Extrinsic Calibration . . . 64
5.3 Intrinsic calibration of each beam . . . 68
5.4 Remittance calibration . . . 69
5.4.1 Deterministic calibration . . . 70
5.4.2 Bayesian calibration . . . 71
5.5 Experimental Results . . . 74
5.5.1 Extrinsic calibration . . . 74
5.5.2 Intrinsic calibration . . . 77
5.5.3 Remittance calibration . . . 79
5.6 Related extensions . . . 80
5.6.1 Single-beam laser calibration . . . 81
5.6.2 Calibrating time delays . . . 83
5.7 Conclusion . . . 84

6 Traffic Light Mapping and State Detection 88

6.1 Introduction . . . 88
6.2 Traffic Light Mapping . . . 91
6.3 Traffic Light State Detection . . . 93
6.3.1 Prominent Failure Cases . . . 93
6.3.2 Traffic Light Tracking . . . 95
6.3.3 Uncertainty Discussion . . . 97
6.3.4 Probabilistic Template Matching . . . 99
6.3.5 State Detection Pipeline . . . 102
6.4 Experimental Results . . . 103
6.5 Localization and Calibration Extension . . . 108
6.5.1 Incorporating Vehicle Localization . . . 108
6.5.2 Extrinsic Camera Calibration . . . 110
6.6 Conclusion . . . 112

7 Conclusions **114**

A Hardware and software architecture **117**

A.1 Vehicle hardware 117

A.2 Software architecture 119

A.3 Object classification 120

A.4 Planning 121

A.4.1 Lateral Motion 122

A.4.2 Longitudinal Movement 122

A.4.3 Combining Lateral and Longitudinal Curves 123

A.5 Control 124

Bibliography **127**

List of Figures

1.1 Junior was the first robot to cross the finish line in the 2007 DARPA Urban Challenge, winning second place overall. 2

2.1 The acquisition vehicle is equipped with a tightly integrated inertial navigation system which uses GPS, IMU, and wheel odometry for localization. It also possesses laser range finders for road mapping and localization. . . . 6

2.2 Visualization of the scanning process: the LIDAR scanner acquires range data *and* infrared ground reflectivity. The resulting maps therefore are 3-D infrared images of the ground reflectivity. Notice that lane markings have much higher reflectivity than pavement. 7

2.3 Example of ground plane extraction. Only measurements that coincide with the ground plane are retained; all others are discarded (shown in green here). As a result, moving objects such as car (and even parked cars) are not included in the map. This makes our approach robust in dynamic environments. 9

2.4 Patch of the map acquired in bright sunlight on a sunny day (left), and at night in heavy rain (right). By correlating scans with the map, instead of taking absolute differences, the weather-related brightness variation has almost no effect on localization. 14

2.5 Aerial view of Burlingame, CA. Regions of overlap have been adjusted according to the methods described in this chapter. Maps of this size tend not to fit into main memory, but are swapped in from disk automatically during driving. 16

2.6 Campus Dr., a loop around Stanford 4 miles in circumference. To acquire this map, we drove the same loop twice, filling in gaps that are occluded by moving vehicles. 17
2.7 Map with many parked cars. Notice the absence of lane markers. 17
2.8 Infrared reflectivity ground map before and after SLAM optimization. Residual GPS drift can be seen in the ghost images of the road markings (left). After optimization, all ghost images have been removed (right). 18
2.9 Filtering dynamic objects from the map leaves holes (left). These holes are often filled if a second pass is made over the road, but ghost images remain (center). After SLAM, the hole is filled and the ghost image is removed. . . 19
2.10 Typical driving path during localization shown in green, and overlayed on a previously built map, acquired during 20 minutes of driving. For this and other paths, we find that the particle filter reliably localizes the vehicle. . . . 20
2.11 (a) GPS localization is prone to error, even (as shown here) with a high-end integrated inertial system and differential GPS using a nearby stationary antenna. (b) The particle filter result shows no noticeable error. 21
2.12 This table compares the accuracy of pose estimation in the absence of GPS or IMU data. The right column is obtained by odometry only; the center by particle filter localization relative to the map. Clearly, odometry alone accumulates error. Our approach localizes reliably without any GPS or IMU. 22
2.13 Estimated velocity from odometry and our particle filter, not using odometry or GPS. The strong correspondence illustrates that a particle filter fed with laser data alone can determine the vehicle velocity, assuming a previously acquired map. Note that in this case, accurate velocity estimation subsumes accurate position estimation. 23

3.1 An infrared refectivity map of a large urban block showing the average reflectivity of each 15x15cm cell. Due to the incorporation of all laser returns, as well as the multiple beams at different angles, this map is noticeably denser, and has fewer holes and occlusions, than those from the preceding chapter. 27

3.2 Without calibration (left), each of the 64 beams has a significantly different response to reflectivity values. After calibration (right), the beams are much better matched. 29
3.3 The two channels of our probabilistic maps. In (a) we see the average infrared reflectivity, of brightness, of each cell. The innovation of this chapter is to also consider (b), the extent to which the brightness of each cell varies. Note that the trails of the passing vehicles are much more prominent in (b). 30
3.4 Incoming laser scans (grayscale) superimposed on map (gold). (a) GPS localization is prone to error, even (as shown here) with a high-end integrated inertial system and differential GPS using a nearby stationary antenna. (b) With localization there is no noticeable error, even in the presence of large dynamic obstacles such as this passing bus. 34
3.5 Comparing the lateral offset applied by our algorithm (red) to the residual error after localization (blue) as measured by offline SLAM alignment. During these ten minutes of driving, RMS lateral error has been reduced from 66cm to 9cm. 36
3.6 Comparing the longitudinal offset applied by our algorithm (red) to the residual error after localization (blue) as measured by offline SLAM alignment. During these ten minutes of driving, RMS longitudinal error has been reduced from 87cm to 12cm. Note also the systematic bias in longitudinal error which is removed after localization. 37
3.7 An infrared reflectivity map of 11th Avenue in New York City, acquired during our autonomous demonstration at the 2008 ITS World Congress. . . 38
3.8 Junior preparing for the the 2008 ITS World Congress in Manhattan (left) and giving an autonomous demo, running the localization algorithms described in this chapter (right). 39
3.9 A four-mile loop around which our vehicle nagivated autonomously in traffic with no localization failures. 40

4.1 Our vehicle is equipped with an integrated GPS-IMU system and several LIDARs. The LIDARs used for localization are labeled above: the SICK and RIEGL scanners are used to find lane markings and the Velodyne is used to detect curbs. 43
4.2 A small RNDF: the 2007 Urban Challenge qualifying event, course "A" . . 44
4.3 A large RNDF: the entire map for the 2007 Urban Challenge race. 45
4.4 Visualization of the scanning process: the LIDAR scanner acquires range data and infrared ground reflectivity. Notice that lane markings have much higher reflectivity than pavement. 46
4.5 Lane-marker matching components: the blue curve is the lane marker prior based on the RNDF, and the green curve is the lateral response of the side lasers to the lane marker filter. Note that the RNDF suggests that the two lanes do not share a common center boundary, when in fact they do. Such inconsistencies between the RNDF and reality were common in the Urban Challenge, but our probabilistic localization algorithm was well equipped to deal with them. 48
4.6 Curb avoidance components: the orange curve is the lane corridor prior, and the pink curve is the lateral curb response function. Even though the curbs in this example are subtle, they are still detected. 49
4.7 The same region captured by a single scan (top) and by integrating several scans over time, with each point scored using the method described (bottom). By integrating scans as the vehicle moves and discounting points that fall within the concentric circles, stationary obstacles such as curbs begin to stand out relative to the ground plane. 51
4.8 Localization showing all components at once. The yellow curve is the final posterior distribution over lateral offsets. Here localization is shifting the vehicle 80 cm to the left. 55
4.9 Road without lane markers or nearby curbs; here localization relies solely on GPS, as is clear from the guassian-shaped yellow posterior distribution symmetrical about the center. 57

4.10 Road with strong lane markers; here localization deviates substantially GPS, causing the RNDF to shift by 75 cm. Without localization our car would have been driving partially into the opposing lane. 58
4.11 A histogram depicting the relative frequencies of various lateral offsets applied by the localization module during the 60-mile 2007 Urban Challege. Offsets of 0-80cm were common. 59

5.1 A single 360-degree scan from the 64-beam Velodyne LIDAR. Points are colored by height for visual clarity. 62
5.2 Our vehicle platform. Velodyne LIDAR unit is circled in red. 63
5.3 Velodyne points from two adjacent beams (of 64) accumulated over time and projected into 3D; one beam colored red, the other white. Due to known vehicle motion, both beams tend to see the same surfaces. 65
5.4 Accumulated points colored by computed surface normal; the red, green, and blue channels respectively are set to the surface's x, y, and z components of the normal vector at each point. 67
5.5 Bayesian prior for beam remittance response function as a function of surface intensity, used to initialize EM. 72
5.6 Initial Velodyne mounting position (left) and after translation and rotation (right) 75
5.7 Moving and rotating the Velodyne, but continuing to use the original position calibration, projecting accumulated Velodyne points into 3D results in massively deformed structures and significant blurring of surfaces (a). After calibration (b), the points are much better aligned as a result of the improved objective function, and the computed transform for the Velodyne is extremely accurate, as verified by hand measurements. Points are colored by both surface normal and intensity return. 76
5.8 Unsupervised horizontal angle and range calibration using 10 seconds of data (all scans depicted above). Points colored by surface normal. Even starting with an unrealistically inaccurate calibration (a) we are still able to achieve a very accurate calibration after optimization (d). 77

5.9 Comparing the horizontal angles of all 64 beams with the factory calibration, starting with a uniform initial estimate and optimizing over 400 iterations. Here, the update step size was very low for clarity. Initially beams were miscalibrated by up to 9°, and after calibration all beams' angles agreed with the factory calibration to within 1°, with an RMS deviation of only .25°. 78
5.10 Improvement in wall planarity with calibration. 79
5.11 Quantitative improvement in wall planarity during calibration procedure. Final result exceeds factory calibration for both wall and ground. 80
5.12 The expected environment intensity given each beam's intensity return. All 64 beams are shown here; note significant variation between beams. The noise in the darkest and brightest returns is due to the extremely low fraction of returns that fall in that region. 81
5.13 The learned generative model $P(a_i|m)$ for beam 37 of 64. 82
5.14 Improvement from calibration. We compare an orthographic intensity map of a street with the horrible angle and range calibration used in Fig. 5.8 (left), the same map with the learned angles and ranges (center), and finally adding intensity calibration (right). The final result is much improved. . . . 83
5.15 Using an initial estimate for the single-beam LDLRS calibration (a), the projected LDLRS points do not align well with the calibrated (and much more abundant) Velodyne points. After calibration, all scans are in good alignment, so much so that it is often difficult to spot the pink LDLRS points amongst the Velodyne points. 84
5.16 An artificial time delay between the Velodyne laser and Applanix position system results in a blurry pointcloud (a), in comparison to the correct pointcloud with no time delay (b). 85
5.17 The calibration objective function as a function of the time delay between the Velodyne and Applanix. There is a clear local minimum at .05 seconds, which is exactly the artificially induced time offset between the two sensors. 86
5.18 3D pointcloud of a campus parking lot after SLAM and calibration; 60 seconds of accumulated data colored by intensity return. 87

5.19 Closeup from above scene from another perspective; points colored by both intensity return and surface normal. 87

6.1 This figure shows two consecutive camera images overlaid with our detection grid, projected from global coordinates into the image frame. In this visualization, recorded as the light changed from red to green, grid cells most likely to contain the light are colored by their state predictions. 89
6.2 Simultaneous detection of three distinct traffic lights at one intersection. . . 91
6.3 Many traffic lights appear almost as dim as their surroundings either by design or by accident. 93
6.4 Often, a lens flare or back lighting will obscure a light's state. Using cheaper consumer-grade cameras could necessitate additional noise filtering and perhaps explicit flare detection. For example, an actuated camera could rotate itself in order to detect and remove flare. 93
6.5 (Top) A left-pointing red arrow appears in the camera image as circular, making non-contextual state detection difficult. (Bottom) The vehicle needs to know the state of the next intersection far in advance so appropriate actions are taken. In this case, 200mm green lights appear far apart among a cluttered scene with many light sources. 94
6.6 Several random variables and their uncertainties influence the accuracy of the localization result. This diagram shows the dependencies of the different coordinate frames: [W]orld, [V]ehicle, [C]amera and [O]bject (here: traffic light). 96
6.7 Hue histograms $\mathbf{H}_{red,yellow,green}$ are generated from individual lights at various distances, captured during a pre-drive. 100

6.8 (Left) is a visualization constructed by merging $\{G_{red}, G_{green}, G_{yellow}\}$ as the *red*, *green* and *blue* channels of an image. Color saturation corresponds to a higher certainty of a specific state in the current frame (before the prior probability is processed). (Center) depicts the detection grid image P, in which each cell's color represents a grid cell's predicted state and its intensity represents that state's score. (Right) is the resulting prior as described in Section 6.3.2. Each light captured in figure 6.2 is shown from top to bottom. 102
6.9 Correct detection rates of individual lights and complete intersections from noon are depicted. 103
6.10 Correct detection rates of individual lights and complete intersections from sunset are depicted. 104
6.11 Correct detection rates of individual lights and complete intersections from night are depicted. 104
6.12 Combined results from noon, sunset and night are depicted. 104
6.13 Considering multiple lights with separate search windows (here, shown in blue) improves robustness. In the top frame, the window on the left reports a yellow light with 38% probability and the window on the right reports a red light with 53% probability. The final state decision of the intersection, taking into account probabilities for red, yellow and green states from both windows, yields (correctly) a red state with probability 53%. In the bottom frame, probabilities are 80% and 48% for the left and right lights and 87% for the intersection, whose state is correctly detected as red. 106
6.14 Our belief that a given light has a particular state versus the accuracy of that belief, using raw output from histogram filters. Although the filter is generally overconfident, its certainty does correlate strongly with accuracy, as desired. 106

6.15 Confusion matrix of light detections. Entries are number of detections. Each row and column has an associated percent accuracy. Yellow lights are more easily confused due to their shared hues with red lights and because it is more difficult to obtain a large training set for relatively rare yellow lights as compared to red and green lights. 106
6.16 Confusion matrix of intersection decisions. Entries are number of decisions. Each row and column has an associated percent accuracy. 107
6.17 Relatively large search windows necessitated by standard GPS without laser localization 109
6.18 Search windows are reduced in area by 90% thanks to laser localization . . 109
6.19 Assuming the camera was mounted exactly in line with the vehicle frame, even the best estimated mapping locations of the traffic lights do not match well with their observed locations across all frames. 111
6.20 Using the optimized extrinsic camera calibration transform, the new estimated mapping locations of the lights gives estimates in image coordinates which track the lights much more accurately across all frames. 112

A.1 Junior senses the environment with laser rangefinders (a, b, c, h), cameras (d, e), radars (f), and a positioning system with an inertial measurement unit and GPS receivers (h). 118
A.2 Virtual orthographic camera intensity images from three tracks of objects that Junior can recognize. Depth segmentation of objects provides invariance to background clutter. Classification of depth data allows one to put objects in a canonical orientation, thus providing some measure of perspective invariance. 120
A.3 Smooth trajectory set: The z axis shows the velocity; overall costs are indicated by the color. 124

Chapter 1

Introduction

The concept of self-driving robotic vehicles has taken a significant leap from fantasy towards reality over the past several years. Our lab has played an integral role, from Stanford's contributions to the 2005 DARPA Grand Challenge, which featured autonomous cars driving through harsh desert terrain [12, 85, 60], to the 2007 DARPA Urban Challenge, which added dynamic traffic and real California traffic laws [13, 61], and to Google's new robot car project, which builds off a large part of our research [80].

Each year, more than a million human lives are lost due to traffic accidents, and over 20 million people suffer vehicle-related injuries [90]. An overwhelming majority of these are due to human error, and thus could theoretically be avoided by robotic vehicles. In addition to reducing deaths and injuries, autonomous vehicles could provide enhanced mobility for children and adults unable to drive, free up billions of hours of otherwise unproductive commuting time, and revolutionize parking in cities.

An autonomous vehicle requires a significant amount of hardware and software not found in a regular car. Most basically, this includes a drive-by-wire system for actuating vehicle controls such as throttle, brake, and steering, sensors for perceiving the environment, and artificial intelligence algorithms to process the sensor data and make safe decisions about how to control the vehicle in response to the environment. An overview of our hardware and software approach to these topics is included in the Appendix of this document; for further information, please see [61] and [56].

This dissertation presents several algorithms that realize key enabling components of

Figure 1.1: Junior was the first robot to cross the finish line in the 2007 DARPA Urban Challenge, winning second place overall.

autonomous vehicles. We discuss several methods for mapping dynamic urban environments and localizating a vehicle with centimeter-level accuracy in realtime relative to these maps [52, 53]. We present algorithms for automatically calibrating multi-beam sensors to a moving vehicle with accuracy that exceeds that of tedious hand measurements [54]. Finally, we discuss an end-to-end pipeline for mapping traffic lights and subsequently detecting their state in realtime [55].

Chapter 2 presents a system for generating high-resolution laser reflectivity maps of urban environments, and for localizing a vehicle in realtime to those maps. Many urban navigation applications, including autonomous navigation and driver assistance systems, can benefit greatly from localization with centimeter accuracy. Yet such accuracy cannot be achieved reliably with GPS-based inertial guidance systems, especially in urban settings. Our approach integrates GPS, IMU, wheel odometry, and LIDAR data acquired by an instrumented vehicle, to generate high-resolution environment maps. To localize a moving vehicle relative to these maps, we present a particle filter method for correlating LIDAR measurements with this map. As we show by experimentation, the resulting relative accuracies exceed that of conventional GPS-IMU-odometry-based methods by more than an

order of magnitude.

Chapter 3 presents an extension of the approach derived in Chapter 2. Using multi-beam lasers, we generate reflectivity maps that incorporate probabilistic intensity returns at each grid cell, thereby significantly increasing the expressivity of the maps, adding the ability to learn and improve maps over time, and enabling more robust and accurate localization in the face of environment changes and dynamic obstacles. We validate the effectiveness of our approach by using these algorithms to localize our vehicle against probabilistic maps in various dynamic environments, achieving RMS accuracy in the 10cm-range and thus outperforming previous work. Importantly, this approach has enabled us to autonomously drive our vehicle for hundreds of miles in dense traffic on narrow urban roads which were formerly unnavigable with previous localization methods.

Chapter 4 presents a method of laterally localizing a vehicle to a compact vector road map, in contrast to the high-resolution maps discussed in the previous chapters. Our approach simultaneously minimizes GPS shift and the presence of curbs within lanes while maximizing the matching of lane markers on the ground to those implicit in the map. By combining all three of these constraints into a single objective function, we are able to systematically determine the most probable lateral offset of the vehicle relative to the map in real time. We demonstrate the success of our approach as an integral component of an end-to-end autonomous vehicle capable of navigating in dynamic urban environments. In the 60-mile 2007 DARPA Urban Challenge, this localization method successfully enabled our vehicle to travel in the center of the lanes and complete the course, providing an RMS lateral correction of 40 cm relative to GPS, with occasional corrections of up to 130 cm, and no localization failures.

Chapter 5 presents algorithms for automatically calibrating multi-beam laser rangefinders on a moving vehicle platform, provided the vehicle has a sensor that measures local or intertial motion updates. Using these techniques, we are able to derive the precise mounting location and orientation of sensors on our vehicle from only seconds of driving, to an accuracy which exceeds our ability to measure the transform by hand. In addition, we can calculate intrinsic beam parameters, align multiple beams within a sensor or separate sensors to each other, and recover time delays between sensors. Crucially, our approach requires no specific calibration target, instead relying only on the weak assumption that

points in space tend to lie on contiguous surfaces.

Chapter 6 presents a collection of algorithms for mapping the location of traffic lights and for tracking their location in realtime and detecting their state. First, we introduce a convenient technique for mapping traffic light locations from recorded video data using tracking, back-projection, and triangulation. In order to achieve robust real-time detection results in a variety of lighting conditions, we combine several probabilistic stages that explicitly account for the corresponding sources of sensor and data uncertainty. In addition, our approach is the first to account for multiple lights per intersection, which yields superior results by probabilistically combining evidence from all available lights. To evaluate the performance of our method, we present several results across a variety of lighting conditions in a real-world environment. These techniques have allowed us to autonomously nagivate our vehicle through traffic-light-controlled intersections.

Although all of the algorithms are discussed primarily in the context of autonomous driving, most of them are fundamental building blocks for robots that move about and perceive their environment, and thus they apply equally to a wide variety of mobile robotics platforms and scenarios. In particular, the mapping and localization algorithms are applicable to many types of outdoor robots, and the automatic calibration algorithms are useful for multi-beam lasers attached to almost any robot.

Unifying all the methods presented in this dissertation are probabilistically derived explanations for the way sensors perceive the environment; from these first principles, and thoughtfully chosen optimization objectives, we arrive at straightforward but elegant solutions for challenging problems.

Chapter 2

Mapping and Localization

2.1 Introduction

Many urban navigation applications (e.g., autonomous navigation, driver assistance systems) can benefit greatly from localization with centimeter accuracy. For example, in order for an autonomous robot to follow a road, it needs to know where the road is. To stay in a specific lane, it needs to know where the lane is. For an *autonomous* robot to stay in a lane, the localization requirements are in the order of decimeters. As became painfully clear in the 2004 DARPA Grand Challenge [85], GPS alone is insufficient and does not meet these requirements. There, the leading robot failed after driving off the road because of GPS error.

The core idea of this chapter is to augment inertial navigation by learning a detailed map of the environment, and then to use a vehicle's LIDAR sensor to localize relative to this map. In the present chapter, maps are 2-D overhead views of the road surface, taken in the infrared spectrum. Such maps capture a multitude of textures in the environment that may be useful for localization, such as lane markings, tire marks, pavement, and vegetation near the road (e.g., grass). The maps are acquired by a vehicle equipped with a state-of-the-art inertial navigation system (with GPS) and multiple SICK laser range finders.

The problem of environment mapping falls into the realm of SLAM (Simultaneous Localization and Mapping), hence there exists a large body of related work on which our

Figure 2.1: The acquisition vehicle is equipped with a tightly integrated inertial navigation system which uses GPS, IMU, and wheel odometry for localization. It also possesses laser range finders for road mapping and localization.

approach builds [9, 22, 23, 51, 47, 65]. SLAM addresses the problem of building consistent environment maps from a moving robotic vehicle, while simultaneously localizing the vehicle relative to these maps. SLAM has been at the core of a number of successful autonomous robot systems [2, 78].

At first glance, building urban maps is significantly easier than SLAM, thanks to the availability of GPS most of the time. While GPS is not accurate enough for mapping—as we show in this chapter—it nevertheless bounds the localization error, thereby sidestepping hard SLAM problems such as the loop closure problem with unbounded error [33, 35, 82]. However, a key problem rarely addressed in the SLAM literature is that of dynamic environments. Urban environments are dynamic, and for localization to succeed one has to distinguish static aspects of the world (e.g., the road surface) relative to dynamic aspects (e.g., cars driving by). Previous work on SLAM in dynamic environments [32, 34, 87] has mostly focused on tracking or removing objects that move at the time of mapping, which might not be the case here.

Our approach addresses the problem of environment dynamics, by reducing the map to features that with very high likelihood are static. In particular, using 3-D LIDAR information, our approach only retains the flat road surface; thereby removing the imprints of potentially dynamic objects (e.g., other cars, even if parked). The resulting map is then simply an overhead image of the road surface, where the image brightness corresponds to the infrared reflectivity.

Once a map has been built, our approach uses a particle filter to localize a vehicle in

Figure 2.2: Visualization of the scanning process: the LIDAR scanner acquires range data *and* infrared ground reflectivity. The resulting maps therefore are 3-D infrared images of the ground reflectivity. Notice that lane markings have much higher reflectivity than pavement.

real-time [16, 19, 58, 68]. The particle filter analyzes range data in order to extract the ground plane underneath the vehicle. It then correlates via the *Pearson product-moment correlation* [66] the measured infrared reflectivity with the map. Particles are projected forward through time via the velocity outputs from a tightly-coupled inertial navigation system, which relies on wheel odometry, an IMU and a GPS system for determining vehicle velocity. Empirically, we find that the our system very reliably tracks the location of the vehicle, with relative accuracy of $\sim$10 cm. A dynamic map management algorithm is used to swap parts of maps in and out of memory, scaling precision localization to environments too large to be held in main memory.

Both the mapping and localization processes are robust to dynamic and hilly environments. Provided that the local road surface remains approximately laterally planar in the neighborhood of the vehicle (within $\sim$10m), slopes do not pose a problem for our algorithms.

2.2 Road Mapping with GraphSLAM

Our mapping method is a version of the GraphSLAM algorithm and related constraint relaxation methods [4, 21, 25, 27, 43, 47, 81, 44, 46]. Similar to other GraphSLAM approaches is our incorporation of odometry constraints between consectutive poses and map-matching constraints between temporally separated but spatially near poses. Unique to our approach is the incorporation of a GPS measurement model into the optimization and the filtering out of obstacles which may otherwise cause disagreements in the map.

2.2.1 Modeling Motion

The vehicle transitions through a sequence of poses. In urban mapping, poses are five-dimensional vectors, comprising the x-y coordinates of the robot, along with its heading direction (yaw) and the roll and pitch angle of the vehicle (the elevation z is irrelevant for this problem). Let x_t denote the pose at time t. Poses are linked together through relative odometry data, acquired from the vehicle's inertial guidance system.

$$x_t = g(u_t, x_{t-1}) + \varepsilon_t \tag{2.1}$$

Here g is the non-linear kinematic function which accepts as input a pose x_{t-1} and a motion vector u_t, and outputs a projected new pose x_t. The variable ε_t is a Gaussian noise variable with zero mean and covariance R_t. Vehicle dynamics are discussed in detail in [30].

In log-likelihood form, each motion step induces a nonlinear quadratic constraint of the form

$$(x_t - g(u_t, x_{t-1}))^T \; R_t^{-1} \; (x_t - g(u_t, x_{t-1})) \tag{2.2}$$

which penalizes consecutive poses in which the distance between the poses disagrees with the corresponding odometry measurement. As in [21, 81], these constraints can be thought of as edges in a sparse Markov graph.

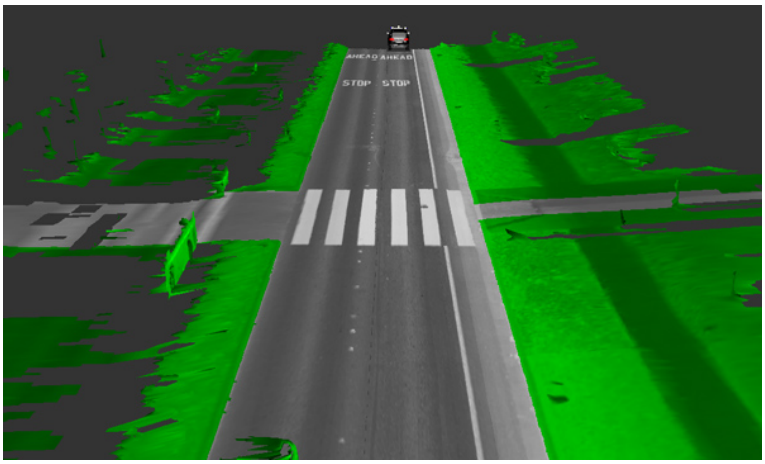


Figure 2.3: Example of ground plane extraction. Only measurements that coincide with the ground plane are retained; all others are discarded (shown in green here). As a result, moving objects such as car (and even parked cars) are not included in the map. This makes our approach robust in dynamic environments.

2.2.2 Map Representation

The map is a 2-D grid which assigns to each *x*-*y* location in the environment an infrared reflectivity value. Thus, we can treat the ground map as a orthographic infrared photograph of the ground.

To acquire such a map, multiple laser range finders are mounted on a vehicle, pointing downwards at the road surface (see Fig. 2.1). In addition to returning the range to a sampling of points on the ground, these lasers also return a measure of infrared reflectivity. By texturing this reflectivity data onto the 3-D range data, the result is a dense infrared reflectivity image of the ground surface, as illustrated in Fig. 2.2 (this map is similar to vision-based ground maps in [83]).

To eliminate the effect of non-stationary objects in the map on subsequent localization, our approach fits a ground plane to each laser scan, and only retains measurements that coincide with this ground plane. The ability to remove vertical objects is a key advantage of using LIDAR sensors over conventional cameras. As a result, only the flat ground is mapped, and other vehicles are automatically discarded from the data. Fig. 2.3 illustrates this process. The color labeling in this image illustrates the data selection: all green areas protrude above the ground plane and are discarded. Maps like these can be acquired even at night, as the LIDAR system does not rely on external light. This makes the mapping

result much less dependent on ambient lighting than is the case for passive cameras.

For any pose x_t, map m, and any (fixed and known) laser angle relative to the vehicle coordinate frame α_i, the expected infrared reflectivity can easily be calculated. Let $h_i(m,x_t)$ be this function, which calculates the expected laser reflectivity for a given map m, a robot pose x_t, and a laser angle α_i. We model the observation process as follows

$$z_t^i = h_i(m,x_t) + \delta_t^i \tag{2.3}$$

Here δ_t^i is a Gaussian noise variable with mean zero and noise covariance Q_t.

In log-likelihood form, this provides a new set of constraints, which are of the form

$$(z_t^i - h_i(m,x_t))^T \; Q_t^{-1} \; (z_t^i - h_i(m,x_t))$$

The unknowns in this function are the poses $\{x_t\}$ and the map m.

2.2.3 Latent Variable Extension for GPS

In outdoor environments, a vehicle can use GPS for localization. GPS offers the convenient advantage that its error is usually limited to a few meters. Let y_t denote the GPS signal for time t. (For notational convenience, we treat y_t as a 3-D vector, with yaw estimate simply set to zero and the corresponding noise covariance in the measurement model set to infinity).

At first glance, one might integrate GPS through an additional constraint in the objective function J. The resulting constraints could be of the form

$$\sum_t (x_t - y_t)^T \; \Gamma_t^{-1} \; (x_t - y_t) \tag{2.4}$$

where Γ_t is the noise covariance of the GPS signal. However, this form assumes that GPS noise is *independent.* In practice, GPS is subject to *systematic* noise. Because GPS is affected through atmospheric properties, which tend to change slowly with time.

Our approach models the systematic nature of the noise through a Markov chain, which

uses GPS bias term b_t as a latent variable. The assumption is that the actual GPS measurement is corrupted by an additive bias b_t, which cannot be observed (hence is latent but can be inferred from data). This model yields constraints of the form

$$\sum_t (x_t - (y_t + b_t))^T \; \Gamma_t^{-1} \; (x_t - (y_t + b_t)) \tag{2.5}$$

In this model, the latent bias variables b_t are subject to a random walk of the form

$$b_t \;\; = \;\; \gamma \, b_{t-1} + \beta_t \tag{2.6}$$

Here β_t is a Gaussian noise variable with zero mean and covariance S_t. The constant $\gamma < 1$ slowly pulls the bias b_t towards zero.

2.2.4 The Extended GraphSLAM Objective Function

Putting this all together, we obtain the goal function

$$\begin{aligned} J \;\; = \;\; & \sum_t (x_t - g(u_t, x_{t-1}))^T \; R_t^{-1} \; (x_t - g(u_t, x_{t-1})) \\ & + \sum_{t,i} (z_t^i - h_i(m, x_t))^T \; Q_t^{-1} \; (z_t^i - h_i(m, x_t)) \\ & + \sum_t (x_t - (y_t + b_t))^T \; \Gamma_t^{-1} \; (x_t - (y_t + b_t)) \\ & + \sum_t (b_t - \gamma b_{t-1})^T \; S_t^{-1} \; (b_t - \gamma b_{t-1}) \end{aligned} \tag{2.7}$$

Unfortunately, this expression cannot be optimized directly, since it involves many millions of variables (poses and map pixels).

2.2.5 Integrating Out the Map

Unfortunately, optimizing J directly is computationally infeasible. A key step in GraphSLAM, which we adopt here, is to first integrate out the map variables. In particular, instead of optimizing J over all variables $\{x_t\}$, $\{b_t\}$, and m, we first optimize a modified version of J that contains only poses $\{x_t\}$ and biases $\{b_t\}$, and then compute the most likely map.

This is motivated by the fact that, as shown in [81], the map variables can be integrated out in the SLAM joint posterior. Since nearly all unknowns in the system are with the map, this simplification makes the problem of optimizing J much easier and allows it to be solved efficiently.

In cases where a specific surface patch is only seen once during mapping, our approach can simply ignore the corresponding map variables during the pose alignment process, because such patches have no bearing on the pose estimation. Consequently, we can safely remove the associated constraints from the goal function J, without altering the goal function for the poses.

Of concern, however, are places that are seen more than once. Those *do* create constraints between pose variables from which those places were seen. These constraints correspond to the famous loop closure problem in SLAM [33, 35, 82]. To integrate those map variables out, our approach uses an effective approximation known as map matching [43]. Map matching compares local submaps, in order to find the best alignment.

Our approach implements map matching by first identifying regions of overlap, which will then form the local maps. A region of overlap is the result of driving over the same terrain twice. Formally it is defined as two disjoint sequences of time indices, $t_1, t_2, \ldots$ and $s_1, s_2, \ldots$, such that the corresponding grid cells in the map show an overlap that exceeds a given threshold θ.

Once such a region is found, our approach builds two separate maps, one using only data from $t_1, t_2, \ldots$, and the other only with data from $s_1, s_2, \ldots$. It then searches for the alignment that maximizes the measurement probability, assuming that both adhere to a single maximum likelihood infrared reflectivity map in the area of overlap.

Specifically, a normalized cross-correlation field is computed between these maps, for different x-y offsets between these images. Since the poses prior to alignment have already been post-processed from GPS and IMU data, we find that rotational error between matches is insignificant. Because one or both maps may be incomplete, our approach only computes correlation coefficients from elements whose infrared reflectivity value is known. In cases where the alignment is unique, we find a single peak in this correlation field. The peak of this correlation field is then assumed to be the best estimate for the local alignment. The relative shift is then labeled δ_{st}, and the resulting constraint of the form above is added to

the objective J.

This map matching step leads to the introduction of the following constraint in J:

$$(x_t + \delta_{st} - x_s)^T \; L_{st} \; (x_t + \delta_{st} - x_s) \tag{2.8}$$

Here δ_{st} is the local shift between the poses x_s and x_t, and L_{st} is the strength of this constraint (an inverse covariance). Replacing the map variables in J with this new constraint is clearly approximate; however, it makes the resulting optimization problem tractable.

The new goal function J' then replaces the many terms with the measurement model by a small number of between-pose constraints. It is of the form:

$$\begin{aligned} J' \;\; = \;\; & \sum_t (x_t - g(u_t, x_{t-1}))^T \; R_t^{-1} \; (x_t - g(u_t, x_{t-1})) \\ & + \sum_t (x_t - (y_t + b_t))^T \; \Gamma_t^{-1} \; (x_t - (y_t + b_t)) \\ & + \sum_t (b_t - \gamma b_{t-1})^T \; S_t^{-1} \; (b_t - \gamma b_{t-1}) \\ & + \sum_t (x_t + \delta_{st} - x_s)^T \; L_{st} \; (x_t + \delta_{st} - x_s) \end{aligned} \tag{2.9}$$

This function contains no map variables, and thus there are orders of magnitude fewer variables in J' than in J. J' is then easily optimized using the conjugate gradient method. For the type of maps shown in this chapter, the optimization takes significantly less time on a PC than is required for acquiring the data. Most of the time is spent in the computation of local map links; the optimization of J' only takes a few seconds. The result is an adjusted robot path that resolves inconsistencies at intersections.

2.2.6 Computing the Map

To obtain the map, our approach needs to simply fill in all map values for which one or more measurements are available. In grid cells for which more than one measurement is available, the average infrared reflectivity minimizes the joint data likelihood function.

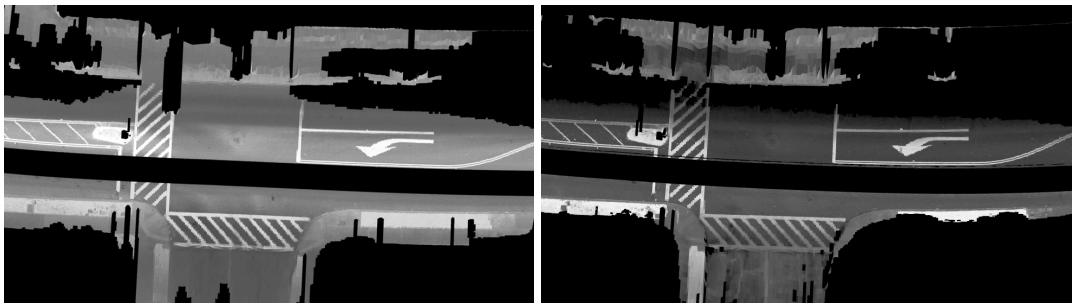


Figure 2.4: Patch of the map acquired in bright sunlight on a sunny day (left), and at night in heavy rain (right). By correlating scans with the map, instead of taking absolute differences, the weather-related brightness variation has almost no effect on localization.

We notice that this is equivalent to optimizing the missing constraints in J, denoted J'':

$$J'' = \sum_{t,i} (z_t^i - h_i(m, x_t))^T \; Q_t^{-1} \; (z_t^i - h_i(m, x_t)) \tag{2.10}$$

under the assumption that the poses $\{x_t\}$ are known. Once again, for the type of maps shown in this chapter, the computation of the aligned poses requires only a few seconds.

When rendering a map for localization or display, our implementation utilizes hardware-accelerated OpenGL to render smoothly interpolated polygons whose vertices are based on the distances and intensities returned by the three lasers as the robot traverses its trajectory. Even with a low-end graphics card, this process is faster than real-time.

2.3 Online Localization

2.3.1 Localization with Particle Filters

Localization uses the map in order to localize the vehicle relative to the map. Localization takes place in real-time, with a 200 Hz motion update (measurements arrive at 75 Hz per laser).

Our approach utilizes the same mathematical variable model discussed in the previous section. However, to achieve real-time performance, we utilize a particle filter, known in robotics as Monte Carlo localizer [15]. The particle filter maintains a three-dimensional pose vector (x, y, and yaw); roll and pitch are assumed to be sufficiently accurate as is. The

motion prediction in the particle filter is based on inertial velocity measurements, as stated in Sect. 2.2.1.

Measurements are integrated in the usual way, by affecting the importance weight that sets the resampling probability. As in the mapping step, a local ground plane analysis removes measurement that correspond to non-ground objects. Further, measurements are only incorporated for which the corresponding map is defined, that is, for which a prior reflectivity value is available in the map.

For each measurement update, the importance weight of the k-th particle is the product of its likelihood according to the GPS measurement model (a Gaussian) and the laser measurement model (a correlation, which accounts for systematic brightness changes between the map and the sensor data). The particle weights are therefore computed as:

$$w_t^{[k]} = \exp\{-\frac{1}{2}\,(x_t^{[k]}-y_t)^T\,\Gamma_t^{-1}\,(x_t^{[k]}-y_t)\}\;\cdot \qquad (2.11)$$
$$\left(\operatorname{corr}\left[\begin{pmatrix} h_1(m,x_t^{[k]}) \\ \vdots \\ h_{180}(m,x_t^{[k]}) \end{pmatrix}, \begin{pmatrix} z_t^1 \\ \vdots \\ z_t^{180} \end{pmatrix}\right] + 1\right)$$

where corr() is the Pearson product-moment correlation[66].

We resample the particles at a much lower frequency than the measurement/pose updates, to reduce variance [79]. Thus, as GPS and laser readings are processed, the corresponding measurement scores for each particle are integrated multiplicatively into the particle weights, and as odometry measurements are processed, particles are moved according to the motion model. Then, particles are resampled once per second, such that particles with higher weights are probabilistically chosen to survive.

To avoid catastrophic localization errors, a small number of particles are continuously drawn from current GPS pose estimate. This "sensor resetting" trick was proposed in [50].

One complicating factor in vehicle localization is weather. As illustrated in Fig. 2.4, the appearance of the road surface is affected by rain, in that wet surfaces tend to reflect less infrared laser light than do dry ones. To adjust for this effect, the particle filter normalizes the brightness and standard deviation for each individual range scan, and also for the corresponding local map stripes. This normalization transforms the least squares difference method into the computation of the Pearson product-moment correlation with missing

Figure 2.5: Aerial view of Burlingame, CA. Regions of overlap have been adjusted according to the methods described in this chapter. Maps of this size tend not to fit into main memory, but are swapped in from disk automatically during driving.

variables (empty grid cells in the map). Empirically, we find that this local normalization step is essential for robust performance .

We also note that the particle filter can be run entirely without GPS, where the only reference to the environment is the map. In this case, we simply omit the GPS term from the measurement model. Some of our experiments are carried out in the absence of GPS, to illustrate the robustness of our approach in situations where conventional GPS-based localization is plainly inapplicable.

2.3.2 Data Management

Maps of large environments at 5-cm resolution occupy a significant amount of memory. We have implemented two methods to reduce the size of the maps and to allow relevant data to fit into main memory.

When acquiring data in a moving vehicle, the rectangular area which circumscribes the resulting laser scans grows quadratically with distance, despite that the data itself grows only linearly. In order to avoid a quadratic space requirement, we break the rectangular area into a square grid, and only save squares for which there is data. When losslessly

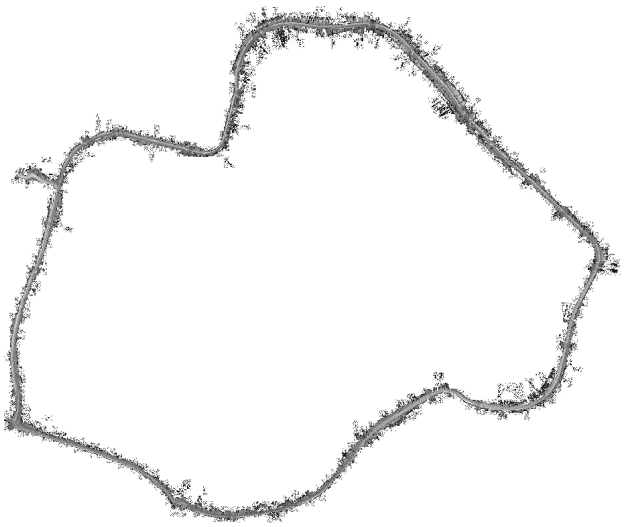


Figure 2.6: Campus Dr., a loop around Stanford 4 miles in circumference. To acquire this map, we drove the same loop twice, filling in gaps that are occluded by moving vehicles.

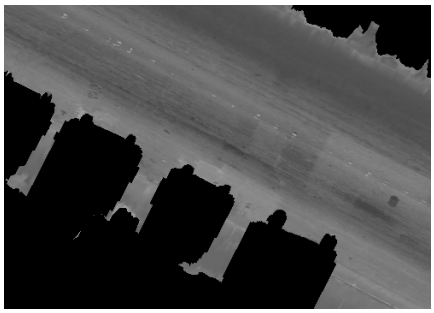


Figure 2.7: Map with many parked cars. Notice the absence of lane markers.

compressed, the grid images require approximately 10MB per mile of road at 5-cm resolution. This would allow a 200GB hard drive to hold 20,000 miles of data.

Although the data for a large urban environment can fit on hard drive, it may not all be able to fit into main memory. Our particle filter maintains a cache of image squares near the vehicle, and thus requires a constant amount of memory regardless of the size of the overall map.

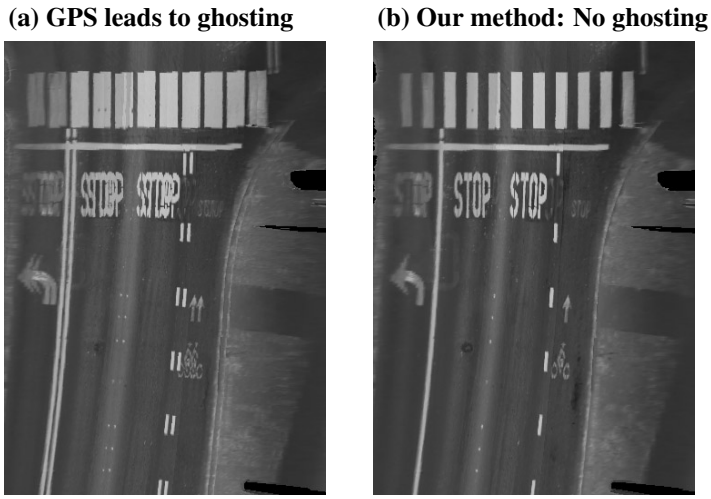


Figure 2.8: Infrared reflectivity ground map before and after SLAM optimization. Residual GPS drift can be seen in the ghost images of the road markings (left). After optimization, all ghost images have been removed (right).

2.4 Experimental Results

We conducted extensive experiments with the vehicle shown in Fig. 2.1. This vehicle is equipped with a state-of-the-art inertial navigation system (GPS, inertial sensors, and wheel odometry), and three roof-mounted, down-facing laser rangefinders: one facing the left side, one facing the right side, and one facing the rear. In all experiments, we use a pixel resolution of 5cm.

2.4.1 Mapping

We tested the mapping algorithm successfully on a variety of urban roads. One of our testing environments is an urban area, shown in Fig. 2.5. This image shows an aerial view of the testing environment, with the map overlayed in red. To generate this map, our algorithm automatically identified and aligned several hundreds match points in a total of 32 loops. It corrected the trajectory and output consistent imagery at 5-cm resolution. Fig. 2.7 shows a close-up of this map in a residential area lined with parked cars, whose positions changed between different data runs.

One of the key results of the map alignment process is the removal of the so-called

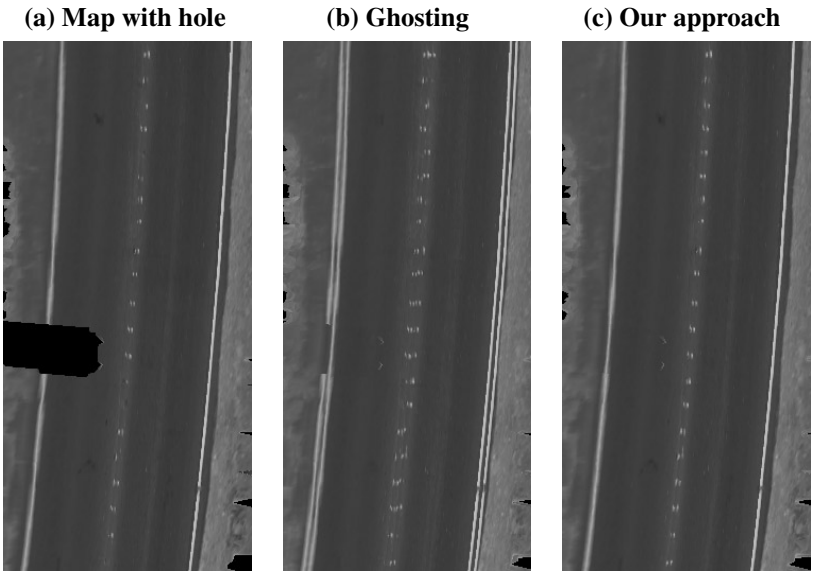


Figure 2.9: Filtering dynamic objects from the map leaves holes (left). These holes are often filled if a second pass is made over the road, but ghost images remain (center). After SLAM, the hole is filled and the ghost image is removed.

"ghosting" effects. Ghosting occurs in the absence of the map alignment step, where the location of measurements are set by GPS alone. Fig. 2.8a shows an example of ghosting, in which features occur twice in the map. Clearly, such a map is too inaccurate for precision localization. Fig. 2.8b shows the adjusted map, generated with the approach described in this chapter. Here the ghosting effect has disappeared, and the map is locally consistent.

Figure 2.6 shows a large loop which is 4 miles in circumference. Our algorithm automatically identified and aligned 148 match points during the second traversal of the loop. In this experiment, the average magnitude of the corrections calculated by the matching algorithm for these two loops is 35 cm, indicating that without corrections, the two loops are quite poorly aligned, even with post-processed poses.

An interesting result is that areas missing in the first traversal can be filled in during the second. Fig. 2.9 shows three local maps. The map from the first traversal has a hole caused by oncoming traffic. The center map in this figure is the GPS aligned map with the familiar ghosting effect. On the right is the fused map with our alignment procedure, which possesses no hole and exhibits no ghosting.

The examples shown here are representative of all the maps we have acquired so far.

Figure 2.10: Typical driving path during localization shown in green, and overlayed on a previously built map, acquired during 20 minutes of driving. For this and other paths, we find that the particle filter reliably localizes the vehicle.

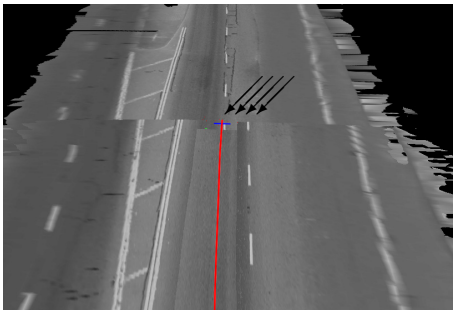
We find that even in dense urban traffic, the alignment process is robust to other cars and non-stationary obstacles. No tests were conducted in open featureless terrain (e.g., airfields without markings or texture), as we believe those are of low relevance for the stated goal of urban localization.

2.4.2 Localization

Our particle filter is adaptable to a variety of conditions. In the ideal case, the vehicle contains an integrated GPS/IMU system, which provides locally consistent velocity estimates and global pose accuracy to within roughly a meter. In our tests in a variety of urban roads, our GPS-equipped vehicle was able to localize in real-time relative to our previously created maps with errors of less than 10 cm, far exceeding the accuracy with GPS alone.

We ran a series of experiments to test localization relative to the learned map. All localization experiments used separate data from the mapping data; hence the environment was subject to change that, at times, was substantial. In fact, the map in Fig. 2.5 was acquired at night, but all localization experiments took place during daylight. In our experimentation, we used between 200 and 300 particles.

(a) GPS localization induces $\geq$1 meter of error.



(b) No noticeable error in particle filter localization.

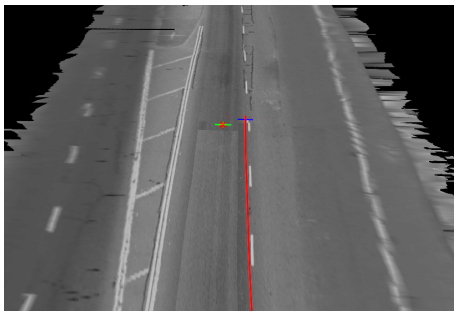


Figure 2.11: (a) GPS localization is prone to error, even (as shown here) with a high-end integrated inertial system and differential GPS using a nearby stationary antenna. (b) The particle filter result shows no noticeable error.

Fig. 2.10 shows an example path that corresponds to 20 minutes of driving and localization. During this run, the average disagreement between real-time GPS pose and our method was 66cm. Manual alignment of the map and the incoming LIDAR measurements suggests that the lateral error was almost always within 10cm. Occasionally, errors were larger; when turning, errors were sometimes as large as 30cm. However, this is still well below the error rate of the GPS-based inertial system.

A typical situation is shown in Fig. 2.11. Fig. 2.11a superimposes data acquired online by the vehicle and the previously acquired map. The error here is greater than one meter. Fig. 2.11b shows the result with our approach, where the error is below the map resolution. The red line in both figures corresponds to the path of the localizing vehicle.

One of the key aspects of the localization method is its ability to localize entirely in the absence of GPS. For the following experiments, we switched off the GPS receiver, and exclusively used wheel odometry and steering angle for localization. Clearly, odometry

Stanford Ave.		
Distance Traveled (m)	Our Error (cm)	Odometry Error (cm)
50	7	98
100	3	149
150	35	0
200	13	8
250	4	133
300	22	272
350	8	428
400	23	589
450	13	783
499	10	924

Figure 2.12: This table compares the accuracy of pose estimation in the absence of GPS or IMU data. The right column is obtained by odometry only; the center by particle filter localization relative to the map. Clearly, odometry alone accumulates error. Our approach localizes reliably without any GPS or IMU.

alone eventually diverges from the true position. Nevertheless, our localization method reliable tracks the location of the car.

This is illustrated by the result in Fig. 2.12, which quantitatively compares the accuracy of the particle filter localizer with odometry data. In both cases, the filters are initialized with the correct pose. Error is measured along 10 hand-labeled reference points in 50 meter increments.

Not surprisingly, the odometry estimate's error grows quickly as the vehicle moves, whereas our method results in a small error. The error is slightly larger than in the experiments above, due to the absence of GPS in the forward projection step of the particles. Nevertheless, this experiment illustrates successful tracking in the absence of GPS.

Finally, an extreme test of our software's ability to localize using laser data was performed by disallowing any motion or position data whatsoever in the motion update. Thus odometry, IMU, and GPS measurements were all ignored. In this case, the particles' state vector included x and y position, yaw, steering angle, velocity, and acceleration. The particles were initialized near the true position, and reasonable values were assumed for the rate of change of the control parameters. Remarkably, our system was able to track the position and the velocity of the vehicle using nothing but laser data (see Fig. 2.13).

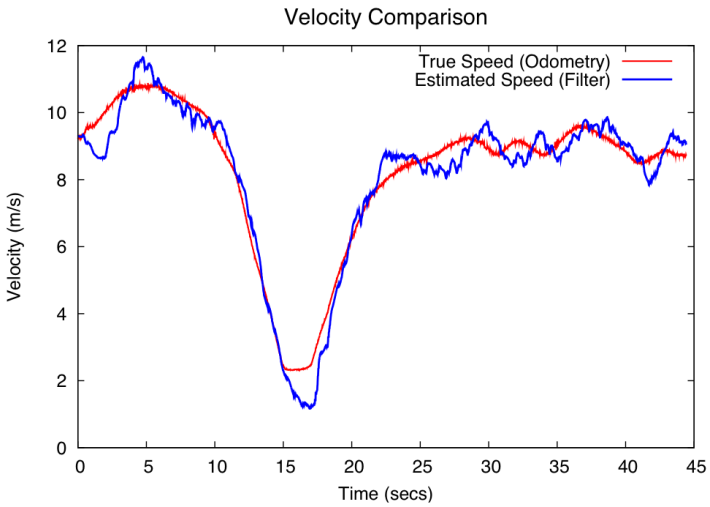


Figure 2.13: Estimated velocity from odometry and our particle filter, not using odometry or GPS. The strong correspondence illustrates that a particle filter fed with laser data alone can determine the vehicle velocity, assuming a previously acquired map. Note that in this case, accurate velocity estimation subsumes accurate position estimation.

2.4.3 Autonomous Driving

In a series of experiments, we used our approach for semi-autonomous urban driving. In most of these experiments, gas and brakes were manually operated, but steering was controlled by a computer. The vehicle followed a fixed reference trajectory on a university campus. In some cases, the lane width exceeded the vehicle width by less than 2 meters.

Using the localization technique described here, the vehicle was able to follow this trajectory on ten out of ten attempts without error; never did our method fail to provide sufficient localization accuracy. Similar experiments using only GPS consistently failed within meters, illustrating that GPS alone is insufficient for autonomous urban driving. We view the ability to use our localization method to steer a vehicle in an urban environment as a success of our approach.

2.5 Conclusion

Precision localization is a key enabling factor for urban robotic operation. With accurate and precise localization, autonomous cars can perform accurate lane keeping and obey

traffic laws (e.g., stop at stop signs). Although nearly all outdoor localization work is based on GPS, GPS alone is insufficient to provide the accuracy necessary for urban autonomous operation.

This chapter presented a localization method that uses an environment map. The map is acquired by an instrumented vehicle that uses infrared LIDAR to measure the 3-D structure and infrared reflectivity of the environment. A SLAM-style relaxation algorithm transforms this data into a globally consistent environment model with non-ground objects removed. A particle filter localizer was discussed that enables a moving vehicle to localize in real-time, at 200 Hz.

Extensive experiments in urban environments suggest that the localization method surpasses GPS in two critical dimensions: accuracy and availability. The proposed method is significantly more accurate than GPS; its relative lateral localization error to the previously recorded map is mostly within 5cm, whereas GPS errors are often in the order of 1 meter. Second, our method succeeds in GPS-denied environments, where GPS-based systems fail. This is of importance to navigation in urban canyons, tunnels, and other structures that block or deflect GPS signals.

The biggest disadvantage of our approach is its reliance on maps. While road surfaces are relatively constant over time, they still may change. In extreme cases, our localization technique may fail. While we acknowledge this limitation, the present work does not address it directly. Possible extensions would involve filters that monitor the sensor likelihood, to detect sequences of "surprising" measurements that may be indicative of a changed road surface. Alternatively, it may be possible to compare GPS and map-based localization estimates to spot positioning errors. Finally, it may be desirable to extend our approach to incorporate 3-D environment models beyond the road surface for improved reliability and accuracy, especially on unusually featureless roads.

Chapter 3

Extension to Probabilistic Maps

3.1 Introduction

The mapping and localization algorithms presented in the previous chapter enable a significant increase in localization precision over traditional GPS/IMU systems. However, as they rely on a small number of single-beam laser rangefinders, and thus perform explicit removal of both static and dynamic obstacles, they are not sufficiently robust in the face of heavy occlusions, partially changing environments, or areas with limited features. In order for a vehicle to handle a variety of environments, including ones with dense traffic, it should be able to localize itself in such situations without relying entirely on particular patterns or features of the road surface itself. In fact, in order to enable autonomous driving, a localization system should be able to handle situations where the environment has changed since the map was created.

In this chapter we present a new method of map-based driving that extends previous work by considering maps as probability distributions over environment properties rather than as fixed representations of the environment at a snapshot in time. As in the previous chapter, we build infrared reflectivity maps of the environment and align overlapping portions of the same or disparate trajectories with GraphSLAM, using similar offline relaxation techniques to recent SLAM methods [4, 21, 25, 27, 81, 44, 46]. However, by extending the format of the map to encapsulate the probabilistic nature of the environment, and by using a multi-beam laser rangefinder that observes most surfaces more than once, we are able to

represent the world more accurately and localize with fewer errors. In particular, previous methods often use a binary classification for deciding whether or not to incorporate a piece of evidence into a map. That is, a sensor reading such as a laser scan is either assumed to be a legitimate part of the map, or it is assumed to be spurious and therefore ignored. In [52], exactly this approach was used; any scans that were thought to be vertical according to a binary classification were thrown out.

However, instead of having to explicitly decide whether each measurement either is or is not part of the static environment, we propose as an alternative considering the sum of all observed data and modeling the variances observed in each part of the map. This new approach has several advantages compared to the non-probabilistic alternative. First, while much research incorporating laser remission data has assumed surfaces to be equally reflective at any angle of incidence, this approximation is often quite poor [6, 48, 36]. Whereas Lambertian and retroreflective surfaces have the fortuitous property that remissions are relatively invariant to angle of incidence, angular-reflective surfaces such as shiny objects yield vastly different returns from different positions. Instead of ignoring these differences, which can lead to localization errors, we now implicitly account for them.

A further advantage of our proposed method is an increased robustness to dynamic obstacles; by modeling distributions of reflectivity observations in the map, dynamic obstacles are automatically discounted in localization via their trails in the map. Finally, in addition to capturing more information about the environment, our approach enables a remarkably straightforward probabilistic interpretation of the measurement model used in localization.

Traditionally, autonomous vehicles which require precise localization rely on GPS with Real Time Kinematic (RTK) corrections; this technique requires multiple fixed ground-based receivers, and is thus not applicable to arbitrary locations. We believe the methods described here are the first to allow a vehicle to localize itself precisely enough for autonomous urban navigation without requiring additional ground-based infrastructure.

Our vehicle is equipped with an Applanix LV-420 tightly coupled GPS/IMU system that provides both intertial updates and global position estimates at 200 Hz. The environment is sensed by a Velodyne HD-LIDAR laser rangefinder with 64 separate beams; the entire unit spins at 10 Hz and provides approximately one million 3-D points and associated infrared intensity values per second. This vast quantity of data ensures that most map cells are hit

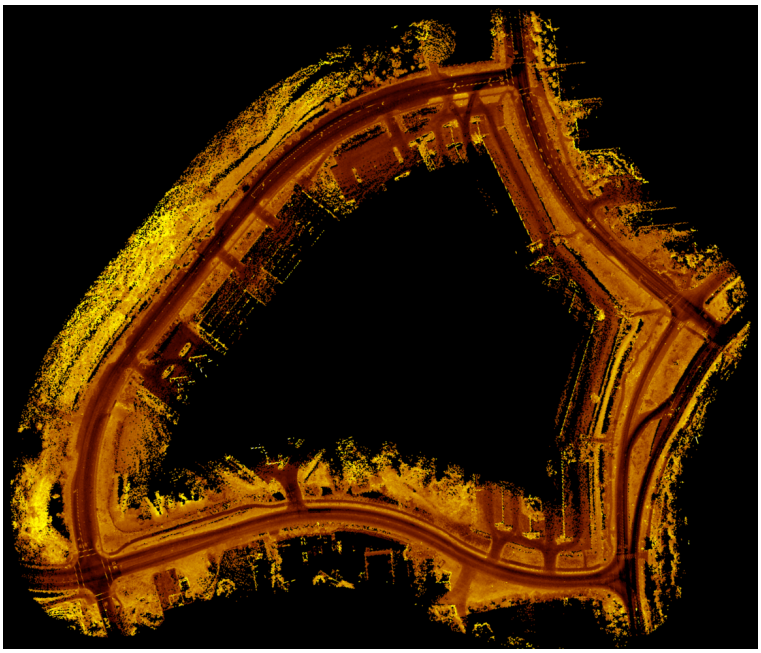


Figure 3.1: An infrared refectivity map of a large urban block showing the average reflectivity of each 15x15cm cell. Due to the incorporation of all laser returns, as well as the multiple beams at different angles, this map is noticeably denser, and has fewer holes and occlusions, than those from the preceding chapter.

multiple times, thereby enabling the computation of intensity variances on a per-cell basis.

We first present the details of our mapping algorithm, including a novel unsupervised laser calibration routine, and then explain how the localizer uses incoming data and a probabilistic map to localize the vehicle. Finally, we show results from several experiments that demonstrate the accuracy and robustness of our approach.

3.2 Probabilistic Maps

Our ultimate goal in building a map is to obtain a grid-cell representation of the observed environment in which each cell stores both the average infrared reflectivity observed at that location as well as the variance of those values. We generate such a map in three steps: first, we post-process all trajectories so that areas of overlap are brought into alignment; second, we calibrate the intensity returns of each laser beam so that the beams have similar response curves; and finally, we project the calibrated laser returns from the aligned trajectories into

a high-resolution probabilistic map. Each of these steps is described in detail below.

3.2.1 Map alignment using GraphSLAM

As in the previous chapter, given one or more logfiles containing GPS, inertial, and laser data, we wish to refine the trajectories in order to bring areas of overlap into alignment. Specifically, when there exist sections of the logfiles that are spatially near but temporally separated, the vehicle's pose in these sections must be corrected so that the two sections align properly. As before, GraphSLAM is used to optimize an objective function in which adjacent vehicle poses are linked by inertial and odometry data, vehicle poses are linked to their estimated global position, and matched sections from the logfile (e.g. for loop closure) are linked by their computed alignment offsets.

Unlike the image convolution approached used in previous chapter, in this implementation, due to the abudance of laser data available, alignment offsets between matched poses are computed from full 3D point clouds. Using iterative closest point on clusters of 5 adjacent 360-degree laser scans from each of the two sections, the x, y, z, yaw, pitch, and roll are all jointly optimized. As in [44] we only consider matches between poses that exceed a particular spatial distance from existing nodes. Once a list of matches has been computed, the GraphSLAM objective function is minimized and the vehicle trajectories are updated accordingly, using the same optimization method as described in the previous chapter.

3.2.2 Laser calibration

Before we generate our map, it is important to calibrate the separate laser beams so that they respond similarly to the objects with the same brightness. With a well calibrated laser, this step can be skipped without effect, but creating a probabilistic map from a poorly calibrated laser suffers from two disadvantages: first, the intensity averages for each cell will depend heavily on which beams happened to hit it; and second, the computed intensity variances will significantly overstate the reality. In practice, we find that it is not necessary to recalibrate our laser every time we use it, but using the factory calibration is unquestionably detrimental.

A single 360-degree scan from the uncalibrated laser can be seen in Fig. 3.2(a). As is

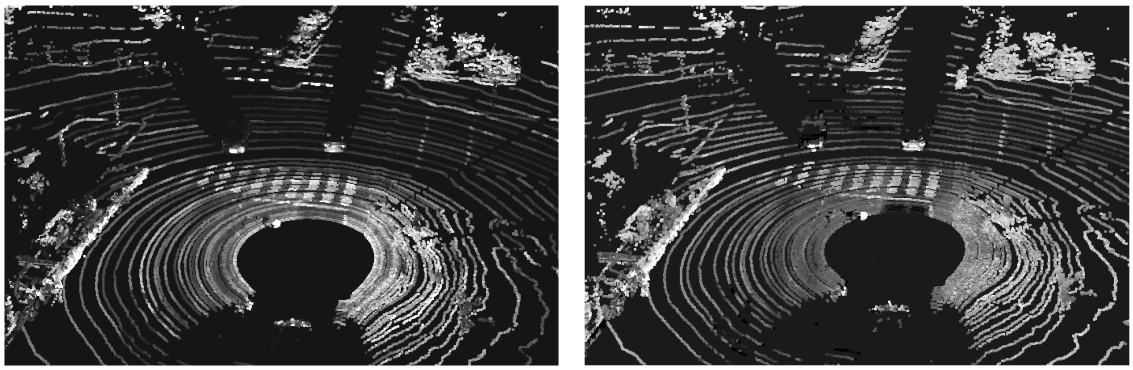


Figure 3.2: Without calibration (left), each of the 64 beams has a significantly different response to reflectivity values. After calibration (right), the beams are much better matched.

readily apparent, some beams are generally too bright and others are generally too dark. Rather than compute a single parameter for each beam (which would still an improvement over the uncalibrated data), we instead compute an entire response curve for every beam, so that we have a complete mapping function for each beam from observed intensity to output intensity. This more sophisticated approach is superior because, due to the particularities associated with the hardware of the laser, each beam has its own unique nonlinear response function.

The details of our calibration approach are presented in Chapter 5; the result of the calibration can be seen in Fig. 3.2(b), which shows significantly improved agreement between beams.

3.2.3 Map creation

Given a calibrated laser and one or more logfiles with properly aligned trajectories, it is now possible to generate a high resolution map.

In order to create a probabilistic map, we store not only the average laser intensity for each map cell, but also the variance of those values. Thus, the map has two channels of data: an intensity, and a variance. Explicitly estimating the actual reflectance properties as [14] do with multiple photographs is not necessary for our approach; simply accounting for the observed variance of infrared remittance values at each map cell is enough for our mapping and localization goals.

(a) Average infrared reflectivity. **(b) Standard deviation of infrared reflectivity.**

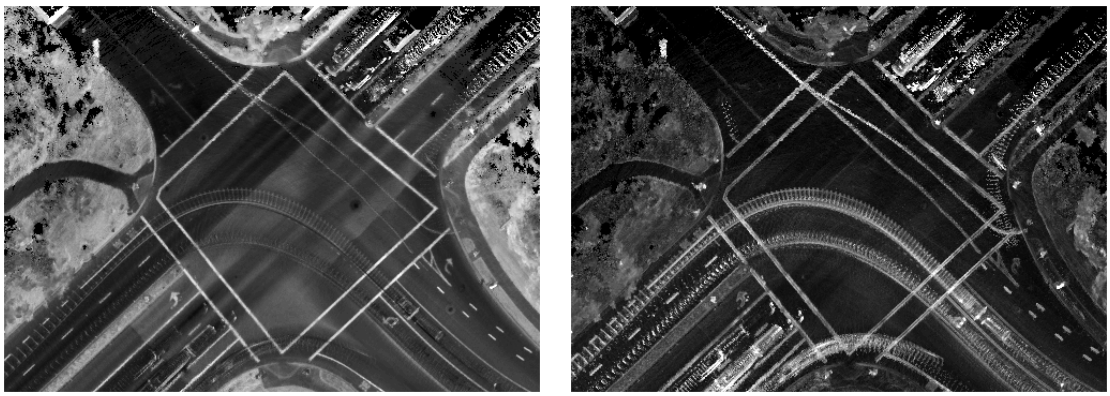


Figure 3.3: The two channels of our probabilistic maps. In (a) we see the average infrared reflectivity, of brightness, of each cell. The innovation of this chapter is to also consider (b), the extent to which the brightness of each cell varies. Note that the trails of the passing vehicles are much more prominent in (b).

The algorithm for generating a probabilistic map is straightforward. As the vehicle transitions through its series of poses, the laser points are projected into an orthographic x, y representation in which each map cell represents a 15x15cm patch of ground. Every cell maintains the necessary intermediate values to update its intensity average and variance with each new measurement. Infrared reflectivity is a rich source of environmental data, as can be seen in Fig. 3.3(a). Unlike camera-based data, this data is immune from shadows and other artifacts caused by passive lighting. Upon close inspection, trails from passing cars can be seen; because the maps are simply averages over time, observations of dynamic obstacles will taint the map. Rather than attempt to delete these, which is impossible to do perfectly, we instead take advantage of the fact that dynamic obstacles tend to leave a signature by causing large intensity variances for the cells in which they pass.

Indeed, Fig. 3.3(b) shows the standard deviations of each cell, in which the dynamic trails stand out very visibly. Here, the probabilistic map encodes the fact that its intensity estimation for those cells is uncertain, so that when the map is later used for localization, intensity values from incoming sensor data that do not match those in the map will not be overly punished. It is also interesting to note that, while it appears at first glance that the lane markings also have high variances, upon closer inspection it is clear that it is actually the edges of the markings which have high variance. This observation is easily explained

by the fact that slight sensor miscalibrations and pose errors will cause the cells near a large gradient to have larger range of intensity returns.

3.3 Online Localization

Once we have built a map of the environment, we can use it to localize the vehicle in real time. We represent the likelihood distribution of possible x and y offsets with a 2-dimensional histogram filter.[1] As usual, the filter is comprised of two parts: the motion update, to reduce confidence in our estimate based on motion, and the measurement update, to increase confidence in our estimate based on sensor data.

3.3.1 Motion update

Our GPS/IMU system reports both inertial updates and a global position estimate at 200Hz. By integrating the inertial updates we maintain a "smooth coordinate" system which is invariant to jumps in GPS pose but which, necessarily, diverges arbitrarily over time. Fortunately, because the smooth coordinate system is updated by integrating velocities, its offset from the true global coordinate system can be modeled very accurately by a random walk with Gaussian noise. Of course, recovering the offset between the two coordinate frames is equivalent to knowing our true global position, so it is this offset that we strive to estimate.

As a result, the motion model for our filter is surprisingly simple; rather than needing to model the uncertainty of the motion of the vehicle itself as is typically done, we need only model the drift between the smooth and global coordinate systems. We note that the car's motion model is not actually being ignored; rather, it is used internally in the tightly-coupled GPS/IMU system precisely to minimize the rate at which the smooth and global coordinate systems drift apart. Vehicle dynamics are discussed in [30].

[1] Although particle filters are a popular alternative method, and were used in the previous chapter, having GPS available allows us to constrain our search space to within several meters of the GPS estimate and thus to calculate directly the probability of all possible offsets at a 15-cm cell size. This method confers the significant advantage of not having to worry that a particle is missing near the correct location. Further, the possibility of achieving accuracy much better than the size of one grid cell would afford us no additional advantage given the other sources of error in our system (e.g. sensor miscalibration, controller error, etc.)

Because the smooth coordinate system's drift is modeled as a Gaussian noise variable with zero-mean, the motion model updates the probability of each cell as follows:

$$\overline{P}(x,y) = \eta \cdot \sum_{i,j} P(i,j) \cdot \exp\left(-\frac{1}{2}(i-x)^2(j-y)^2/\sigma^2\right) \tag{3.1}$$

where $\overline{P}(x,y)$ is the posterior probability, after the motion update, that the vehicle is in cell (x,y), η is the normalizing constant, and σ is the parameter describing the rate of drift of the smooth coordinate system.

Although this update is theoretically quadratic in the number of cells, and thus quartic in the search radius, because the drift rate is relatively low and the update frequency can be arbitrarily high, it is in practice necessary to only consider consider neighboring cells with a small distance of the cell to be updated. For instance, the probability that the smooth coordinate frame drifts more than 1m in .1 seconds is vanishingly small, even though such jumps are relatively common for the global GPS estimate. We process the motion update at a rate proportional to the speed of the vehicle, as the expected drift in the smooth coordinate system is roughly proportional to the magnitude of the vehicle's velocity.

3.3.2 Measurement update

The second component of the histogram filter is the measurement update, in which incoming laser scans are used to refine the vehicle's position estimate.

The way in which we process incoming laser scans is identical to the mapping process described in the previous section. That is, rather than treating every laser return as its own observation, we instead build a rolling grid from accumulated sensor data in the exact same form as our map. This method enables us to directly compare cells from our sensor data to cells from the map, and avoids overweighting cells which have a high number of returns (e.g. trees and large dynamic obstacles).

If z is our sensor data and m is our map, and x and y are possible offsets from the GPS pose, then Bayes' Rule gives:

$$P(x,y|z,m) = \eta \cdot P(z|x,y,m) \cdot P(x,y) \tag{3.2}$$

We approximate the uncertainty of the GPS/IMU pose estimate, which itself is the result of proprietary processing of raw GPS and IMU sensor measurements, by a Gaussian with variance σ^2_{GPS}. Therefore, we can estimate $P(x,y)$ simply as a product of the GPS Gaussian and the posterior belief after the motion update:

$$P(x,y) = \eta \cdot \exp\left(\frac{x^2+y^2}{-2\sigma^2_{GPS}}\right) \cdot \overline{P}(x,y) \tag{3.3}$$

To calculate the probability of sensor data z given an offset (x,y) and map m, we take the product over all cells of the probability of observing the sensor data cell's average intensity given the map cell's average intensity and both of their variances. This value is then raised to an exponent $\alpha < 1$ to account for the fact that the data are likely not entirely independent, for example due to systemic calibration errors or minor structural changes in the environment which are measured in multiple frames [79].

Let us call the two-dimensional arrays of the standard deviations of the intensity values in the map and sensor data m_σ and z_σ, respectively. Then, for example, the standard deviation of the intensity values in the map cell .45m east and 1.2m north of the GPS estimate would be expressed as $m_{\sigma_{(.45,1.2)}}$.

We will use the same notation for the average intensity value, with r denoting the average intensity (reflectivity) of the cell. Again, to use the same example, the average intensity seen in the map at the cell .45m east and 1.2 north of the GPS estimate would be expressed as $m_{r_{(.45,1.2)}}$.

Thus, we have:

$$P(z|x,y,m) = \prod_{i,j} \exp\left(\frac{-(m_{r_{(i-x,j-y)}} - z_{r_{(i,j)}})^2}{2(m_{\sigma_{(i-x,j-y)}} + z_{\sigma_{(i,j)}})^2}\right)^\alpha \tag{3.4}$$

Putting it all together, we obtain:

(a) GPS localization induces ≥1 meter of error. **(b) No noticeable error after localization.**

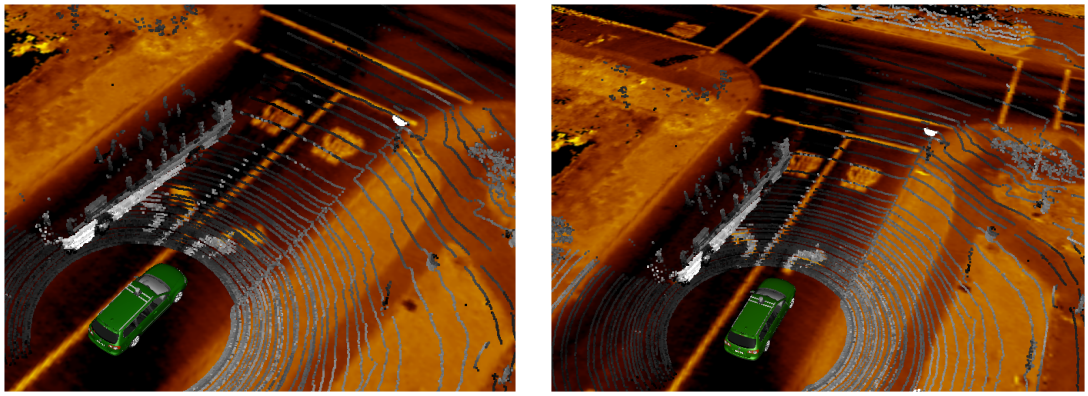


Figure 3.4: Incoming laser scans (grayscale) superimposed on map (gold). (a) GPS localization is prone to error, even (as shown here) with a high-end integrated inertial system and differential GPS using a nearby stationary antenna. (b) With localization there is no noticeable error, even in the presence of large dynamic obstacles such as this passing bus.

$$P(x,y|z,m) = \eta \cdot \prod_{i,j} \exp\left(\frac{-(m_{r_{(i-x,j-y)}} - z_{r_{(i,j)}})^2}{2(m_{\sigma_{(i-x,j-y)}} + z_{\sigma_{(i,j)}})^2}\right)^{\alpha} \cdot \exp\left(\frac{x^2+y^2}{-2\sigma_{GPS}^2}\right) \cdot \overline{P}(x,y) \tag{3.5}$$

Towards the end of achieving robustness to partially outdated maps, we further impose a minimum on the combined standard deviation for the intensity values of the map and sensor data. This implicitly accounts for the not-unlikely phenomenon that an environment change since the map acquisition simultaneously enabled a low variance in both the map and the sensor data, yet with the two showing significantly different intensity values.

For computational reasons we restrict the computation of $P(x,y|z,m)$ to cells within several meters of the GPS estimate; however, this search radius could easily be increased if a less accurate GPS system were used.

3.3.3 Most likely estimate

Given the final posterior distribution, the last step is to select a single x and y offset that best represents our estimation. Taking the offset to be $\max_{x,y} P(x,y)$ is, by definition, probabilistically optimal at any given instant, but such an approach could add unnecessary danger as part of the pipeline in an autonomous vehicle. Because the maximum of a multimodal distribution can easily jump around discontinuously, using that approach may cause the vehicle to oscillate under unfortuitous circumstances, even if the vehicle's navigation planner performed some variety of smoothing. An alternative approach would be to choose the center of mass of the posterior distribution; this would improve consistency, but would tend to cause the chosen offset to be biased too much towards the center. A compromise is to use the center of mass with the variation that $P(x,y)$ is raised to an exponent $\alpha > 1$; in our implementation, we empirically chose $\alpha = 2$. Thus, the (x,y) offset is computed as:

$$x = \frac{\sum_{x,y} P(x,y)^{\alpha} \cdot x}{\sum_{x,y} P(x,y)^{\alpha}} \qquad y = \frac{\sum_{x,y} P(x,y)^{\alpha} \cdot y}{\sum_{x,y} P(x,y)^{\alpha}} \tag{3.6}$$

This offset is the final value that is sent to the vehicle's navigation planner. While an advanced planning and decision-making algorithm could take advantage of the entire posterior distribution over poses rather than requiring a single, unimodal pose estimate, our vehicle's planner expects a single pose estimate and thus this equation has proven useful as it constitutes a practical compromize between the high bias of mean-filtering and the high variance of the mode.

In our vehicle this offset is computed and sent at 10 Hz. Subsequently, the vehicle is able to plan paths in global coordinates using the best possible estimate of its global position. An example of the localizer's effect is shown in Fig. 3.4.

3.4 Experimental Results

The above algorithms were implemented in C such that they are capable of running in real time in a single core of a modern laptop CPU. Map data requires roughly 10 megabytes of data per mile of road, which enables extremely large maps to be stored on disk. Maps are

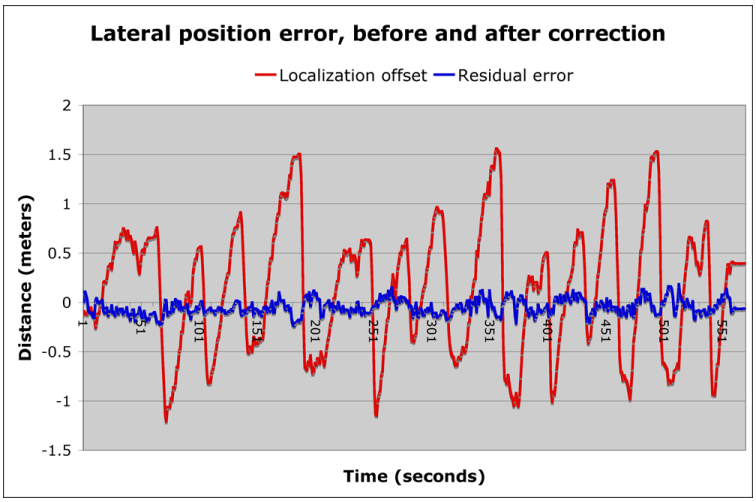


Figure 3.5: Comparing the lateral offset applied by our algorithm (red) to the residual error after localization (blue) as measured by offline SLAM alignment. During these ten minutes of driving, RMS lateral error has been reduced from 66cm to 9cm.

stored in a tiled format so that data grows linearly with terrain covered, and RAM usage is constant regardless of map size.

We conducted extensive experiments of the localization system, both manually and autonomously. We now present both quantitative results demonstrating the high accuracy of our localizer and discuss autonomous results we were unable to achieve without the present techniques.

3.4.1 Quantitative Results

In order to quantitatively evaluate the performance of our methods in the absence of known "ground truth," we employ the same offline GraphSLAM alignment described in the mapping section to align a recorded logfile against an existing map; we then compare the offsets generated in this alignment to the offsets reported by the localizer. Ideally, the offline and online methods should yield similar results, though the offline SLAM approach would likely have higher accuracy as it uses more than 100 times the amount of data. In fact, our probabilistic maps can be thought of as an efficient reduction of the entire set of laser data that, by virtue of storing the intensity variances, loses much less information than does a typical map.

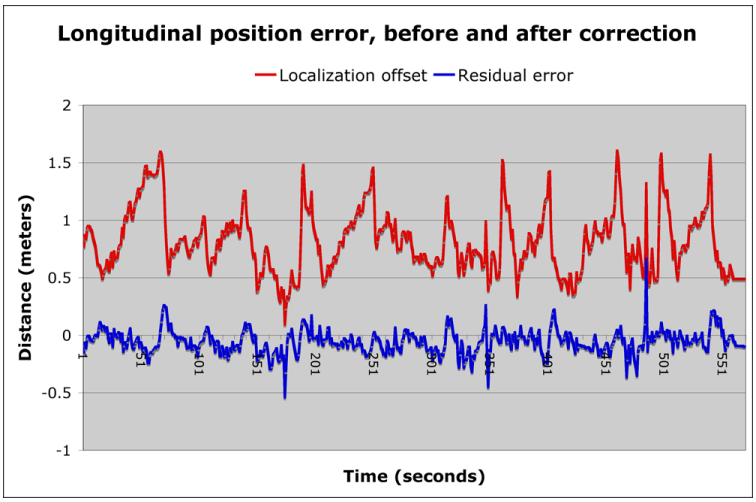


Figure 3.6: Comparing the longitudinal offset applied by our algorithm (red) to the residual error after localization (blue) as measured by offline SLAM alignment. During these ten minutes of driving, RMS longitudinal error has been reduced from 87cm to 12cm. Note also the systematic bias in longitudinal error which is removed after localization.

For our first test, we drove around a very large urban block several times in July and used our mapping method to create a probabilistic map. We then collected a separate logfile in September of the vehicle driving the same block three times; this was approximately ten minutes of driving in dense traffic. At this point, the online localizer was used to align the September route against the July map. Separately, offline GraphSLAM was also used to align the two trajectories using all available data.

The map is shown in Fig. 3.1. Fig. 3.5 shows the lateral offsets applied by the localizer during the ten-minute September drive compared against the difference in the localizer alignment and the offline SLAM alignment. During this drive, an RMS lateral correction of 66cm was necessary, and the localizer corrected large errors of up to 1.5 meters. As the graph illustrates, the resulting error after localization was extremely low, with an RMS value of 9cm. It should be noted that this value is quite a bit less than the grid cell size of 15cm, and also that the GraphSLAM alignment itself is likely to have minor errors; thus, this result is about as good as we could hope to achieve with this method of evaluation.

A companion graph for longitudinal corrections of the same drive is shown in Fig. 3.6; here, the localizer corrected an RMS longitudinal error of 87cm and agreed with the GraphSLAM alignment to within 12cm RMS. Interestingly, whereas the lateral corrections had

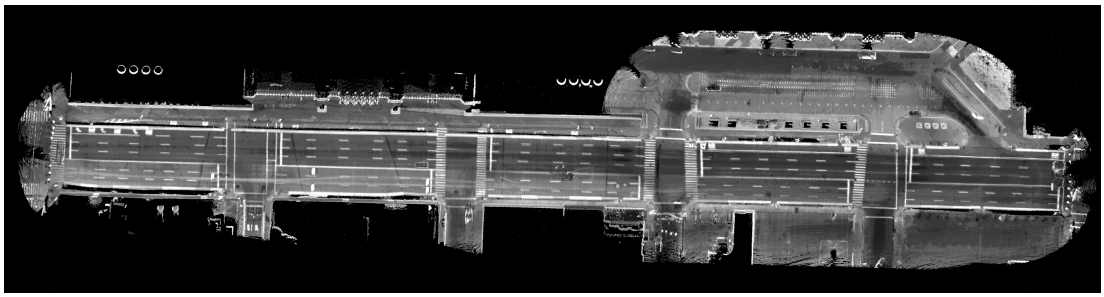


Figure 3.7: An infrared reflectivity map of 11th Avenue in New York City, acquired during our autonomous demonstration at the 2008 ITS World Congress.

roughly a zero mean, that is not the case for the longitudinal corrects. The fact that the average longitudinal correction was about 75cm forward suggests that there was a systematic bias somewhere; perhaps the wheel encoder or GPS signal was systematically off during the mapping or localization run. In any case, the localizer was clearly able to correct for both the systematic and non-systematic effects with great success.

3.4.2 Autonomous Success

In addition to evaluating our performance quantitatively, we also ran several autonomous experiments in which the vehicle navigated autonomously in real urban environments. Using previously published localization methods, we were able to drive autonomously on moderately wide roads and only in low traffic, because turns could not be made with sufficient accuracy and narrow roads posed too great a risk.

However, using the methods presented here, we are now able to drive autonomously in several urban environments that were previously too challenging. In one example, at an invitation to the 2008 Intelligent Transport Systems World Congress, our vehicle participated in an autonomous vehicle demonstration in downtown Manhattan in which several blocks of 11th Avenue were closed to regular traffic. Our vehicle operated fully autonomously with other autonomous and human-driven vehicles and succesfully stayed in the center of its lane, never hitting a curb or other obstacle, despite that the environment configuration had changed considerably since the map had been acquired. Our localization system performed successfully at this event during both day and night and in dry weather and heavy rain. The laser map we acquired is shown in Fig. 3.7, and our vehicle participating in the

Figure 3.8: Junior preparing for the the 2008 ITS World Congress in Manhattan (left) and giving an autonomous demo, running the localization algorithms described in this chapter (right).

event is shown in Fig. 3.8.

In another example, we mapped a four-mile loop around a local campus that includes roads as narrow as 10 feet, tight intersections, and speed limits up to 40 MPH. We were able on the first attempt to drive the entire loop with the vehicle completely controlling its own steering; an intervention was never necessary even amidst heavy rush-hour traffic. This route is depicted in Fig. 3.9.

Since performing these localization-specific experiments, the algorithms presented in this chapter have already enabled our vehicle to drive several hundred miles autonomously in traffic on urban roads without a single localization-related failure.

3.5 Conclusion

Localization is a critical enabling component of autonomous vehicle navigation. Although vehicle localization has been researched extensively, no system to our knowledge has yet proven itself able to reliably localize a vehicle in dense and dynamic urban environments with sufficient accuracy to enable truly autonomous operation. Previous attempts have successfully improved upon GPS/IMU systems by taking environment into account, but suffered from an inability to handle changing environments. While our approach is not infinitely adaptable and can be hindered by sufficiently severe changes in weather or environmental configuration, we believe it is a significant step forward towards allowing vehicles

Figure 3.9: A four-mile loop around which our vehicle nagivated autonomously in traffic with no localization failures.

to navigate themselves in even the trickiest of real life situations.

By storing maps as probability models and not just expected values, we are able to much better describe any environment. Consequently, localizing becomes more robust and more accurate; compared with previous work, we suffer far fewer complete localization failures and our typical localization error is significantly reduced, especially in the more challenging longitudinal direction. The fact that we were able for the first time to autonomously nagivate our vehicle in our most difficult local streets (with lanes as narrow as 10 feet), amidst rush-hour traffic, is a testament to the precision and robustness of our approach. Indeed, extensive experiments suggest that we are able to reduce the error of the best GPS/IMU systems available by an order of magnitude, both laterally and longitudinally, thus enabling decimeter-level accuracy; as we have shown, this performance is more than sufficient for autonomous driving in real urban settings.

There remain promising areas for further research on this topic. In the interest of efficiency and robustness we project points to the xy plane in our maps, but height information is surely useful; while infrared reflectivity is abundantly rich in information, a more complex approach could also reason in the space of map elevation. For example, ray tracing could be used in both the map-making and localization phases to explicitly process and remove dynamic obstacles, and vertical static obstacles could be incorporated into the

measurement model.

If maps are to be learned over extremely long periods of time, with large numbers of passes over the same streets, techniques such as [45, 46] could be used to prune the ever-growing graph structure. In addition, future work could attempt to explicitly account for correlations between the errors at adjacent grid positions.

Chapter 4

Localization Using a Vector Road Map

4.1 Introduction

A crucial component of any intelligent vehicle is the ability to determine its location relative to a road or map represented in global coordinates. In the previous two chapters, we discussed methods of localizing a vehicle in dense urban environments that require the creation and availability of high-density environment maps. A localization approach offering sufficient precision and reliability in urban environments without requiring such a dense map would afford several advantages over these methods, and in some situations may represent an appropriate tradeoff even if some degree of absolute accuracy were sacriced.

First, dense environment maps are expensive and time-consuming to acquire. Second, sufficiently major changes in the environment (e.g. construction or lane repainting) may require the acquisition of new map imagery. A sparser GPS waypoint representation containing vector descriptions of lanes would sacrifice information but significantly simplify map creation and enable maps encompassing the entire area of the United States to fit on an inexpensive hard drive. Furthermore, changes to road configurations could be easily applied via simple measurements or aerial photographs.

Localization of a vehicle relative to such a sparse map is a more difficult task since sensor data cannot be directly compared to a map consisting of a similar data type. Instead, environmental features such as lane markings and curbs must be dynamically extracted and fused with GPS into an appropriate location estimate. In this chapter, we describe such

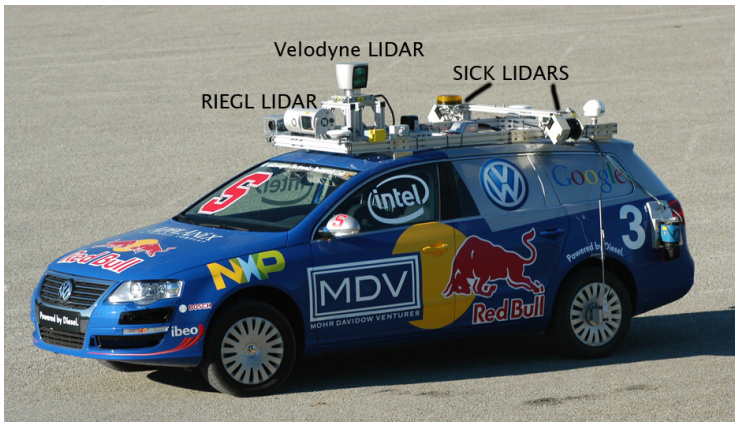


Figure 4.1: Our vehicle is equipped with an integrated GPS-IMU system and several LIDARs. The LIDARs used for localization are labeled above: the SICK and RIEGL scanners are used to find lane markings and the Velodyne is used to detect curbs.

an approach that enables highly accurate and reliable localization of a vehicle in dynamic urban environments based solely on a vector description of a road network.

The primary motivator of this localization approach was the 2007 Defense Advanced Research Projects Agency (DARPA) Urban Challenge event. Following the highly successful 2005 DARPA Grand Challenge, in which robotic vehicles drove a desert course with no dynamic obstacles and no traffic rules, the 2007 Urban Challenge offered an opportunity to demonstrate urban driving in the presence of other human-driven and autonomous vehicles, while obeying a large number of California driving laws. Multiple robotic vehicles carried out missions in the same environment at the same time, while needing to perform maneuvers including passing parked or slow-moving vehicles, left turns across oncoming traffic, parking in a parking lot, and execution of U-turns in the presence of road blocks. For almost all aspects of the event, localization of the vehicle relative to the lanes of the road was critical. Our entry in the competition, "Junior," is shown in Fig. 4.1.

The method presented in this chapter localizes a vehicle relative to a vector map format specified by DARPA, called a Road Network Distribution File (RNDF) [13]. An RNDF contains a list of road segments, each of which is composed of a list of lanes, each of which is composed of a list of GPS waypoints. Some additional information, such as lane width and lane marker type (if known), is also specified. A graphical depiction of a simple RNDF is shown in Fig. 4.2, while a large and complex RNDF is shown in Fig. 4.3.

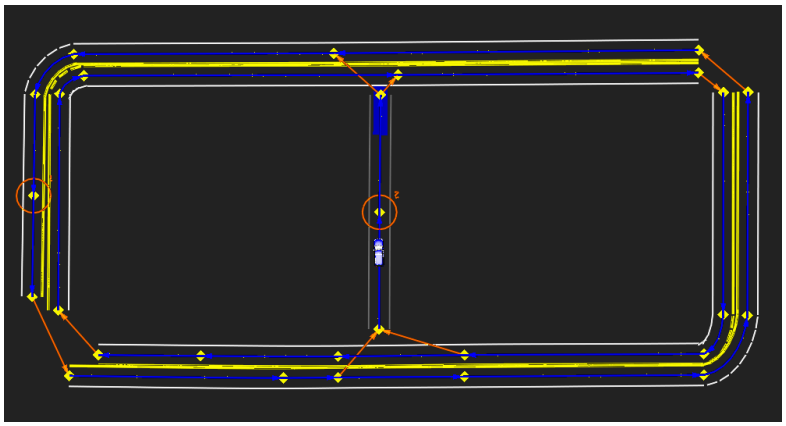


Figure 4.2: A small RNDF: the 2007 Urban Challenge qualifying event, course "A"

As DARPA's provided RNDFs in some places had such sparse waypoints that linear interpolation would result in a path deviating quite far from the actual road, we manually adjusted and augmented the DARPA-provided RNDFs prior to the events. Using DARPA-provided aerial imagery, we made an attempt to adjust an upsampled RNDF such that it more closely aligned with the roads. Since the aerial imagery was neither current nor especially accurate, such adjustments were necessarily coarse and did not obviate the need for precision localization. Nevertheless, the adjustment was helpful because our localization method was not designed to deal with immense RNDF errors.

Because an RNDF, and indeed any road, is highly variable in the direction perpendicular to the vehicle, but slowly changing in the direction of travel, we simplify the problem to localizing the vehicle laterally relative to the direction of the RNDF and relying on GPS and previous calculations for the longitudinal location.

The most advanced integrated GPS-IMU systems available offer global position estimates accurate to within 30 to 60 cm in the ideal case, but often worse in urban environments. Uncorrected, this amount of precision is generally insufficient for reliable driving [85]. In addition, any global inaccuracies in the RNDF only compound the GPS error, and to the extent that an online localization system can accomodate moderate RNDF errors, it will be that much more robust in real-world scenarios (e.g. the 2007 DARPA Urban Challenge). Furthermore, in the absence of a detailed environment map, a vehicle will need to deal with uncertainty in many forms, including incomplete or missing lane markers, barely

Figure 4.3: A large RNDF: the entire map for the 2007 Urban Challenge race.

visible curbs, sensor noise, and other moving traffic.

Thus our goal is a hardware and software system comprised of algorithms capable of dealing with uncertainty at all levels to solve the problem of lateral localization relative to an RNDF. Towards this end, we equip our vehicle (see Fig. 4.1) with an integrated GPS-IMU system for global position estimates and inertial updates, and several LIDAR sensors which measure not only distance but also infrared remittance; the latter is used to detect lane markings in liu of the traditional camera-based approach. Fig. 4.4 shows distance and intensity LIDAR data.

As the vehicle drives, the localization algorithm processes laser scans to detect lane markings and curbs. The filter probabilistically rewards lateral offsets for which lane-marker-like reflectivity patterns align with the lane markers implicit in the RNDF. In a similar but opposite fashion, the filter penalizes lateral offsets for which observed curbs would intrude into the lane corridors specified in the RNDF. Finally, the filter prefers lateral offsets closer to the current best GPS estimate. Data is integrated over time and all three of these constraints are combined into a single objective function which is computed directly. The objective function yields a posterior probability distribution over possible lateral offsets, from which the best offset can be readily computed.

A key contribution of our approach is the notion that features are never classified in a binary or discrete fashion; instead, the degree to which sensor data suggests a particular

Figure 4.4: Visualization of the scanning process: the LIDAR scanner acquires range data and infrared ground reflectivity. Notice that lane markings have much higher reflectivity than pavement.

feature (such as a lane marker or a curb) probabilistically informs the final result. In this manner, the typical necessity of selecting arbitrary thresholds is largely eliminated, and subtle features and their interactions are still considered. Consequently, difficult-to-detect environment features such as small curbs and faint or broken lane markers are not ignored, and particularly in the absence of more obvious features, can and do affect the result of the localization filter.

The specific mathematics and implementation of the probabilistic constraints and objective function follow.

4.2 Lane Marker Matching Constraint

The first of the three localization constraints is the lane-marker matching constraint. Intuitively, the locations of detected lane markers in the road ought to match their expected locations based on the RNDF map. If for a given lateral offset, the detections agree with the map, that is evidence in favor of that offset. On the other hand, if lane markers are detected in locations where there shouldn't be any according to the map, or if the map indicates a lane marker should be present and none is detected, that is evidence against that offset. Thus, the lane-marker constraint gives preference to lateral offsets for which observed lane markers match those implicit in the RNDF.

Computation of this constraint entails three steps. First, a lane marker response "prior"

is computed from the RNDF. Second, an individual laser scan is filtered for its ”lane marker response”. Finally, these two responses are convolved to yield a response function representing the strength of alignment between the observed data and the RNDF for various possible lateral offsets.

4.2.1 RNDF lane marker response prior

The computation of the lane marker response prior from the RNDF is straightforward:

1. Select all RNDF lane segments whose borders intersect an imaginary line through the vehicle, perpendicular to its yaw.

2. For each intersecting lane border, generate a peak with an exponential decay. Lane borders marked as ”solid” receive twice the intensity of lane borders marked as ”dashed” or ”unknown”.

We call this response prior RNDF_R(s, x), where s is the laser scan corresponding temporally to this prior and x is the lateral offset, such that RNDF_R(4, 1.5) is the response of the prior corresponding to the 4th most recent laser scan 1.5 meters to the right of the vehicle. Thus, according to our convention, an offset of 0 would imply that the RNDF coordinate system were exactly aligned with the GPS output.

An example RNDF lane marker response prior is the blue curve shown in Fig. 4.5.

4.2.2 Laser lane marker response filter

Given a laser scan perpendicular to the vehicle containing of a list of 3D points and infrared remittance intensities, we compute the lateral lane marker response as follows:

1. Resample the intensities from the laser scan points into an intensity response perpendicular to the vehicle. Discard non-ground points, using a simple z-threshold.

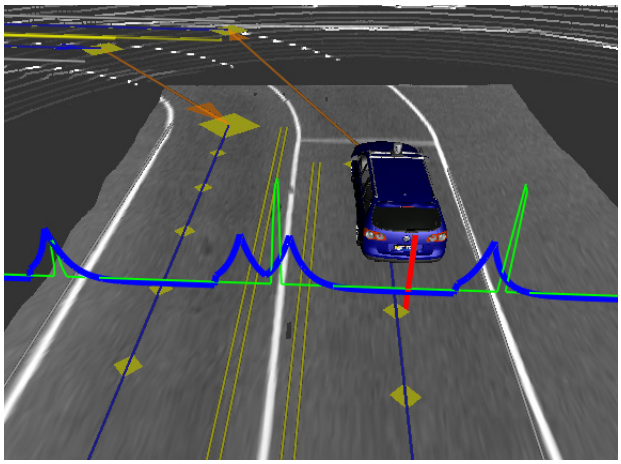


Figure 4.5: Lane-marker matching components: the blue curve is the lane marker prior based on the RNDF, and the green curve is the lateral response of the side lasers to the lane marker filter. Note that the RNDF suggests that the two lanes do not share a common center boundary, when in fact they do. Such inconsistencies between the RNDF and reality were common in the Urban Challenge, but our probabilistic localization algorithm was well equipped to deal with them.

2. At each lateral location, compute the lane marker filter response. The filter response at location x is the intensity at location x minus the brighter of the two intensities 20 cm to the right and left of x. If negative, the response is set to 0.

We call this laser lane marker response function L_R(s, x), where s is the laser scan and x is the lateral offset, such that L_R(4, 1.5) is the response of the 4th most recent laser scan to the lane marker filter 1.5 meters to the right of the vehicle.

An example laser lane marker response is the green curve shown in Fig. 4.5.

4.2.3 Computation of RNDF alignment strength

Given the RNDF prior RNDF_R() and the laser lane marker responses L_R() for all laser scans and lateral offsets, we can compute the alignment strength for various offsets by performing a convolution between the two functions. We give a higher weight to more recent scans, because they are more relevant to our current location. Thus the lane marker alignment response LM_AR(x) to a lateral offset x is defined as:

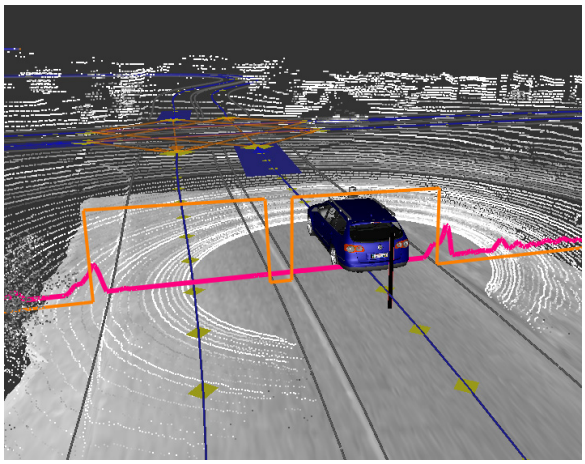


Figure 4.6: Curb avoidance components: the orange curve is the lane corridor prior, and the pink curve is the lateral curb response function. Even though the curbs in this example are subtle, they are still detected.

$$LM\_AR(x) = \eta \cdot \sum_{s} \gamma^{s} \int_{l} L\_R(s,l) \cdot RNDF\_R(s,l-x) \tag{4.1}$$

where η is the normalizer and $\gamma < 1$ is the discount factor for older scans. We use $\gamma = .99$, and compute the value by sampling the lateral response functions in 5-cm intervals and using a discrete summation rather than an integral.

This lane marker alignment response function will be used as one of the three probabilistic constraints in the final objective function to compute the overall probabilities of possible lateral offsets.

4.3 Curb Avoidance Constraint

The second probabilistic constraint is the curb avoidance constraint, which gives preference to lateral offsets for which detected curbs fall outside the lane corridors specified in the RNDF. Intuitively, an RNDF alignment which results in a detected curb being inside a lane is less probable than one in which curbs lie outside of the lanes. Note that the use of curbs in this algorithm is *not* for the explicit purpose of avoiding hitting them; that is the planner's job. Rather, in this section we use curbs as a probabilistic hint for where the lanes

themselves are.

The general approach to computing the curb avoidance constraint is similar to that of the lane marker matching constraint, with two important differences. First, our method of detecting small curbs is substantially more involved than the lane marker response filter; rather than analyzing single 2D laser scans one-at-a-time we integrate 3D laser scans over time into a rolling grid and perform a convolution and edge filter to find curbs parallel to the RNDF. Second, the RNDF prior in this case is simply a binary function denoting whether or not a particular location is inside or outside a lane.

4.3.1 RNDF lane corridor prior

The computation of the binary lane corridor prior from the RNDF is straightforward:

1. Select all RNDF lane segments whose interiors intersect an imaginary line through the vehicle, perpendicular to its yaw.

2. For each intersecting lane segment, mark all lateral offsets within the lane as 1; mark all remaining offsets as 1/10. This ratio implies that lanes are ten times as likely to be free of detected curb-like objects than are non-lanes.

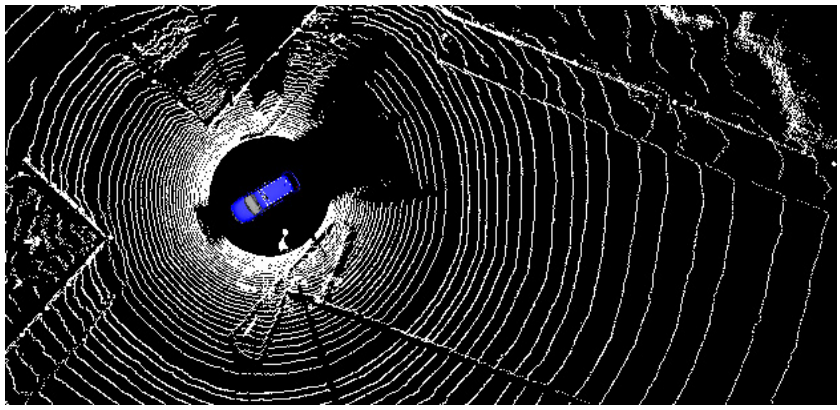
We call this lane corridor prior LC(s, x), where s is the laser scan corresponding temporally to this prior and x is the lateral offset, such that LC(4, 1.5) is 1 if at the time corresponding to the 4th most recent laser scan, the location 1.5 meters to the right of the vehicle was inside an RNDF lane, else it is 0.

An example RNDF corridor prior is the orange curve shown in Fig. 4.6.

4.3.2 Curb response filter

The goal of the curb response filter is to compute the amount of "curbness" present at various lateral offsets from the vehicle; that is, the likelihood that at a particular lateral offset, there exists a curb that should not be in the lane. We use data from the 360° 64-beam Velodyne laser for this task.

(a) A single 360° 64-beam scan.



(b) Several scans integrated over time.

Figure 4.7: The same region captured by a single scan (top) and by integrating several scans over time, with each point scored using the method described (bottom). By integrating scans as the vehicle moves and discounting points that fall within the concentric circles, stationary obstacles such as curbs begin to stand out relative to the ground plane.

Although the simplest approach would be to perform single-scan analysis, we find that better results can be obtained from a more thorough approach involving the integration of data over time, and exploiting the fact that curbs are generally parallel to the lanes specified in the RNDF. The filter response is generated as follows:

1. Maintain a rolling local 2-D grid. We use a 100x100m grid with 10x10cm cells.

2. For each laser point, add to the corresponding 2-D grid element the dot product of the angle of the beam and the angle the point forms with the next point in the scan. This

weighting function causes ground points to receive a low score, because ground returns tend to generate concentric circles. Fig. 4.7 illustrates this step.

3. Simultaneously compute and cache the yaw of the nearest RNDF lane to each laser point.

4. Convolve each element in the scored 2-D grid (of step 2) by a linear point-spread function in the direction of the nearest (cached) RNDF yaw.

5. Perform an edge filter perpendicular to the vehicle yaw and project all filter responses within 10 meters longitudinally of the vehicle down to a single lateral response curve.

We call this curb response function Curb_R(s, x), where s is the last relevant laser scan and x is the lateral offset, such that Curb_R(4, 1.5) is the curb response at the time of the 4th most recent laser scan at the location 1.5 meters to the right of the vehicle.

An example curb response is the pink curve shown in Fig. 4.6.

4.3.3 Computation of RNDF alignment strength

Given the RNDF lane corridor prior LC() and the curb responses Curb_R() for all laser scans and lateral offsets, we can again compute the alignment strength for various offsets by performing a convolution between the two functions. Since a maximal value of the convolution ought to imply that the selected offset results in minimal overlap between a lane corridor and a curb, we use the inverse of Curb_R() in the convolution.

Thus the curb avoidance alignment response CA_AR(x) to a lateral offset x is defined as

$$CA\_AR(x) = \eta \cdot \sum_s \gamma^s \int_l Curb\_R(s,l) \cdot LC(s, l-x)^{-1} \tag{4.2}$$

This curb avoidance alignment response function will be used as the second of three probabilistic constraints in the final objective function to compute the overall probabilities

of possible lateral offsets.

4.4 GPS Constraint

The final probabilistic constraint is the preference to select an offset near the current GPS estimate. Offsets which are increasingly far from the GPS estimate are increasingly unlikely, and the noise model for the error arising from an integrated GPS-IMU system is well approximated by a 0-mean Gaussian.

Importantly, GPS error alone is not the only cause of location uncertainty, as the RNDF itself may not be exactly correct. RNDF waypoints with incorrect positions are the most obvious source of potential error, but error can also arise from a sparsity of waypoints; if the interpolation between waypoints (whether automated or manual) is not perfect, then the RNDF error between waypoints may be significant.

Because the RNDF error should be close to 0 on average, and smaller errors are more common than larger ones, we also approximate RNDF error with a 0-mean Gaussian. Although an arbitrarily bad RNDF could in theory deviate from this assumption, our system is not designed to deal with enormous offsets. As the GPS and RNDF errors are uncorrelated, we consider their combined error to be a single (changing) value and do not distinguish between the two. Decoupling them would add an unnecessary layer of complexity to the estimation and would not yield an obvious benefit; after all, the goal is to know how the map aligns with the vehicle, not whether the alignment is due to GPS or RNDF error. Thus we define the GPS alignment response to a lateral offset x as:

$$GPS\_AR(x) = \eta \cdot \exp\{-\frac{1}{2}x^2/\sigma^2\} \tag{4.3}$$

where σ is the standard deviation of the combined expected GPS and RNDF errors. We note that it would be feasible in principle to make an RNDF map with RTK-processed GPS data and a high enough frequency of waypoints that σ could effectively be reduced to just GPS error. However, in the case of the Urban Challenge, the RNDF was of neither sufficient accuracy nor density to make that assumption. Therefore we used σ = 1.8m for

the event.

4.5 Lateral Localization

With the lane marker matching, curb avoidance, and GPS constraints calculated, we can now derive the objective function for the full posterior distribution over lateral alignment offsets. To do so, we multiply the three constraint functions together and normalize the result so that the integral of the probability density function is 1. Thus the posterior distribution $p(x)$ is defined as:

$$p(x) = \eta \cdot GPS\_AR(x) \cdot LM\_AR(x) \cdot CA\_AR(x) \tag{4.4}$$

where η is the normalization constant.

For the sake of completeness, we expand (4) by substituting in the three constraint functions:

$$\begin{aligned} p(x) = \quad & \eta \cdot \exp\{-\frac{1}{2}x^2/\sigma^2\} \\ & \cdot \sum_s \gamma^s \int_l L\_R(s,l) \cdot RNDF\_R(s,l-x) \\ & \cdot \sum_s \gamma^s \int_l Curb\_R(s,l) \cdot LC(s,l-x)^{-1} \end{aligned} \tag{4.5}$$

Computing this equation requires discretization of the integrals; we sample the lateral constraint functions in 5-cm increments within a range of +/- 4 meters from an offset of 0.

It is readily apparent that much of equation (4.5) can be cached upon computation so that it need not be recomputed every time a posterior distribution is generated. In particular, the convolution integrals need only be computed once per laser scan. We maintain a sufficient cache (1000 scans or 100 seconds, whichever is longer) such that discarded scan convolutions are effectively irrelevant to the outcome.

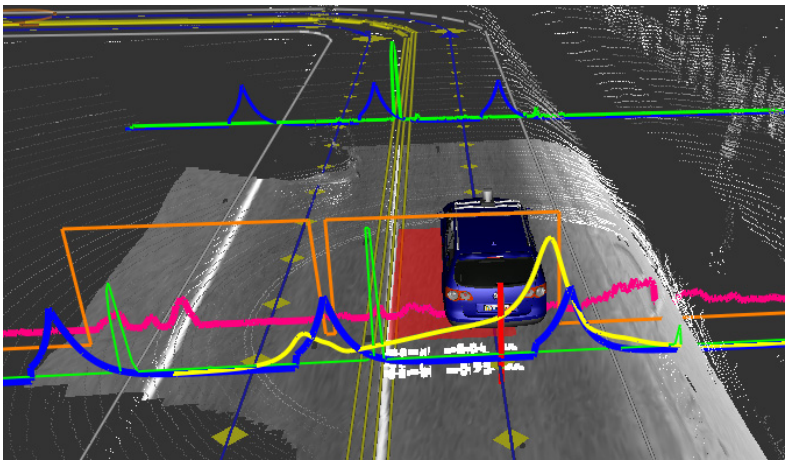


Figure 4.8: Localization showing all components at once. The yellow curve is the final posterior distribution over lateral offsets. Here localization is shifting the vehicle 80 cm to the left.

An example posterior distribution is the yellow curve shown in Fig. 4.8. This posterior distribution simultaneously incorporates the influences of lane marker matching, curb avoidance, and GPS.

Given the final posterior distribution, the last step is to select a single lateral offset with which to align the RNDF. As we noted in the previous chapter, taking the offset to be $max_x p(x)$ is, by definition, probabilistically optimal at any given time, but such an approach could be unreasonably dangerous as part of the pipeline in an autonomous vehicle. As before, we use a compromise in which we take the center of mass with the variation that we raise p(x) to some exponent $\alpha > 1$, as follows:

$$\text{Offset} = \frac{\int_x p(x)^\alpha \cdot x}{\int_x p(x)^\alpha} \tag{4.6}$$

This offset changes more smoothly than the instantaneous maximum of the distibution, and is the final value which is sent to the vehicle's navigation planner. In our vehicle this value is computed and sent at 5 Hz. Subsequently, the vehicle is able to plan paths in global coordinates using the best possible estimate of its global position.

4.6 Results

We conducted extensive tests of our localization system both independently and as an integral component of an end-to-end autonomous vehicle. The 2007 Urban Challenge provided an ideal venue to showcase the localization performance, enabling our vehicle to stay in the center of the lane without ever hitting a curb or mistakenly crossing the center line. In this section we will discuss the hardware requirements of the localizer as implemented, as well as its quantitative and qualitative performance in the Urban Challenge events.

4.6.1 Hardware requirements

Although the main algorithms presented in this chapter are intended to apply to a variety of vehicle and hardware configurations, so long as an ability to detect lane markers and curbs is present, the specifics of the configuration we employed on our vehicle relevant to localization will be discussed briefly.

An Applanix POS-LV420 tightly coupled GPS-IMU system provided interial updates as well as global positioning estimates. A RIEGL LMS-Q120 LIDAR scanning a line in front of the vehicle at 10 Hz and two SICK LMS 291-S14 LIDARs scanning the sides of the vehicle at 75 Hz provided both distance and intensity data for lane marker detection. Finally, a Velodyne HDL-64E LIDAR with 64 beams scanning in all directions at 10 Hz provided 1 million 3D points per second for curb detection.

The localization software was implemented in C to run in real time on a single core of a quad-core Intel Xeon processor, alongside the other modules operating on our autonomous vehicle (including a navigation planner, vehicle controller, and obstacle perception module.) The localization module was tested continuously for up to six hours at a time without failure.

4.6.2 Qualitative performance

Prior to the 2007 Urban Challenge, we performed extensive testing of our vehicle's localization system on our local roads. Fortunately, it is possible to test the output of the localizer

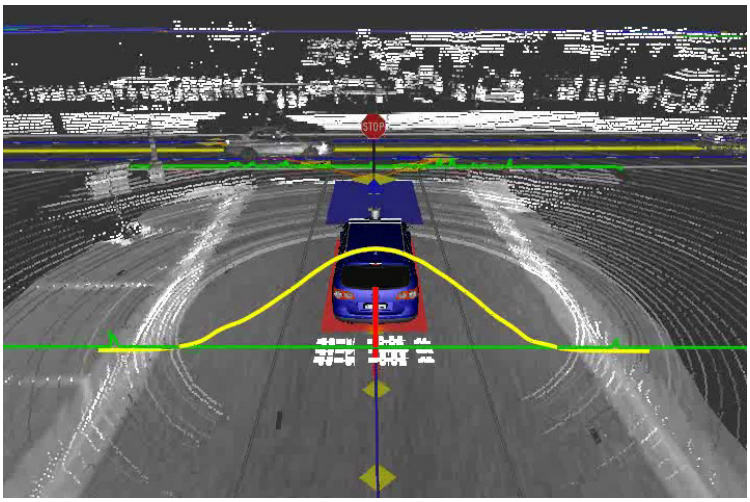


Figure 4.9: Road without lane markers or nearby curbs; here localization relies solely on GPS, as is clear from the guassian-shaped yellow posterior distribution symmetrical about the center.

on human-driven logfiles, so such experiments do not pose a safety risk. We tested localization in a wide variety of environments, including freeways, major urban roads, side roads, and even dirt roads. As expected, results were best when lane markers were visible and/or when curbs were clearly defined. Even in cases where lane markers were faint or broken, or curbs were low, the localizer remained useful thanks to its probabilistic pipeline. We did encounter an example where the terrain was too difficult to reliably localize, but it was in an abandoned town with worn curbs and completely eroded lane markings.

More relevant as a strenuous benchmark was the 2007 Urban Challege, which consisted of several qualifying events during which the vehicle had to perform various autnomous driving missions, followed by the actual 60-mile race alongside both human-driven vehicles and other robots. This was an ideal demonstration because it was, by definition, unrehearsable.

During all qualifying events and the race itself, our vehicle's localization was reliable and precise. Parts of the course were exceedingly challenging from a localization perspective, affording little leeway for errors; many competing robots veered into an opposite lane, forcing opposing traffic to swerve away, and many robots even hit curbs on the side of the road. However, our vehicle had no obvious localization errors in any qualifying events or the race.

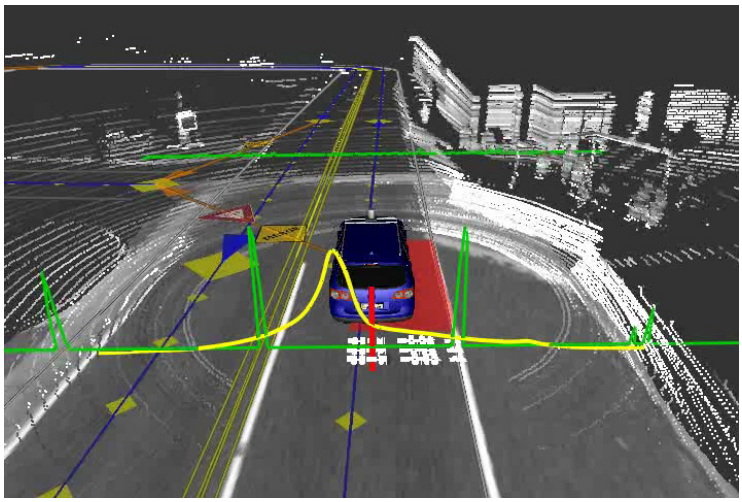


Figure 4.10: Road with strong lane markers; here localization deviates substantially GPS, causing the RNDF to shift by 75 cm. Without localization our car would have been driving partially into the opposing lane.

4.6.3 Quantitative performance

An analysis of the logfiles from the Urban Challenge events reveals extent of localization's influence on the vehicle.

In the National Qualifying Event (NQE) "A", localization resulted in an RMS RNDF/GPS alignment offset of 32 cm, with a maximum offset of 75 cm. This course was a loop which tested vehicles' ability to merge into moving traffic. Parts of the loop contained no lane markings or nearby curbs, which resulted in the localizer having no effect in those areas (see Fig. 4.9). In contrast, Fig. 4.10 shows our vehicle waiting to make a left turn across traffic, with the localizer shifting us 75 cm to the right. Relying only on the GPS estimate, our vehicle would have been substantially across the center lane marker and into the opposite lane, and we would have been severely penalized or disqualified.

In the NQE "C" event, which our vehicle completed in a brisk 18 minutes, localization had an even larger influence, causing a 52-cm RMS offset with occassional offsets of up to 120 cm. Proper localization in this event proved especially difficult for other vehicles, with the course containing faint lane markers and gentle curbs. Many vehicles repeatedly pushed into the opposite lane, and several drove (or attempted to drive) up the curb. The unusually large localization offsets resulting in this event may indicate that DARPA's RNDF for this event was of worse quality than their other maps.

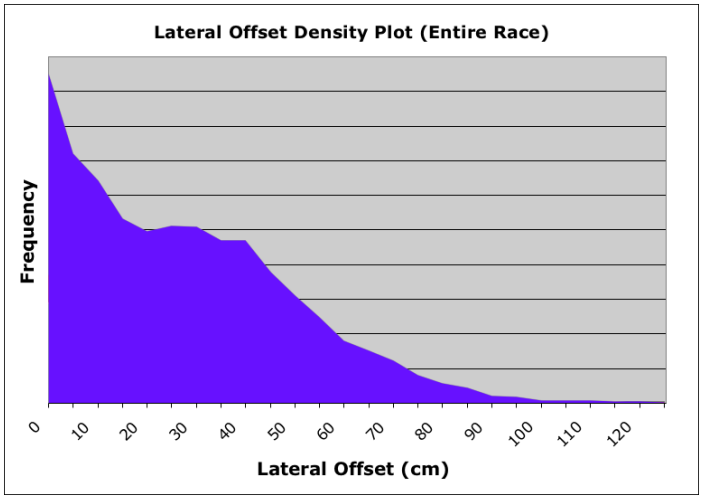


Figure 4.11: A histogram depicting the relative frequencies of various lateral offsets applied by the localization module during the 60-mile 2007 Urban Challege. Offsets of 0-80cm were common.

The actual race itself was a 60-mile course divided into three large missions. During the entire race, localization resulted in an RMS offset of 40 cm, with offsets of up to 130 cm. The frequency distribution of the various offsets throughout the race is shown in Fig. 4.11.

Although no visible localization error manifested itself during the race, a later analysis of the race logs indicated a single example where the lack of longitudinal localization caused our vehicle a delay of a few seconds. In this case, the GPS estimate of our vehicle's longitudinal position caused a static obstacle to appear as if it were in the lane our car was trying to turn into. After a few seconds, the vehicle's navigation planner decided to go around the obstacle anyway, without incident.

4.7 Conclusion

Reliable and precise localization will be one of the crucial components of future autonomous urban vehicles. Currently there exist several substantially different high-level approaches to tackling the problem: one could embed technology into the roads themselves, one could build a dense environment map, or one could use a compact vector map, as in the approach described here. All have significant advantages and drawbacks, but the undeniable appeal of the compact map approach is its flexibility and simplicity. It is interesting that in many

ways, the simpler the map, the steeper the requirements are for the system that localizes against it.

We have developed a robust and reliable localiation algorithm that feeds features extracted from laser data into several probabilistic constraints which together yield a full posterior distribution over possible lateral offsets. Extensive experimentation and field-tested results demonstrate the benefits of our approach. Although this method does not achieve the same absolute accuracy as the approach in the previous chapter, which requires laser pre-mapping of the terrain, it has proven to be sufficient for a variety of use cases.

By using lasers instead of cameras, we are able to avoid many of the pitfalls associated with traditional computer vision systems (e.g. shadows, ground extraction). By using a probabilistic model from start to finish, we avoid needing to pick arbitrary thresholds for what constitutes a lane marker or a curb, and are able to propegate subtle sensor data all the way to the final result. And as a component in a complete end-to-end autonomous vehicle, the fact that we have the full posterior distribution available is especially important, because it allows us to ensure that our localization output doesn't suddenly flip to another mode, which would likely cause problems for the navigation planner down the line. The fact that our localization approach enabled our vehicle to maintain its position in the center of the lane during the entire Urban Challenge race, and the fact that we were one of the few vehicles never to hit a curb in any qualifying events or the race, validates our approach.

Chapter 5

Unsupervised Calibration for Multi-beam Lasers

5.1 Introduction

Light Detection and Ranging (LIDAR) sensors have become increasingly common in both industrial and robotic applications. LIDAR sensors are particularly desirable for their direct distance measurements and high accuracy, but traditionally have been configured with only a single rotating beam. However, recent technological progress has spawned a new generation of LIDAR sensors equipped with many simultaneous rotating beams at varying angles, providing at least an order of magnitude more data than single-beam LIDARs and enabling new applications in mapping [53], object detection and recognition [20], scene understanding [76], and SLAM [70].

In order to effectively harness this massive increase in beam count, new calibration methods are required. First, calibrating angles and range readings for tens or hundreds of beams is a substantially harder problem than calibrating one or a few beams. Second, for applications that utilize the intensity returns of LIDAR sensors, e.g. recent mapping and localization work [53], it is important that the intensity remittance values agree across beams. In both cases, the number of parameters makes supervised measuring and calibration at best tedious and at worst infeasible.

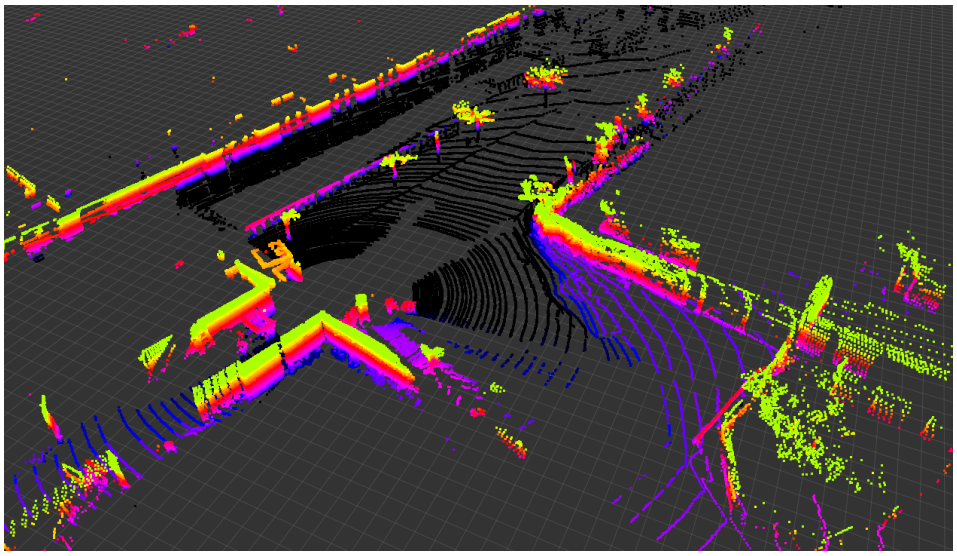


Figure 5.1: A single 360-degree scan from the 64-beam Velodyne LIDAR. Points are colored by height for visual clarity.

To date, there has been limited research on supervised calibration for multi-beam LIDARs. The most popular such unit as of this writing is the Velodyne HD-64E spinning LIDAR, which has been used extensively for many recent robotics applications. A representative scan from such a sensor is shown in Fig. 5.1. In [63], the authors present a supervised calibration technique for this LIDAR requiring a dedicated calibration target and many hand measurements, followed by a traditional optimization step. Indeed, the manufacturers of this laser built a dedicated calibration facility which they use to collect thousands of measurements followed by an unpublished optimization routine, in order to provide a calibration of each beam to the customer.

In the case of single-beam LIDARs, there have been attempts at unsupervised recovery of roll, pitch, and yaw in a known rectangular enclosure [8] as well as an attempt to estimate the noise parameters of a single-beam LIDAR [41]. Recent work has provided algorithms for calibrating one [84] or two [10] single-beam LIDARs on a moving vehicle platform using hand-placed retroreflective calibration targets, which additionally requires an intensity threshold for correspondences. [74] calibrate three single-beam LIDARs arranged vertically on a stationary horizontally rotating platform in an unsupervised manner, but they require that the individual units spin perpendicularly to the rotation of the base,

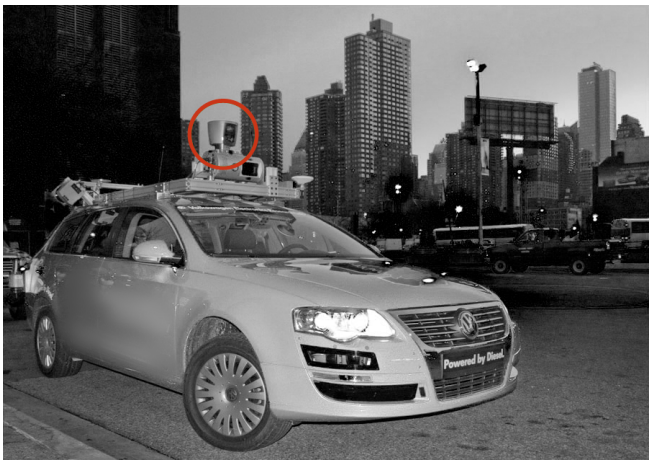


Figure 5.2: Our vehicle platform. Velodyne LIDAR unit is circled in red.

and their technique does not extend to a moving platform.

None of these techniques extends to the unsupervised calibration of a multi-beam LIDAR. Indeed, we are unaware of any algorithm in the literature to date that is able to recover the extrinsic pose of any LIDAR unit relative to a vehicle frame when it is the only such sensor on the vehicle and when the environment has no particular known features. In addition, we are similarly unaware of an existing algorithm that calibrates individual beam parameters for multi-beam units without hand-measured environmental features.

Although hand measurements are practical in some cases for single-beam sensors, especially for translational offsets, it is particularly difficult to measure sensor orientation with high accuracy in the absence of a dedicated calibration environment. Furthermore, for sensors with many beams, such measurements may take prohibitively long and would inevitably be suboptimal in accuracy. Finally, techniques that require specifically placed retroreflective targets with careful thresholding to pick them apart from the rest of the environment are not applicable in all settings a robot might encounter; they also result in the consideration of only a tiny fraction of the available data.

Thus, we propose a novel fully unsupervised extrinsic and intrinsic calibration method for multi-beam LIDAR sensors that requires no calibration target, no labeling, and no manual measurements. Given a multi-beam LIDAR attached to a moving platform or robotic vehicle along with accompanying inertial measurment unit (IMU) readings, our algorithm

computes hundreds of sensor parameters from only seconds of data collected in an arbitrary environment without a map.

Our contribution consists of three complementary unsupervised calibration algorithms. The first discovers the LIDAR's extrinsic 6-dimensional pose relative to the vehicle's inertial frame, including translation and rotation. The second estimates optimal vertical and horizontal angles for each individual beam and an additive distance offset for each beam's range reading. Finally, the third derives a fully Bayesian generative model for each beam's remittance intensity response to varying surface reflectivities in the environment.

In the sections that follow, we will describe each of the above algorithms conceptually. We will then provide details about our particular implementation of these techniques on a ground vehicle with a roof-mounted 64-beam rotating LIDAR, along with several results demonstrating the effectives of these algorithms even when presented with poor initial calibrations. Finally, we will discuss implications for related applications and possible extensions for future research.

5.2 Extrinsic Calibration

In the case of a multi-beam LIDAR, extrinsic calibration considers the mounting location of the entire unit relative to the vehicle's own coordinate frame, while intrinsic calibration considers the configuration of each individual beam inside the unit. In this section we present a method for extrinsic calibration, assuming a known intrinsic calibration.[1]

At the most basic level, our approach for both calibrations leverages the simple observation that laser returns projected into three dimensions are not randomly distributed in space. Indeed, because the returned points are reflections off of physical surfaces, it is impossible for a properly calibrated sensor traveling a known trajectory to return a collection of accumulated points that is randomly distributed in three dimensions. As such, the proposed method relies only on the weak assumption that points in space tend to lie on contiguous surfaces.

[1] If neither the extrinsic nor intrinsic calibration is known precisely, then the two separate calibration procedures can be performed iteratively until both converge.

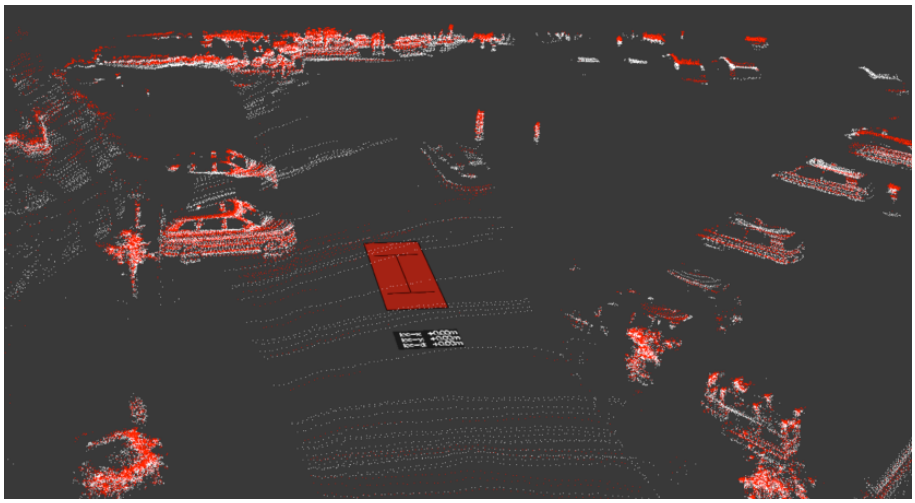


Figure 5.3: Velodyne points from two adjacent beams (of 64) accumulated over time and projected into 3D; one beam colored red, the other white. Due to known vehicle motion, both beams tend to see the same surfaces.

Consider Fig. 5.3, which depicts LIDAR returns from just two adjacent beams accumulated over several seconds of vehicle motion along a known trajectory. Here, we color one beam in red and the other in white; it is apparent that due to the LIDAR's movement through space, to a large extent both beams end up hitting the very same surfaces.

The location of the LIDAR unit relative to the vehicle's coordinate frame will be expressed with an x (longitudinal), y (lateral), and z (height) offset along with roll, pitch, and yaw angles. The (0, 0, 0) reference point and reference orientation is specified by the coordinate system being used, i.e. the three-dimension point and orientation that the vehicle's positioning system considers to be the origin.

In contrast to existing methods, our approach makes no assumptions about the environment other than that it is generally static and contains some 3D features, i.e. is not just smooth ground. In order to achieve an accurate calibration, we record LIDAR measurements as the vehicle transitions through a series of known poses.[2] Global pose information is irrelevant, as there is no existing map, so only local pose information is required. Local

[2]The vehicle trajectory can be arbitrary, but must include a change in yaw so that lateral and longitudinal offsets of the LIDAR can be detected; if the vehicle only moves straight, these cannot be disambiguated. Similarly, we do not attempt to recover the sensor's height, as our vehicle always remains nearly parallel to the ground, though height can trivially be determined by considering the distance to points the vehicle drove over.

pose data may be acquired in any number of ways, e.g. from a wheel encoder and IMU, from an integrated GPS/IMU system, or from a GPS system with real-time corrections. Again, it is only the relative motion of the vehicle through the trajectory that is relevant to the calibration, so no global pose data is necessary.

Now, we define an energy function on point clouds which penalizes points that are far away from surfaces defined by points from other beams:

$$J = \sum_{b_i=1}^{B} \sum_{b_j=b_i-N}^{b_i+N} \sum_{k} w_k \| \eta_k \cdot (p_k - m_k) \|^2 \tag{5.1}$$

where

- B is the total number of beams and N is the number of neighboring beams to which we align each beam
- k iterates over the points seen by beam b_j
- p_k is the kth point projected according to the current transform
- m_k is the closest point to p_k seen by beam b_i
- η_k is the surface normal at point m_k
- w_k is 1 or 0 based on whether $\|p_k - m_k\| < d_{max}$

This energy function bears similarity to the point-to-plane iterated closest point (ICP) error function [11], with two key differences. First, we compare surfaces defined by points in each beam individually against points in neighboring beams. This has the crucial benefit that an erroneous calibration between beams will not significantly affect surface normals in the set of points seen by any individual beam. Second, unlike in ICP, we are not dealing with rigid point clouds, as a change in any calibration parameter will transform the points in that beam's point cloud in a complex way, since in this case the points in each cloud were observed at different times and thus from different poses of the sensor.

Surface normals are computed separately per beam, by fitting a plane to each point's 20 nearest neighbors in the accumulated projected points from the entire trajectory. Due to the density of data from multi-beam LIDARs, this local neighborhood for each point is

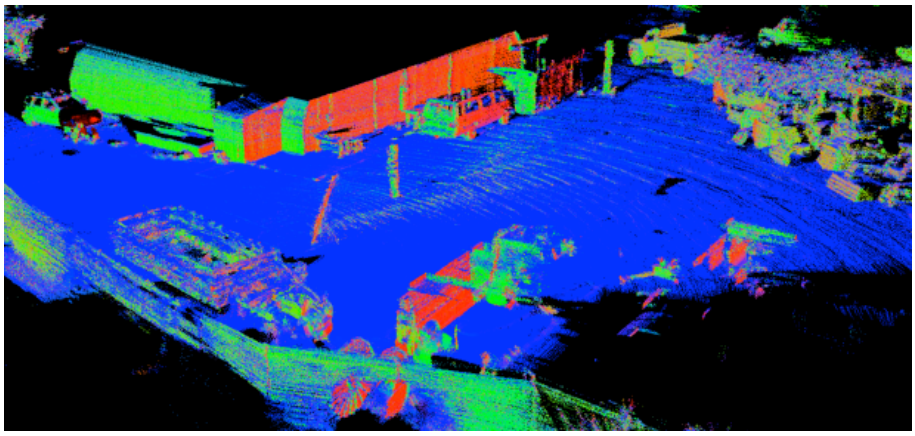


Figure 5.4: Accumulated points colored by computed surface normal; the red, green, and blue channels respectively are set to the surface's x, y, and z components of the normal vector at each point.

very small. We show an example of these surface normals in Fig. 5.4, where the red, green, and blue channels are colored according to normal vector's x, y, and z components at each point.

A further benefit of the high density of points returned by multi-beam LIDARs is that almost any surface will be nearly locally planar at the resolution of the pointcloud; thus, projecting points from one beam onto the surfaces defined by points in neighboring beams results in very low errors when all calibrations are accurate.

Given the above energy function, all that remains is to select the extrinsic calibration that minimizes the total score. Although in theory the objective is not necessarily convex, and thus finding the true global optimum cannot be guaranteed in any reasonable amount of time, in practice the energy function is quite smooth and standard search heuristics perform very well.

In our approach, we alternatively optimize the translation parameters and rotation parameters until both have converged. For each optimization, we utilize grid search, which compares the current energy score with the score that results from adjusting the variables in question in all possible directions jointly. Whereas a coordinate descent iteration takes time linearly proportional to the number of variables, grid search takes time exponential in the number of variables, as it considers all combinations of directions. On the other hand, it is

less prone to getting stuck in local minima, and as neither translation nor rotation individually has more than three variables, grid search is computationally tractable. For example, when considering a rotation change, each of roll, pitch, and yaw can be increased, held constant, or decreased, which results in 26 new comparisons to the current score.[3]

We start with a relatively large step size, iterate until convergence, and repeatedly reduce the step size until we've reached the finest granularity we desire. At the end of the last optimization, we obtain our final calibration parameters.

5.3 Intrinsic calibration of each beam

The motivation in the previous section applies equally to the case of intrinsic calibration. That is, an intrinsic calibration that computes each beam's horizontal and vertical angle and range offset correctly will necessarily yield a lower energy score than an incorrect calibration.

It is worth emphasizing that this property is a direct consequence of the fact that the vehicle moves during data collection. For a stationary vehicle, it is impossible to disambiguate certain calibrations; indeed, many possible angles and range offsets may be equally plausible in that case. But when the vehicle moves in a known trajectory, no longer will incorrect calibrations result in plausible pointclouds when each beam's returns are accumulated over time and projected appropriately into 3D space.

Although the energy function used to calibrate the sensor's extrinsic pose is equally applicable to its intrinsic calibration, it is intractable to perform grid search over 3 parameters for each of tens or hundreds of beams jointly. Instead, we alternately consider all horizontal angles, all vertical angles, and all range offsets until convergence. At each step, for the variables in question, we take empirical derivatives of the energy function across pairs of beams with respect to the individual parameters. Consider again the energy function:

[3] It is important to note that for every possible calibration considered, all points must be reprojected into 3D space based on the vehicle's pose at the time each point was acquired. Thus, unlike with ICP, a calibration change does not warp or distort all points in the same way, as the effect of a calibration change on each individual point depends on where the vehicle was at the time that return was measured. The projection of a laser return into a 3D point is computed simply by adding the return's distance, in the direction the beam points, to the laser's origin in 3D space; the laser's origin is located at the pose of the vehicle's reference frame at the time of measurement plus the extrinsic transform for the laser relative to the vehicle frame.

$$J = \sum_{b_i=1}^{B} \sum_{b_j=b_i-N}^{b_i+N} \sum_{k} w_k \|\eta_k \cdot (p_k - m_k)\|^2 \tag{5.2}$$

At each iteration, for each beam b_i and neighboring beam b_j we hold fixed the accumulated projected pointclouds and accompanying surface normals associated with beam b_j and then re-project the points from beam b_i with the parameter in question increased and then decreased by some increment α. For each of the two possibilities, the inner part of the energy function, $\sum_k w_k \|\eta_k \cdot (p_k - m_k)\|^2$ is recomputed; the parameter is then changed by α in whichever direction improves the objective maximally, or else the parameter is held constant if all perturbations are worse.

In this manner, we iteratively loop through all parameters and beams, optimizing the objective function at each step, until either some predetermined number of iterations is reached or until the change in the global objective function becomes sufficiently small. We note that although this heuristic works well in practice, unlike with grid search for extrinsic calibration, it is not actually guaranteed to lower the objective function in any given iteration, as it updates a particular parameter for all beams in each iteration. Given the extremely large search space, however, such approximations are reasonable, and, as we show in the results section, work very well in practice.

5.4 Remittance calibration

In addition to estimating the LIDAR's pose and beam parameters, we also calibrate each beam's intensity response to surfaces of varying reflectivity. We first present a deterministic unsupervised calibration algorithm that generates an input/output mapping for each beam's intensity response such that beams agree on the brightness of surfaces. We then extend the method to derive a Bayesian generative model of each beam's response using Expectation Maximization. [17]

5.4.1 Deterministic calibration

As in the previous sections, rather than requiring a fixed calibration target, we present an unsupervised calibration method that can be performed by driving once through an arbitrary environment. To compute the calibrated response functions for each beam, we project laser measurements from a logfile as the vehicle proceeds through a series of poses and, for every map cell, store the intensity values and associated beam ID for every laser hit to the cell. Then, for every 8-bit intensity value a observed by each of the beams j, the response value for beam j with observed intensity a is simply the average intensity of all hits from other beams to cells for which beam j returned intensity a.

Specifically, let T be the set of observations $\{z_1, \ldots, z_n\}$ where z_i is a 4-tuple $\langle b_i, r_i, a_i, c_i \rangle$ containing the beam ID, range measurement, intensity measurement, and map cell ID of the observation, respectively. Then we have:

$$\begin{aligned} T &= \{z_1, \ldots, z_n\} \\ z_i &= \langle b_i, r_i, a_i, c_i \rangle \\ b_i &\in [0, \ldots, 63] \\ r_i &\in \mathbb{R}_+ \\ a_i &\in [0, \ldots, 255] \\ c_i &\in [0, \ldots, X \cdot Y - 1] \end{aligned}$$

where X and Y are the dimensions of the map. Then the calibrated output $c(a, j)$ of beam j with observed intensity a is computed in a single pass as:

$$c(j,a) := \mathbb{E}_{z_i \in T}[a_i \mid ((\exists k : c_i = c_k, b_k = j, a_k = a), b_i \neq j)] \tag{5.3}$$

That is, the calibrated output when beam j observes intensity a is the conditional expectation of all other beams' intensity readings for map cells where beam j observed intensity a.

This equation is computed for all 64x256 combinations of j and a. Thus a calibration file is a 64-by-256 intensity mapping function; values which are not observed directly can be interpolated from those which are. These calibrated response functions need only be

computed once and their results can be stored compactly in a lookup table.

The quality of the calibration may be improved by iteratively applying the calibration procedure. However, care must be taken to ensure that the beam responses do not converge towards a uniform gray intensity, which is unfortunately the optimal way to ensure that beams agree. Instead, we recommend iterating between applying the preceding calibration routine and then expanding the response histogram of each beam to more closely match the overall response statistics of the laser.

In contrast to many calibration algorithms, our unsupervised approach has the desirable property that it does not require a particular calibration environment, but instead adapts automatically to any environment. Due to the abundance of laser data and the averaging over many values, even the presence of dynamic objects does not significantly reduce the quality of the calibration.

5.4.2 Bayesian calibration

Taking this further, we can derive a probabilistic calibration that specifically models the uncertainty and noise characteristics of each beam, which in practice are often quite different. As in the deterministic case, we don't have the ground truth for the brightness of the surfaces we are measuring, so the true intensities for each map cell remain latent variables.

We note that although environment reflectivity (intensity) is of course continuous, for computational reasons we restrict the values to integers between 0 and 255, as this matches the range of the Velodyne's returns. Thus for each map cell c we maintain a discrete distribution $P(m;c)$ indicating the probability that map cell c has intensity m, for $m = 0$ to 255.

Now, for each beam b we wish to estimate a distribution $P(a|m;b)$ indicating the probability that beam b will return intensity a given that its measurement comes from a map cell with intensity m. Each map cell is initialized with a uniform intensity prior, and for each beam we initialize the prior

$$P(a|m;b) = \eta \cdot [\exp\left(\frac{-(a-m)^2}{2\tau}\right) + \varepsilon] \tag{5.4}$$

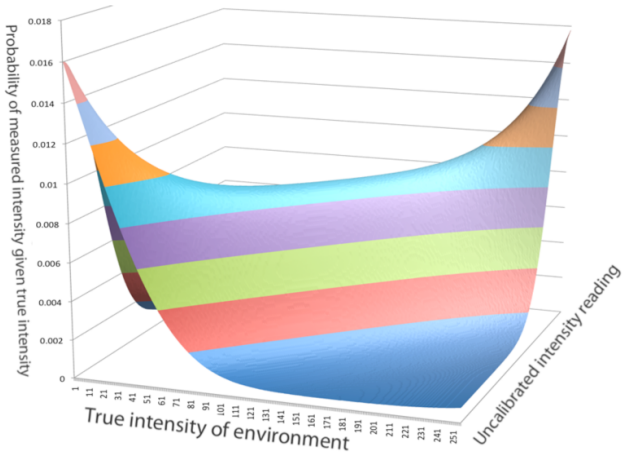


Figure 5.5: Bayesian prior for beam remittance response function as a function of surface intensity, used to initialize EM.

where η is the normalizer, τ controls the peakiness of the distribution, and ε affords a nonzero probability of a random intensity return. Note that while each beam maintains its own distribution $P(a|m;b)$, all beams are initialized with the same 256 x 256 distribution; with this initialization, *a priori* beams are likely to return values near the true brightness of the map, as shown in Fig. 5.5.

Let us define θ as the beam model, where θ is the collection of $P(a|m;b)$ for all 64 beams. Thus, in our implementation, θ is a 64 x 256 x 256 table. Similarly, let us define M as the intensity distribution over map cells, where M is the collection of $P(m;c)$ for all cells in the entire map. Thus, in our implementation, M is a 256 x $|M|$ table, where $|M|$ is the number of cells in our map. We note that after calibration, M can be discarded, as it is ultimately our beam model θ that we are interested in computing.

Starting with the initializations above we alternate between computing $P(m;c)$ for each map cell (E-step) and computing $P(a|m;b)$ for each beam and map intensity (M-step). We note that while the intensities of each map cell are by no means independent of each other, because they are jointly affected by the beam models, if we make the approximation of ignoring spatial correlations between nearby map cells, then they become conditionally independent of each other given the beam models. Since we are ultimately interested in deriving the beam models rather than the map, and given the abundance of data, ignoring potential spatial correlations between map cells is feasible and makes the problem tractable.

The update equations are as follows.

E-step

The E-step updates each map cell independently:

$$P(m_k = m|z;\theta) = \eta \cdot \prod_{i:c_i=k} P(a_i|m;b_i) \tag{5.5}$$

Thus, in the Expectation step we compute the distribution over intensities for each map cell given the sensor data and holding the current beam parameters constant.

M-step

The M-step updates the beam model for each beam independently. First, we compute:

$$C(m,a;M,b) = \sum_{k=1}^{K} P(m_k = m) \cdot \mathbf{1}\{\exists i : b_i = b, c_i = k, a_i = a\} \tag{5.6}$$

where $C(m,a;M,b)$ is the expected count in the data of beam b observing intensity a for cells of map intensity m, given the distribution over map cells in M computed in the previous E-step. Then, using these expected counts, we compute the maximum a posteriori (MAP) estimate for the beam parameters:

$$P(a|m;M,b) = \eta \cdot C(m,a;M,b) \cdot P(a;b) \tag{5.7}$$

where $P(a;b)$ is the prior probability of beam b observing intensity a. Using Bayes' rule, this equation computes the distribution over possible intensity return values for each beam given the distribution over map cell intensities. Thus, in the Maximization step we obtain the most likely beam parameters given the sensor data and holding the current distribution over the intensities for each map cell constant.

After EM converges, we obtain a fully generative model for each beam's response to environment surfaces of different reflectivities.

5.5 Experimental Results

We demonstrate the performance of the calibration algorithms presented here with several experiments. We used a Velodyne HD-64E S2 LIDAR sensor with 64 beams oriented between -22 to +2 degrees vertically and rotating 360 degrees horizontally. The unit spins at 10Hz and provides around 100,000 points per spin. This sensor was mounted to the roof of our research vehicle as shown in Fig. 5.2.

In addition, our vehicle pose was provided by an Applanix LV-420 positioning system that combines GPS, IMU, and wheel encoder readings to provide global and intertial pose updates at 200 Hz. However, as the methods described here only require locally consistent pose data, we ignored the global GPS pose values and only used the unit's local 6-DOF velocity updates, which we integrated over time to produce a smooth local coordinate frame.

We implemented our algorithms in C, taking advantage of the University of Maryland's Approximate Nearest Neighbor (ANN) library [62]. For the following results, we used a maximum matching distance of 20cm and generated per-beam pointclouds and surface normals using all laser returns, but only evaluated the energy function at every 16 points for effiency; with over a million points returned by the Velodyne per second, it is unnecessary to evaluate the energy function at every single point. With this implementation on a modern desktop processor, using about 15 seconds of recorded data, the extrinsic and intrinsic calibrations each take on the order of one hour to converge, given a very bad initialization. The remittance intensity calibration is faster, requiring a few minutes to run.

5.5.1 Extrinsic calibration

First, we show that we can precisely and reliably compute the Velodyne's mounting location on our vehicle, even with a poor initialization. As discussed previously, we attempt to recover the sensor's lateral and longitudinal offset and roll, pitch, and yaw relative the the vehicle's coordinate frame.

We collected two short 15-second logfiles with each of two different mounting positions for the Velodyne, for a total of four logfiles. In all four cases, we drove the vehicle in a tight semi-circular arc close to a building at 2 m/s.

First, we ran the calibration routine on the two logfiles taken with the initial mounting

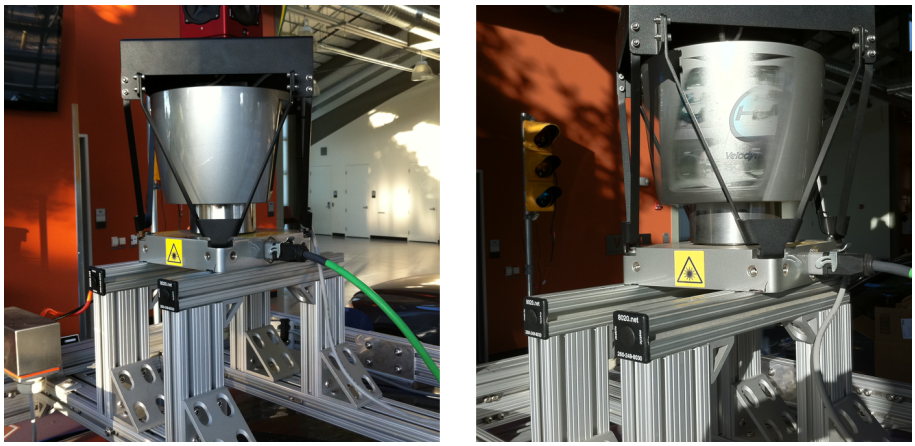


Figure 5.6: Initial Velodyne mounting position (left) and after translation and rotation (right)

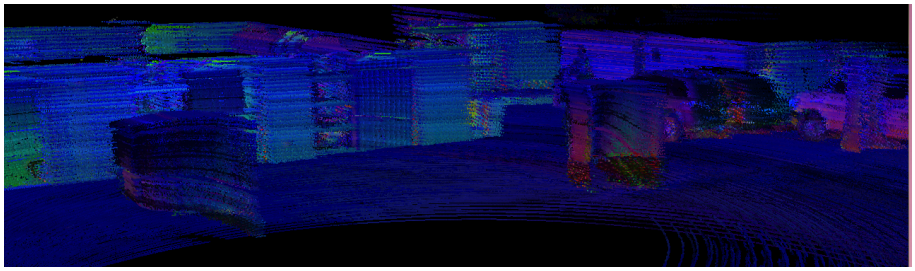
position. Here, the Velodyne sensor was centered laterally on the vehicle, and positioned 1.51 meters forward of the rear axle, which is our positioning system's reference origin. It was pointed straight forward and mounted parallel to the roof rack, as shown in Fig. 5.6 (left).

On the first logfile, starting with an initial calibration that was within 10cm and 1° of the measured location, our algorithm computed a lateral position of 0.00m, a longitudinal position of 1.51m, a roll of -.03°, a yaw of 0.00°, and a pitch of -.46°. On the second logfile, with the Velodyne in the same location, the algorithm computed a lateral position of -0.01m, a longitudinal position of 1.50m, a roll of -.03°, a yaw of 0.03°, and a pitch of -.46°. Thus, from two separate drives, from two different locations, the resulting position estimates were within 1 cm and .03° of each other in all dimensions.[4]

For a more challenging test, we then remounted the Velodyne 5.6 cm to the right and 20.6 cm behind the original location, and we rotated it counter-clockwise (around the Z axis) by 9 to 10°, as shown in Fig. 5.6 (right). Now, starting with the calibrated pose from the original mounting location, we ran our algorithm on each of the two new logfiles. On the second of the new logfiles, it correctly estimated that the sensor had been moved by 6 cm to the right and 21 cm to the rear, and rotated by 9.78° counter-clockwise. The dramatic

[4]We are only able to measure the mounting angles to within 1°, so we cannot empirically verify the angle calibration to within the .03° granularity to which we estimate angles in our algorithm; however, both the energy functional and the surfaces in the resulting 3D pointclouds are optimal with the computed calibration and degrade noticeably if they are changed in either direction.

(a) Assuming previous Velodyne position.



(b) After calibrating for the updated position

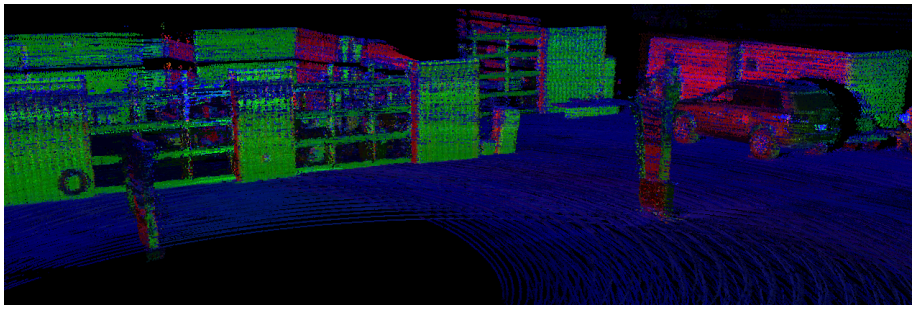


Figure 5.7: Moving and rotating the Velodyne, but continuing to use the original position calibration, projecting accumulated Velodyne points into 3D results in massively deformed structures and significant blurring of surfaces (a). After calibration (b), the points are much better aligned as a result of the improved objective function, and the computed transform for the Velodyne is extremely accurate, as verified by hand measurements. Points are colored by both surface normal and intensity return.

improvement in the resulting 3D pointcloud, comparing the assumed original mounting location to the estimated new location, is shown in Fig. 5.7.

On the first of the new logfiles the estimate was less accurate; the computed offsets were 16cm to the right and 30cm to the rear, along with 9.56° counter-clockwise. Thus, although the directions of the movement were correct, the amounts were not. Upon examining this logfile, we discovered that there had in fact been IMU drift across the trajectory, which resulted in the accumulated 3D pointcloud showing visible smearing. Indeed, the inaccurate Velodyne pose estimate our algorithm computed not only had a better scoring energy function than the correct calibration for that logfile, but it also visually resulted in less smearing than the correct calibration. In this case, our algorithm picked the extrinsic calibration that resulted in the "best" 3D pointcloud, but because the assumption of a correct local trajectory was violated, the estimated calibration was a bit off.

Therefore, these results demonstrate that our algorithm produces extremely accurate

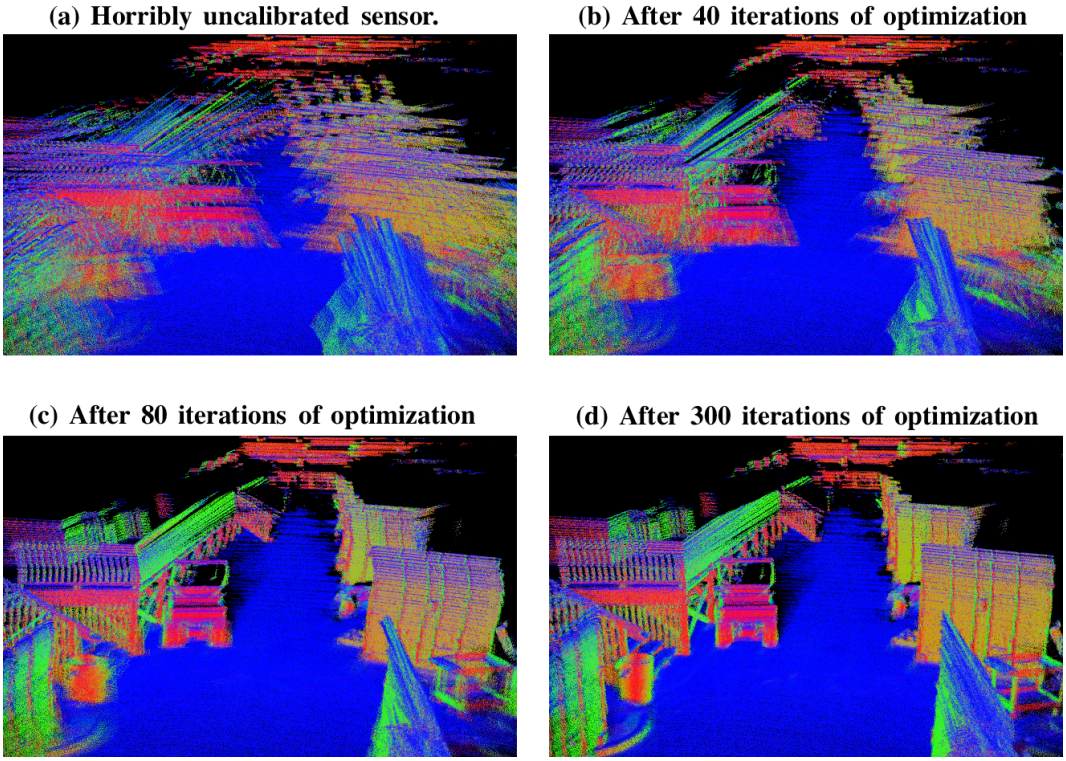


Figure 5.8: Unsupervised horizontal angle and range calibration using 10 seconds of data (all scans depicted above). Points colored by surface normal. Even starting with an unrealistically inaccurate calibration (a) we are still able to achieve a very accurate calibration after optimization (d).

extrinsic pose estimates when the trajectory is known precisely, and that its performance degrades when the trajectory estimate is inaccurate.

5.5.2 Intrinsic calibration

Next, we show that we can accurately calibrate the Velodyne sensor's individual beam angles and range offsets. Recent improvements in Velodyne's meticulous supervised factory calibration give better results than earlier models; not only can our algorithm do better still, we show that we are able to take an artifically bad calibration and use our methods to arrive at a calibration whose accuracy exceeds the best available factory calibration. Fig. 5.8 depicts such an optimization; starting with an unrealistically bad calibration in which we

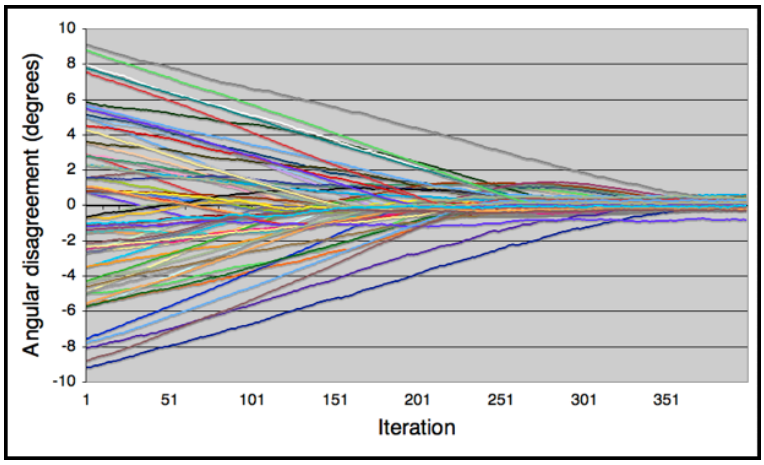


Figure 5.9: Comparing the horizontal angles of all 64 beams with the factory calibration, starting with a uniform initial estimate and optimizing over 400 iterations. Here, the update step size was very low for clarity. Initially beams were miscalibrated by up to 9°, and after calibration all beams' angles agreed with the factory calibration to within 1°, with an RMS deviation of only .25°.

set all horizontal angles to be 0 and all range offsets to be equal,[5] we are able to recover an excellent calibration based on only 10 seconds of data in a complex unlabeled environment.

We show quantitative success in two ways. First, in optimizating the intrinsic calibration, the angles and range offsets we compute are very similar to the factory values, even when initialized to be significantly different. In one experiment, we optimize the horizontal angles starting from a uniform initial estimate of 0° per beam. With this initialization, the beam angles disagree with the factory calibration by up to 9°, with an RMS disagreement of 4.6°; after our optimization, the maximum disagreement is less than 1°, with an RMS disagreement of 0.25° (Fig. 5.9). Thus our result is, for all beams, very similar to the factory calibration. Indeed, although we are unable to measure angles to these tolerances, we find the resulting energy functional and visual appearance of our calibration both outperform the factory calibration, suggesting that much of the disagreement may be due to factory miscalibration, or slight angle shifting over time.

Going further, in Fig. 5.10 we see another application of our calibration in a simpler environment, after which we hand-selected an area of horizontal ground and vertical wall known to be planar, and in Fig. 5.11 we compare the RMS distance of the points to their

[5] In reality, the horizontal beam angles range from -9° to +9° within the Velodyne and the range offsets range from 0.85m to 1.53m.

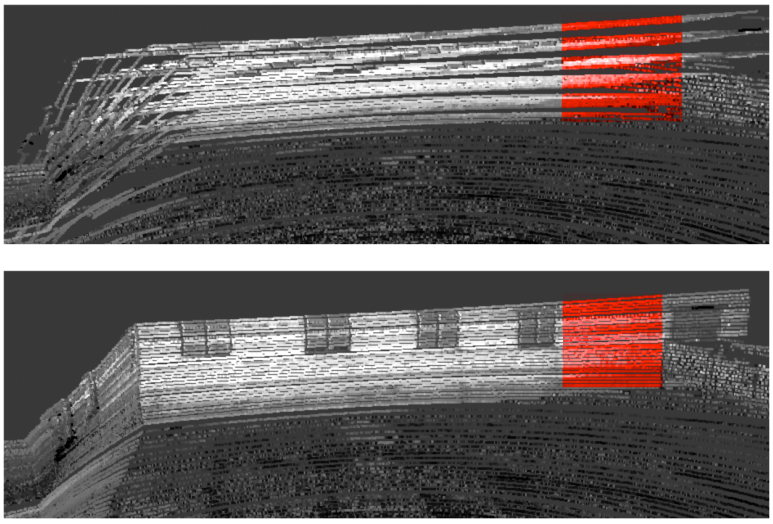


Figure 5.10: Improvement in wall planarity with calibration.

approximating plane over the course of optimization. With our artificially bad calibration we start with ground and wall errors of 54 and 4 cm, respectively, and after optimization these are reduced to 4 and 2 cm, respectively. Importantly, the baseline Velodyne factory calibration gives errors of 6cm and 3cm, respectively, so our unsupervised method provides superior results.

5.5.3 Remittance calibration

The results of the deterministic intensity calibration are shown in Fig. 5.12. We see that each beam has a unique intensity response; beams whose curves fall above the line $y = x$ are being brightened as a result of calibration, while beams whose curves fall below that line are being darkened.

The results of Bayesian intensity calibration can be seen in Fig. 5.13, in which we plot the learned generative model for one of the 64 beams. Here we see, as expected, that brighter surfaces tend to yield brighter returns. The somewhat surprising non-monotonicity of the graph corresponding to the brightest 45% of surfaces may be partially explained by the fact that fewer than 0.1% of the Velodyne returns fall into that brightness region. Thus, there is very little data to generate that section of the graph; at the same time, this

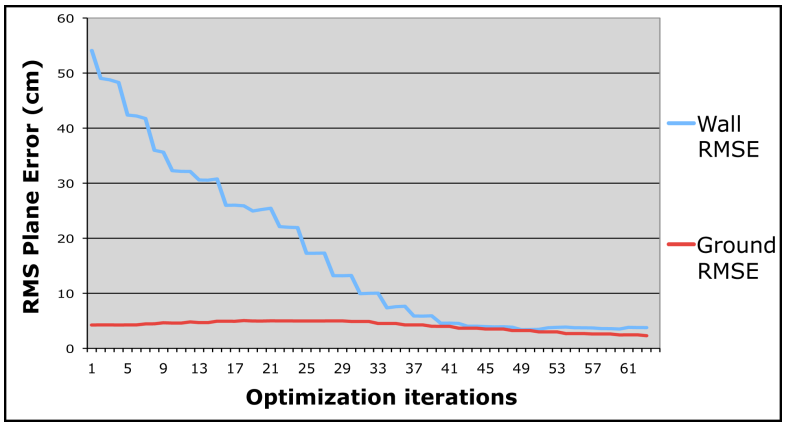


Figure 5.11: Quantitative improvement in wall planarity during calibration procedure. Final result exceeds factory calibration for both wall and ground.

phenomenon causes that region of the response function to rarely be queried.

Finally, we show the significant impact of angle, range, and intensity calibration on the resulting laser maps. In Fig. 5.14 we show an orthographic intensity map of points projected to the ground plane, first with our artifically bad calibration, then with calibrated angles and ranges, and lastly adding calibrated intensities. The final calibrated map displays excellent sharpness and contrast, indicative of a well-calibrated sensor.

5.6 Related extensions

The algorithms presented in this chapter for intrinsic and extrinsic calibration of multi-beam lasers are broadly applicable to a wide range of related problems. In this section, we discuss two such examples. First, we show that our same objective function can be successfully used to calibrate a traditional single-beam laser mounted on the same vehicle as a multi-beam laser. Second, we demonstrate that the objective function can also be used to calibrate time delays between sensors. At a high level, the fact that our calibration methods are so extensible is a testament to the power and simplicity of the fundamental insight that well-calibrated sensors must agree on the locations of surfaces in a static environment.

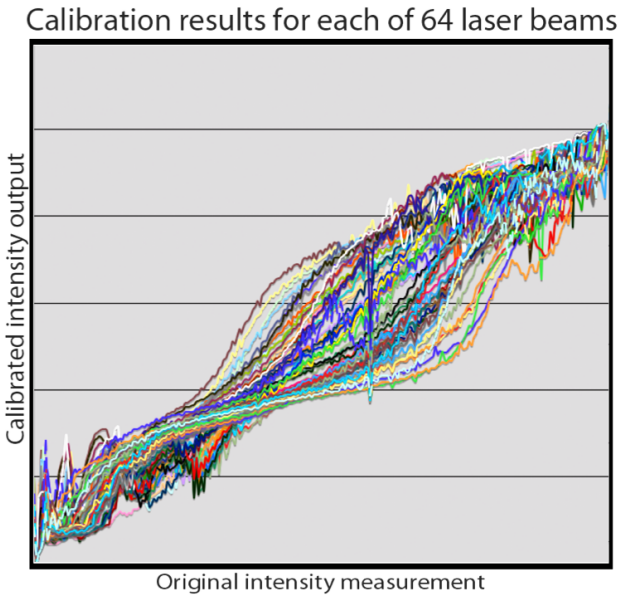


Figure 5.12: The expected environment intensity given each beam's intensity return. All 64 beams are shown here; note significant variation between beams. The noise in the darkest and brightest returns is due to the extremely low fraction of returns that fall in that region.

5.6.1 Single-beam laser calibration

Although multi-beam lasers can generate vastly more data than single-beam lasers, it often remains desirable to include one or more single-beam lasers on a robotic platform that also includes a multi-beam laser. These single-beam lasers, due to their typically smaller size and relative affordability, may be used for redunancy, to fill in blind spots not covered by the multi-beam laser, or because they have a higher maximum range. Indeed, on our autonomous vehicle we employ several single-beam laser rangefinders for all of those reasons. Naturally, it would be helpful to be able to automatically calibrate their positions relative to the vehicle frame, and, implicitly, relative to the Velodyne sensor.

Once the Velodyne's intrinsics and extrinsics have been calibrated using the methods described in this chapter, we can then use the same objective function and optimization approach detailed in section 5.3 to align one or more single-beam lasers to the Velodyne. The algorithm is straightforward: simply project all of the Velodyne points into 3D, and, keeping their locations and surface normals fixed, minimize:

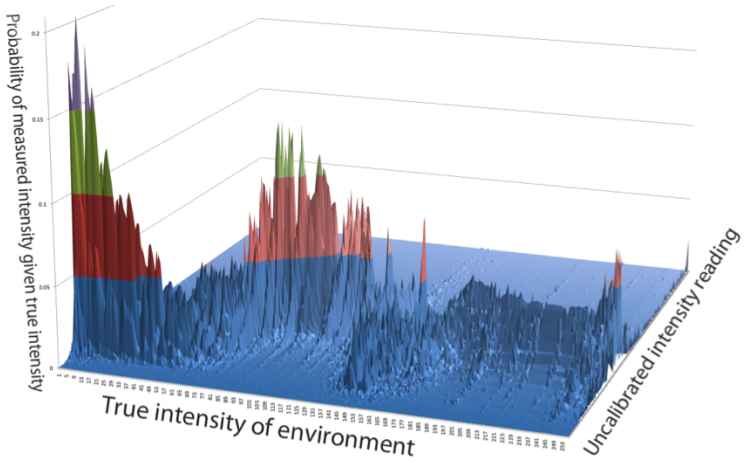


Figure 5.13: The learned generative model $P(a_i|m)$ for beam 37 of 64.

$$J = \sum_{b_i=1}^{B} \sum_{k} w_k \|\eta_k \cdot (p_k - m_k)\|^2 \tag{5.8}$$

- where B is the total number of beams in the multi-beam laser,
- k iterates over the points seen by the single-beam laser,
- p_k is the kth point projected according to the current transform,
- m_k is the closest point to p_k seen by beam b_i in the multi-beam laser,
- η_k is the surface normal at point m_k
- w_k is 1 or 0 based on whether $\|p_k - m_k\| < d_{max}$

using grid search as described previously. Once J is minimized, we should have arrived at the correct transform for the single-beam laser.

An example is shown in Fig. 5.15. Here, we calibrated one of the SICK LDLRS single-beam laser rangefounders mounted on the side of our car, which is over 2m away from the Velodyne. We began with the calibrated Velodyne parameters and an initial estimate of the LDLRS which was off by 40cm in the (x,y) plane and 3.5° in yaw. After optimization, the algorithm recovered the translational and rotational calibration error, bringing the LDLRS points into nearly perfect alignment with the Velodyne points.

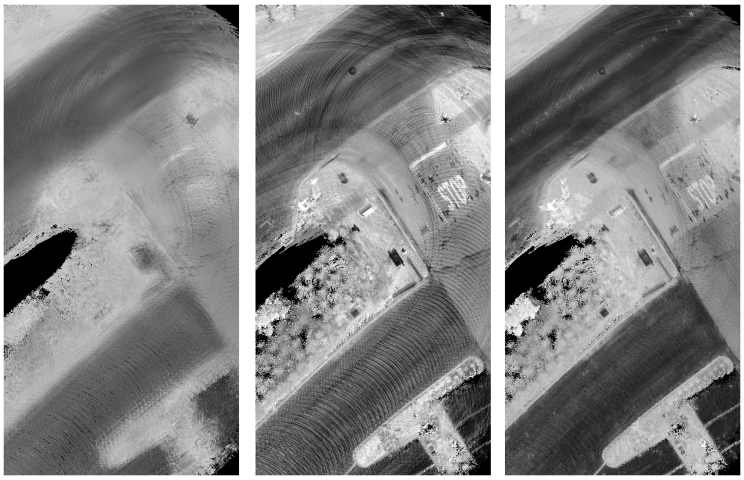


Figure 5.14: Improvement from calibration. We compare an orthographic intensity map of a street with the horrible angle and range calibration used in Fig. 5.8 (left), the same map with the learned angles and ranges (center), and finally adding intensity calibration (right). The final result is much improved.

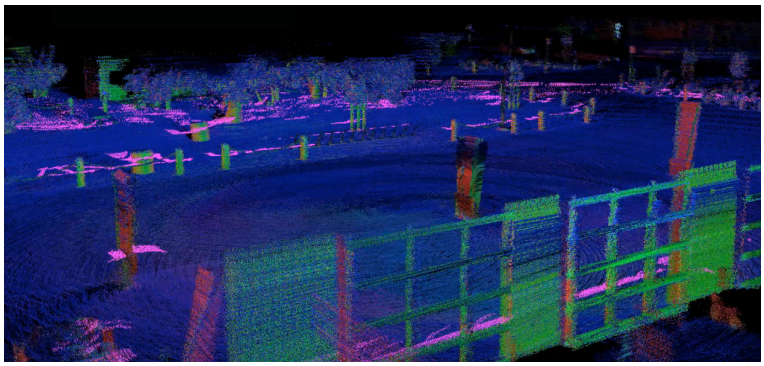
5.6.2 Calibrating time delays

An additional use of these calibration techniques is the recovery of arbitrarily sized time delays between sensors. Suppose readings from the vehicle's positioning system arrived, or were recorded, later or earlier than they should be relative to a laser sensor. In this case, the projection of the laser points based on the laser readings and the incorrect positions of the vehicle (caused by the timestamp disagreement) would result in a lower objective function score than would be achieved using the real time delay between the sensors.

Consequently, we can simply probe various possible time delays and select the time delay that minimizes the objective function when all points are projected according to that delay. An example is shown in Fig. 5.16. Here, we show a projected pointcloud with an artificially induced time delay of 0.1 seconds, compared to the pointcloud derived without a delay. Unsurprisingly, the time delay results in a significant blurring of the pointcloud.

Although our laser and positioning system share a hardware timestamp and thus do not suffer from meaningful time delays, it is straightforward to synthetically test the algorithm by artificially imposing a constant time delay between the sensors, and then asking whether the algorithm can recover the selected delay. To test this hypothesis, we took the scene shown in Fig. 5.16 and added a 0.05-second time offset between the Velodyne and

(a) Uncalibrated LDLRS laser (pink) with Calibrated Velodyne



(b) Both sensors calibrated

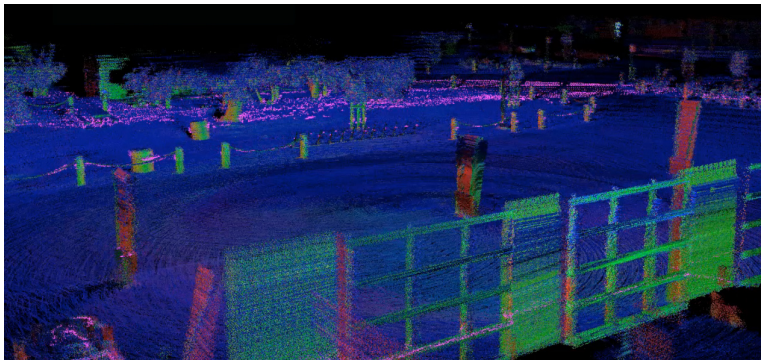


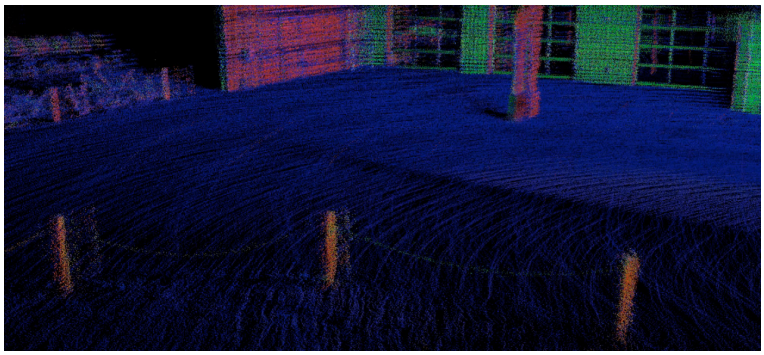
Figure 5.15: Using an initial estimate for the single-beam LDLRS calibration (a), the projected LDLRS points do not align well with the calibrated (and much more abundant) Velodyne points. After calibration, all scans are in good alignment, so much so that it is often difficult to spot the pink LDLRS points amongst the Velodyne points.

Applanix; this delay corresponds to 10 Applanix messages. We then probed the objective function at time offsets from -.1 to +.1 seconds; the objective function is graphed in Fig. 5.17. The graph is smooth with a clear minimum at .05 seconds, thus successfully recovering the delay.

5.7 Conclusion

Multi-beam LIDAR sensors are a key enabling component of advanced mapping and robotic applications, and their use will only increase with time. We have presented what we believe to be the first complete fully unsupervised calibration algorithms for these sensors,

(a) Artificial time delay of 0.1 seconds



(b) No time delay

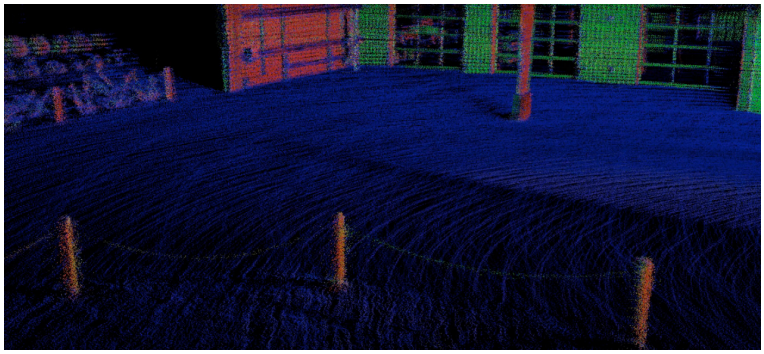


Figure 5.16: An artificial time delay between the Velodyne laser and Applanix position system results in a blurry pointcloud (a), in comparison to the correct pointcloud with no time delay (b).

and have demonstrated that excellent results are achievable with a trivial amount of data collection in arbitrary unlabeled environments even with terrible initializations. In addition, these methods are extensible and have many related applications. For example, these algorithms could easily be extended to the calibration of multiple single-beam LIDARs mounted to one vehicle platform.

Due to the particularly large amount of data generated and extremely large search space, the algorithms discussed here are naturally suited to be run offline. However, it is conceivable that with intelligent data pruning and more aggressive search techniques, a similar realtime algorithm could be developed to enable on-the-fly sensor calibration while driving. For most applications, offline calibration is sufficient, but for sensors that are unable to be perfectly secured to the vehicle, a realtime algorithm would provide some benefits.

Although the algorithms we discuss here make particularly few assumptions about the

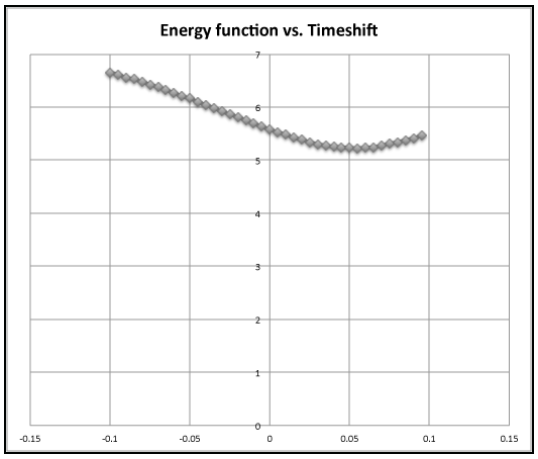


Figure 5.17: The calibration objective function as a function of the time delay between the Velodyne and Applanix. There is a clear local minimum at .05 seconds, which is exactly the artificially induced time offset between the two sensors.

environment, they do treat the environment as static, i.e. we assume it is only the data collection vehicle that is moving. To the extent that there are overwhelming dynamic obstacles during data collection, this motion could interfere with the results; in these cases, existing segmentation and tracking algorithms could be employed [49, 71] to remove such tracks, although an especially poor initial calibration may render segmentation and tracking more difficult than usual. At the same time, normal spurious dynamic obstacles have been seen to have essentially no effect on the results, as they represent a small fraction of the overall points seen and thus have little effect on the shape of the objective function.

In this chapter we presented a solution to a particular instance of the Simultaneous Calibration and Mapping (SCAM) problem, in which neither a calibration nor a map is available *a priori*. Such situations are in fact common in practice, despite being significantly less studied than the more popular Simultaneous Localization and Mapping (SLAM) problems. This discrepancy is perhaps due in part to the fact that reasonable calibrations for simple sensors are often obtainable by hand measurement, whereas SLAM problems cannot be solved similarly. In addition, the SCAM solution presented here is inapplicable to single-beam sensors on their own as they alone do not provide enough data for fully unsupervised calibration in the general setting.

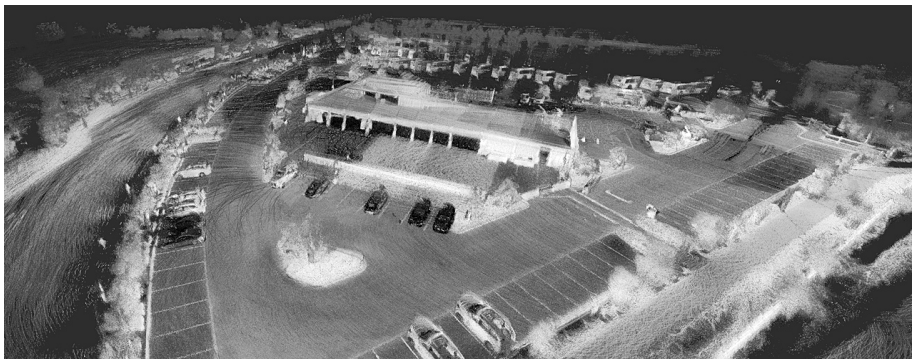


Figure 5.18: 3D pointcloud of a campus parking lot after SLAM and calibration; 60 seconds of accumulated data colored by intensity return.

Figure 5.19: Closeup from above scene from another perspective; points colored by both intensity return and surface normal.

A natural extension would be to combine SCAM with SLAM; that is, to solve Calibration, Localization, and Mapping jointly when none are known precisely. In preliminary work, we have aligned a several-minute logfile with both SLAM and SCAM as presented here; a resulting 3D pointcloud can be seen in Fig. 5.18. However, more research is required to arrive at a consistent approach to jointly optimizing all unknowns in the general case, particularly if the initial pose estimate is poor. The precision of our unsupervised extrinsic calibration - moreso than the rest of the algorithms presented in this dissertation - benefit from a high-end IMU, which is not available or practical for all robots. Thus, a joint algorithm for recovering calibration and localization without a known map would be a worthwhile goal for future research.

Chapter 6

Traffic Light Mapping and State Detection

6.1 Introduction

Reliably detecting the state of traffic lights, where $state \in \{red, yellow, green\}$, is essential for autonomous driving in real-world situations. Even in non-autonomous vehicles, traffic light state detection would also be beneficial, alerting inattentive drivers to changing light status and making intersections safer. Non-passive traffic light systems that broadcast their current state and other information have been demonstrated in industry and academic settings [31], and can reliably provide exact state information to vehicles. However, such active systems require expensive hardware changes to both intersections and vehicles, and thus have yet to materialize in any significant market. Thus, in the domain of autonomous driving, reliance on non-passive systems for conveying traffic signal information is not currently feasible, as it requires new infrastructure.

While prior work has demonstrated proof of concept in camera-based traffic light state detection, in recent years the data rate and accuracy of the sensors required for passive traffic light state detection has increased, and supporting sensors such as GPS have improved to the point where such a system could conceivably be operated safely in real-time. The key to safe operation is the ability to handle common failure cases, such as false positives and transient occlusion, which arise frequently and test the limits of camera-only methods.

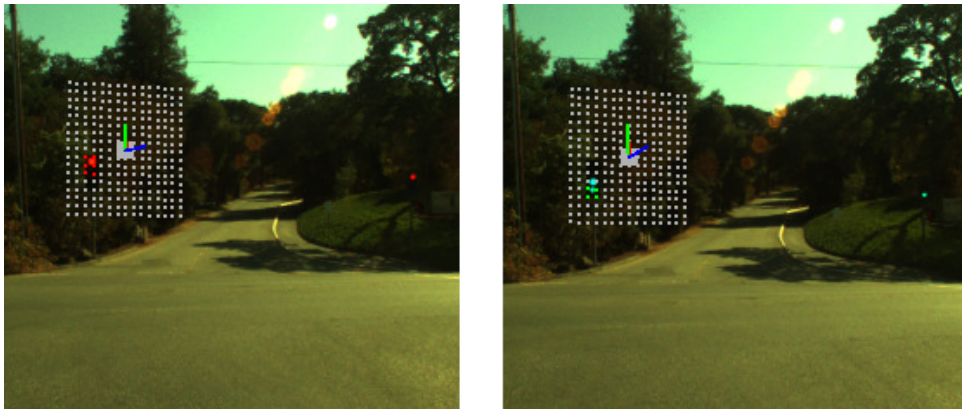


Figure 6.1: This figure shows two consecutive camera images overlaid with our detection grid, projected from global coordinates into the image frame. In this visualization, recorded as the light changed from red to green, grid cells most likely to contain the light are colored by their state predictions.

To overcome the limitations of purely vision-based approaches, we take advantage of temporal information, tracking and updating our estimate of the actual light location and state using a histogram filter. To somewhat constrain our traffic light search region, we pre-map traffic light locations and therefore assume that a prior on the global location of traffic lights is available during detection, utilizing pose data available from a GPS system.

Approaching this problem with prior knowledge of the traffic light location has been mentioned in previous work [39], though the implications of such an assumption, including specific methods for dealing with sources of error, have not be thoroughly explored. One contribution of our work is a principled analysis and modeling of the sources of error in each stage of our traffic light state detection, and the creation of a general framework for evidence gathering in a camera and global object prior system.

With respect to our system, global coordinates of each relevant traffic light are quickly obtained, as discussed in Section 6.2 during a pre-drive of the course. Such prior knowledge also facilitates the assignment of traffic lights to specific lanes of the road, which is required for efficient route planning and safe driving.

As we do in this chapter, several approaches in the literature have focused on camera-based detection of traffic lights. In 1999, [24] described general methods for detection of traffic lights and other urban-driving information, but lacked the sensors and processing power to completely test and deploy their system. In [72] images are analyzed based on

color similarity spaces to detect traffic lights, but the method does not run in real-time. Somewhat similar to our image processing algorithm, the detector in [64] biases toward circular regions of high intensity surrounded by regions of low intensity to represent a light lens inside a darker frame. Their work also attempts to report the state of the light; however, their method does not reliably differentiate between multiple lights, or determine the state of more than one light in a single camera frame.

The experiments of [39] show that traffic light detection is possible in real-time, and that a traffic light detection system can be used in conjunction with a GPS-based navigation system. Although the methods in this work are effective in detecting lights, they do not clearly define a way to avoid false positives in their detection. They also utilize a grid-based probabilistic weighting scheme to estimate the location of a light in the current frame based on its location in the previous frame. Unlike our method, however, they define their grid in image-space, where we define our grid in global coordinates, allowing for motion in image-space due to perspective changes, which may occur as a vehicle approaches a light. The method of [57] uses a three-stage pipeline consisting of a detector, tracker, and classifier, and operates under the assumption that traffic signals may appear anywhere (although they note that this can be alleviated with enhanced maps and GPS information). Using matched filtering and shaped detection based on a Hough Transform, they focus on real-time detection of the lights themselves, as opposed to reliable state detection, showing that the light shapes can be matched in both color and gray scale images. In [40] the focus is also on achieving a high detection rate by thresholding based on intensity information to determine if lights are present in the camera image.

Thus, while employing similar concepts to previous work in some respects, we take a more rigorous probabilistic approach to the entire pipeline which allows for improved handling of several sources of uncertainty. In addition, to our knowledge, ours is the only approach that attempts to detect the state of multiple traffic lights within an intersection in real-time (see Figure 6.2). As we will show, this insight significantly improves reliability. Finally, we believe we are the first to present quantitative accuracy results for a fixed algorithm operating in several different lighting conditions, which is clearly crucial for a real system.

Figure 6.2: Simultaneous detection of three distinct traffic lights at one intersection.

In the interest of providing meaningful results that do not depend on a laser-based localization system, for the majority of this chapter we present results that use only our GPS/IMU pose system for localization. Thus, the main algorithms and results of this chapter do not depend on the availability of a laser rangefinder, and only require a camera and a traditional pose system.

The remainder of the chapter is organized as follows. We discuss traffic light mapping in Section 6.2. In Section 6.3 we detail our traffic light state detection pipeline and discuss how we handle sources of uncertainty. We then evaluate an implementation of our system in Section 6.4. In Section 6.5 we extend our results with laser-based localization and camera calibration techniques inspired from the previous chapter, yielding a significant improvement in detection accuracy.

6.2 Traffic Light Mapping

To obtain light locations in global coordinates, we drive a route once and record a sensor log comprising both vehicle pose and camera data. Upon review of this log, we manually select lights relevant to our trajectory from video and track them using the algorithm CamShift [28], which attempts to adjust the bounds of an ellipse such that a hue histogram taken over its interior matches that of the original selection.

For each frame in which the light is tracked, the set $X := \{(u,v),C,R\}$ is recorded,

where (u,v) are the image coordinates of the ellipse's center pixel and C and R are the estimated global camera position and orientation, respectively. To reduce subtractive cancellation during future calculations, we store the vehicle's global coordinates from the first frame in which the light is selected as a local origin C_0. Then, for each X, we find the ray $d = (a,b,c)$ from the camera lens with local position $C - C_0 = (x,y,z)$ to the traffic light lens using the back projection formula:

$$d = \lambda R^{-1} K^{-1} (u,v,1)^T, \tag{6.1}$$

where K is the camera's intrinsic parameter matrix and λ normalizes d to unit length. Once light tracking has finished, the optimal point of intersection from each ray is determined as in [73] from the following. Suppose the light is tracked for n frames, let

$$A = \begin{pmatrix} \sum_i^n (1-a_i^2) & -\sum_i^n a_i b_i & -\sum_i^n a_i c_i \\ -\sum_i^n a_i b_i & \sum_i^n (1-b_i^2) & -\sum_i^n b_i c_i \\ -\sum_i^n a_i c_i & -\sum_i^n b_i c_i & \sum_i^n (1-c_i^2) \end{pmatrix} \tag{6.2}$$

$$b = \begin{pmatrix} \sum_i^n [(1-a_i^2)x_i - a_i b_i y_i - a_i c_i z_i] \\ \sum_i^n [-a_i b_i x_i + (1-b_i^2) y_i - b_i c_i z_i] \\ \sum_i^n [-a_i c_i x_i - b_i c_i x_i + (1-c_i^2) z_i] \end{pmatrix} \tag{6.3}$$

then the estimated global light position is given by $l_{est} = A^{-1} b + C_0$. The resulting traffic light locations are stored in a text file which is read by the traffic light detection system described in Section 6.3. Also during tracking, a bitmap of the ellipse's interior is stored at five meter intervals of vehicle travel vehicle for use in our probabilistic template matching algorithm (see Section 6.3.4).

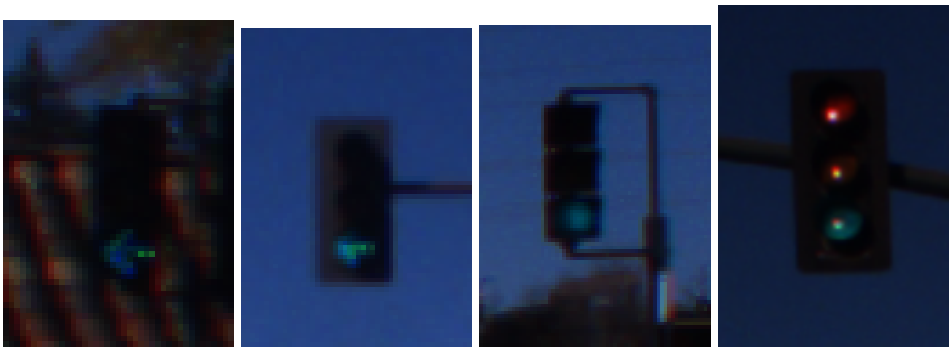


Figure 6.3: Many traffic lights appear almost as dim as their surroundings either by design or by accident.

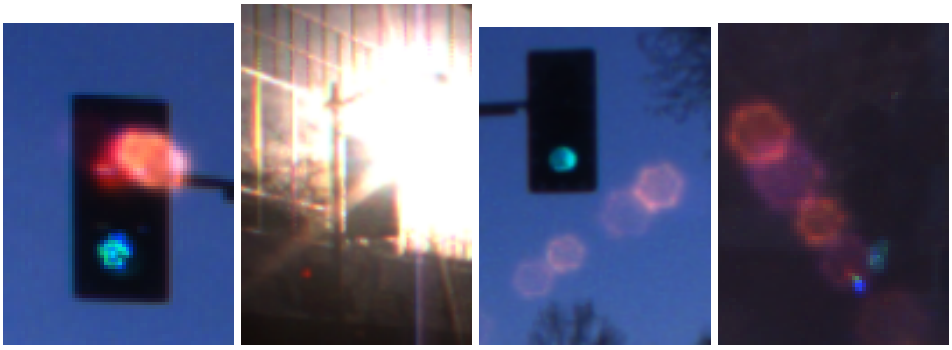


Figure 6.4: Often, a lens flare or back lighting will obscure a light's state. Using cheaper consumer-grade cameras could necessitate additional noise filtering and perhaps explicit flare detection. For example, an actuated camera could rotate itself in order to detect and remove flare.

6.3 Traffic Light State Detection

6.3.1 Prominent Failure Cases

Our system must be robust to pathological situations, like superfluous lights and temporary occlusions, and must also work at all times of day, under all possible lighting conditions.

Assuming that a vision algorithm could distinguish the tail lights from traffic lights during the day in real-time, the task becomes impossible at night with most video cameras. All distinguishing features, aside from light color, are lost. Since in the United States there is no legally-defined standard for light lens material and intensity [7], tail lights near where we expect a traffic light could be considered traffic lights in the image processing algorithm.

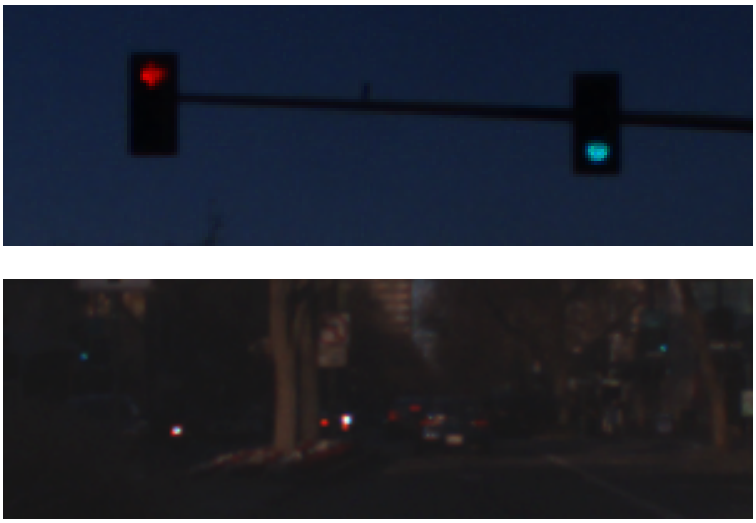


Figure 6.5: (Top) A left-pointing red arrow appears in the camera image as circular, making non-contextual state detection difficult. (Bottom) The vehicle needs to know the state of the next intersection far in advance so appropriate actions are taken. In this case, 200mm green lights appear far apart among a cluttered scene with many light sources.

In the following section, we will describe the probabilistic techniques used to make our system invariant to lighting conditions and reliable in situations that would be challenging for a vision-only approach.

Two major technical problems have to be solved in order to be able to detect the state of a traffic light robustly:

1. Inferring the image region which corresponds to the traffic light.

2. Inferring its state by analyzing the acquired intensity pattern.

Ideally, in solving these problems we would like to choose a reference frame that allows us to take advantage of temporal consistency. The choice of our detection grid as a reference frame assumes several structured error components (as we discuss below), allowing the light's position within the grid to change slowly over time. Given this temporal constraint, and our vision algorithm that performs within the limits of reasonable expectations, we can then apply a histogram filter to infer the image region of the light and determine the color, as discussed in the sections that follow.

6.3.2 Traffic Light Tracking

Consider the diagram in Figure 6.6. Two chains of information lead to the observation of an object in the camera image. Given the distances at which traffic lights have to be detected (70-80m for our vehicle to be able to react appropriately at reasonable traveling speeds), small errors in camera calibration or vehicle localization can lead to large deviations between the expected location of the traffic light on the image and the actual one. From our experience, this error lies in the order of 50 pixels (corresponding to 5-10m in scene space) for state-of-the-art localization hardware and algorithms [61], [52].

Current computer vision algorithms are not yet capable of reliable, holistic scene understanding such that the small image of the traffic light (with the lens less than 3-5 pixels wide) could be distinguished reliably from surrounding patterns.

We propose to make explicit the mismatch between expected perceptions and actual ones (hereafter called perceptual offset) by introducing a random time-varying vector $o^t = (o_1^t, o_2^t)$. Expected perceptions include the position predicted by the mapping, localization, and GPS components of the pipeline while actual perceptions are the data scores returned by our image processing algorithm. We update our belief about o^t sequentially using a Bayes filter, implemented as a histogram filter. One of the critical questions that arises is in which coordinate frame the perceptual offset is represented best. We argue that two straight-forward choices are suboptimal. First, tracking the displacement in the image coordinate frame leads to drastically changing posteriors at every movement of the vehicle. This introduced an unnecessary estimation bias, since the actual displacement effect varies only slowly. Second, tracking the offset in complete three dimensional world space, that is, essentially treating the world pose of the traffic light as a random variable and updating it over time, is hard to accomplish robustly, due to the nature of the construction of a camera image, our main data source. Without a planar constraint, all positions along the ray from the camera sensor through the plane could be weighted equally, since they all project to the same pixel on the image plane.

In contrast to these choices, we found that the most suitable space for modeling the perceptual offset is a bounded plane centered and oriented as the traffic light in world space. Our grid is a natural way to restrict the implied distance from the camera, and is

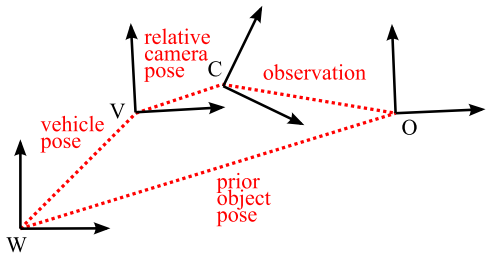


Figure 6.6: Several random variables and their uncertainties influence the accuracy of the localization result. This diagram shows the dependencies of the different coordinate frames: [W]orld, [V]ehicle, [C]amera and [O]bject (here: traffic light).

supported by our assumption of a fairly good light mapping prior with respect to latitude and longitude. We represent our belief about the offset at time t by a histogram over this bounded plane (i.e., a normalized grid) and update it recursively according to Bayes rule [26]

$$P(o^t|z^t, o^{t-1}) = \nu \cdot P(z^t|o^t) \cdot P(o^t|o^{t-1}). \tag{6.4}$$

We assume a relatively peaked Gaussian motion model for $P(o^t|o^{t-1})$ to account for changes to the perceptual offset caused by motion of the vehicle and the vehicle pose uncertainties, and mapping uncertainties (observed as light movement in the detection grid frame due to camera perceptive changes not properly captured) implicit in this motion, where the standard deviation of the Gaussian is proportional to $\varepsilon + k$, where $k \propto$ *vehicle speed*. The observation model $P(z^t|o^t)$, which defines the relationship between camera observations z^t and the perceptual offset o^t of the traffic light is detailed further below.

The prior distribution of o^t on the grid is a two dimensional Gaussian distribution centered at the mapped traffic light location with standard deviations chosen according to the expected error in mapped traffic light location plus the expected perceptual offset. From our experience, these quantities are easy to optimize using a recorded training sequence of traffic light observations.

Our approach works under the following assumptions:

1. The maximal perceptual offset (depending on the localization accuracy of the vehicle and the camera calibration) is smaller than half the side lengths of the grid used to track

it.

2. The resolution of the grid is high enough such that traffic light observations falling onto the border of two grid cells are not represented as two distinct lights with disparate locations (as light location is estimated at the grid cell center).

3. The camera image plane remains approximately parallel to the grid plane during the tracking process. We found that an orientation mismatch of up to about 35 degrees does not pose a problem (and this is already beyond they legally possible orientation mismatch between traffic lights and their corresponding lanes).

4. Neighboring traffic lights are spaced at least half the side length of the grid apart from one another. If this is not the case, the lights should be modeled jointly, which can be achieved in a straightforward way by constructing a joint sensor model with additional color channels, though this is not further addressed here.

When making decisions about the state of the traffic light, the current pose uncertainty of the vehicle — that is, its current posterior distribution over poses — has to be considered. This can be achieved by convolving the current state of the histogram filter for the vehicle pose with the current state of the histogram filter for the perceptual offset, taking the geometric transform between the two grid planes into account.

Given the posterior estimate of the perceptual offset (represented by a histogram) over grid cells, we then select the cell containing the mode as the most likely location of the light. We then report the most likely state at that grid cell, as determined by our data scores.

6.3.3 Uncertainty Discussion

In our system, the quality of input sources varies. Regarding the map input *latitude*, *longitude*, *altitude*, and *orientation* for each light, we assume a greater accuracy in the first two coordinates than in the last. Without a good estimate of *latitude* and *longitude* determining which lane a traffic light applies to would be nearly impossible. This accuracy requirement is also supported by our method for collecting light location data.

Our light mapping procedure depends on accurate GPS and camera calibration. In our system, even when the car is driving directly towards the light, this method typically gives repeatable errors which are easily compensated for with a static offset. The result is a light localized to within approximately two meters across and four meters high.

Another source of uncertainty is the projection of points from global coordinates back to a given pixel in the camera image. To project a three dimensional coordinate in some object space into the two dimensional image plane of a camera, first one usually transforms the coordinate so that its origin is the camera center. To perform this transform beginning with $(latitude, longitude, altitude)$, we first transform our coordinates to the projected Universal Transverse Mercator (UTM) coordinates, then to "smooth coordinates", which are calculated by integrating our GPS-reported velocity over time. This coordinate system is "smooth" because it does not suffer from the abrupt changes of raw GPS information, but does drift over time. We then transform smooth coordinates to the vehicle's local frame, where the origin moves with the vehicle. After this we transform our coordinates to the camera frame. Uncertainty is captured in our random variable C in camera calibration and extrinsic camera parameters, both in terms of static errors and those due to motion or vibration of the vehicle, as well as camera sensor noise.

We used a Flea imager by Point Grey, set to fixed gain, shutter speed and saturation such that the intensity and saturation of the center pixel on an illuminated, 200mm, LED-based green traffic light at 45 meters were reported as 99% and 92%, respectively. The saturation of the sky at noon on a clear day was reported as approximately 66%. Because we are detecting light sources, fixing our camera parameters implies the intensity and saturation of these light sources, as our camera observes them, will remain constant with respect to environmental lighting.

Even with fixed camera parameters, lens type and particular color characteristics of a light are probable culprits for false negatives as discussed in Section 6.3.1 and so are an important set of uncertainty sources. Our image processing algorithm handles these variations by building hue and saturation histograms from the lenses tracked during the light mapping routine. These histograms are then used as lookup tables (henceforth referred to as Histogram As Lookup Tables or HALTs) which condition our images before template matching occurs.

6.3.4 Probabilistic Template Matching

The constraints of the traffic light state detection problem are favorable for using template matching to generate higher-order information from raw camera data. The color of an active lens, since it emits light, remains nearly constant regardless of external lighting conditions. Traffic lights in particular are distinguishable from most other light sources by their strong color saturation. However, variation among lenses and minor effects due to external light direct us to use color probabilistically.

Template Training

Bitmaps of traffic light lenses $\{\mathbf{B}_k\}_{k=1...n}$ are captured during a pre-drive. From these bitmaps, hue and saturation images $\{\mathbf{h}_k\}_{k=1...n}$ and $\{\mathbf{s}_k\}_{k=1...n}$ are extracted. For each state $\omega \in \{red, yellow, green\}$, HALTs H_ω (Figure 6.7) and S_ω, which represent histograms of lens hue and saturation, are generated from (6.5) and (6.6). Lookup Table V, representing a distribution of the expected intensity value of pixels lying on the traffic light's black frame, is defined by (6.7). We denote the height and width of the k^{th} image h_k and w_k.

$$H_\omega[x] = \sum_{k=1}^{n} \sum_{i=1}^{h_k} \sum_{j=1}^{w_k} \delta(\mathbf{h}_{k,i,j}, x), \qquad x = 0...359 \tag{6.5}$$

$$S_\omega[x] = \sum_{k=1}^{n} \sum_{i=1}^{h_k} \sum_{j=1}^{w_k} \delta(\mathbf{s}_{k,i,j}, x), \qquad x = 0...255 \tag{6.6}$$

$$V[x] = 255 \cdot \exp(-0.035x), \qquad x = 0...255 \tag{6.7}$$

$$\delta(a,b) = \begin{cases} 0 & \text{for } a \neq b \\ 1 & \text{otherwise.} \end{cases} \tag{6.8}$$

In California, traffic light lenses have radii of either 200 or 300 mm [7]. A len's resulting radius in pixels can be approximated from the camera's intrinsic parameters magnification m and focal length f, the camera lens to light source lens distance D and the physical radius of the traffic light lens R, as in (6.9).

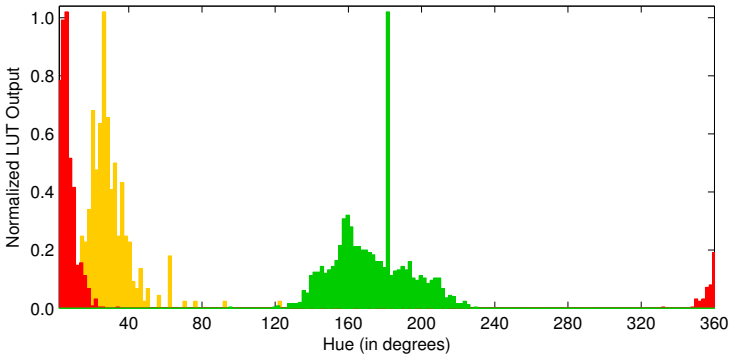


Figure 6.7: Hue histograms $\mathbf{H}_{red,yellow,green}$ are generated from individual lights at various distances, captured during a pre-drive.

$$r = \left\lfloor \frac{2Rmf}{D} \right\rfloor \tag{6.9}$$

The adaptively generated light lens template $\mathbf{T}(r)$ is modeled as a circle of diameter $2r+1$ pixels centered over a black square with side length $l = 4r + (1 - (4r \mod 2))$ after a linear convolution (denoted $\otimes$) with a 3x3 Gaussian kernel $G(\sigma)$ with $\sigma = 0.95$ [3].

$$\hat{\mathbf{T}}_{i,j}(r) = \begin{cases} 0 & \text{if } ||(i-2r+1, j-2r+1)|| > r \\ 255 & \text{otherwise,} \end{cases} \tag{6.10}$$

$$\text{for } i, j = 1 \ldots l$$

$$\mathbf{T}(r) = G(\sigma) \otimes \hat{\mathbf{T}}(r) \tag{6.11}$$

Template Matching

In the image processing pipeline, we limit ourselves to the regions of interest projected onto our image plane from the detection grid (ergo, Figure 6.1). For the remainder of this section, we will consider each such region as an independent image $\mathbf{I}$. Image $\mathbf{I}$ is split into hue, saturation and value images $\mathbf{H}$, $\mathbf{S}$ and $\mathbf{V}$. For an image $\mathbf{N}$, we denote its height $h_{\mathbf{N}}$ and width $w_{\mathbf{N}}$. As a preprocessing step, we heavily weight saturated and high intensity pixels in $\mathbf{S}$ and $\mathbf{V}$ by applying transforms N_S and N_V.

$$N_S(x) = x^5/2^{32},\ x \in \{0...255\} \tag{6.12}$$

$$N_V(x) = \begin{cases} 0 & \text{for } x <= 60 \\ x & \text{otherwise.} \end{cases},\ x \in \{0...255\} \tag{6.13}$$

$$\mathbf{U}_{i,j} = N_S(\mathbf{S}_{i,j}) N_V(\mathbf{V}_{i,j}) \tag{6.14}$$

Image $\mathbf{G}_\omega$ is generated as follows:

$$\mathbf{G}_\omega(\mathbf{H},\mathbf{S},\mathbf{V}) = \sum_{i=1}^{h_\mathbf{I}} \sum_{j=1}^{w_\mathbf{I}} \left\{ \mathbf{U}_{i,j} \sum_{k=1}^{h_{\mathbf{T}(r)}} \sum_{l=1}^{w_{\mathbf{T}(r)}} \left(L(\mathbf{H},\mathbf{S}) + F^2(\mathbf{V}) \right) \right\} \tag{6.15}$$

$$L(\mathbf{H},\mathbf{S}) = \frac{H_\omega[\mathbf{H}_{u(k,l)}] S_\omega[\mathbf{S}_{u(k,l)}] (1 - \delta(\mathbf{T}_{k,l}(r),0))}{\max\{\mathbf{T}(r)\} - \mathbf{T}_{k,l}(r) + 1} \tag{6.16}$$

$$F(\mathbf{V}) = V[\mathbf{V}_{u(k,l)}] \delta(\mathbf{T}_{k,l}(r),0), \tag{6.17}$$

where $u(k,l) = (i+k-h_\mathbf{T}/2, j+l-w_\mathbf{T}/2)$. To eliminate non-circular regions of bright saturation, we then apply a linear convolution with kernel circle template $\mathbf{C}$ over $\mathbf{G}_\omega$:

$$\mathbf{C}_{i,j} = \begin{cases} 0 & \text{if } 2r <= ||(i-2r+1, j-2r+1)|| \\ -1 & \text{if } r <= ||(i-2r+1, j-2r+1)|| < 2r \\ 1 & \text{otherwise,} \end{cases} \tag{6.18}$$

$$\text{for } i,j = 1...l$$

$$\mathbf{Q}_\omega = \mathbf{C} \otimes \mathbf{G}_\omega \tag{6.19}$$

Since the projection of our detection grid $\mathbf{D}$ onto the image plane does not guarantee square regions, we use quad-filling, a convex polygon filling algorithm, as discussed in [38]. For grid cell $\mathbf{D}_{i,j}$, we project the region in $\mathbf{Q}_\omega$ that falls on $\mathbf{D}_{i,j}$'s interior. We then select the maximum value in this region to be this cell's score $\mathbf{E}_{\omega,i,j}$.

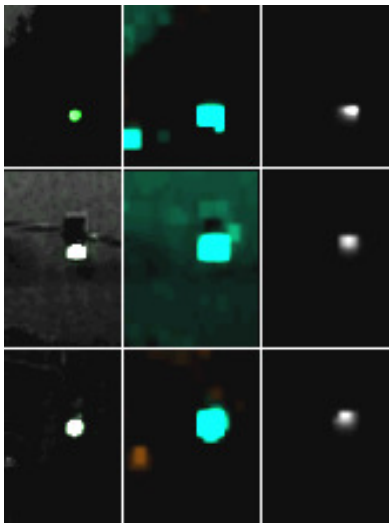


Figure 6.8: (Left) is a visualization constructed by merging $\{G_{red}, G_{green}, G_{yellow}\}$ as the *red*, *green* and *blue* channels of an image. Color saturation corresponds to a higher certainty of a specific state in the current frame (before the prior probability is processed). (Center) depicts the detection grid image P, in which each cell's color represents a grid cell's predicted state and its intensity represents that state's score. (Right) is the resulting prior as described in Section 6.3.2. Each light captured in figure 6.2 is shown from top to bottom.

6.3.5 State Detection Pipeline

For a single light:

1. Supply the module with a pre-defined list of global traffic light locations, specified by $latitude, longitude$ and $altitude$.

2. Supply the module with histograms of hue and saturation for regional traffic lights.

3. Begin looking for lights at a distance determined by the camera resolution and vehicle braking distance.

4. Generate a grid, in global coordinates, centered on the expected location of the traffic light.

5. Project the grid $\mathbf{D}$ onto the camera image plane to determine region of interest image $\mathbf{I}$.

6. Compute $\mathbf{Q}_\omega$ for each $\omega \in \{red, yellow, green\}$.

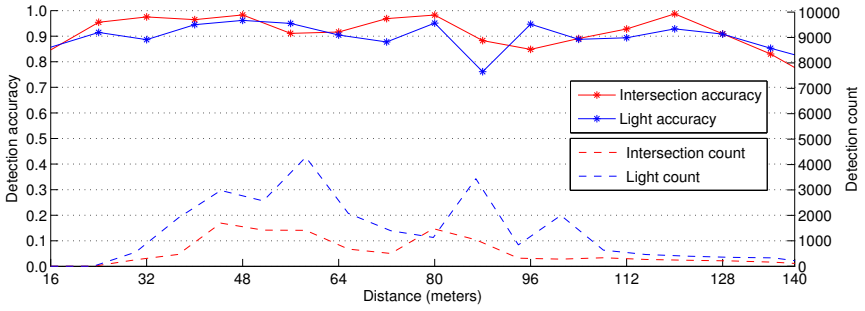


Figure 6.9: Correct detection rates of individual lights and complete intersections from noon are depicted.

7. Back-project $\mathbf{Q}_\omega$ onto $\mathbf{D}$ to get score image $\mathbf{E}_\omega$.

8. Set $P(state) := (G(\sigma_s) \otimes \overline{P}(state))\mathbf{E}_\omega^2$

For multiple lights per intersection:

Given a state $\omega \in \{red, yellow, green\}$, an intersection I with n lights, and a traffic light l_k, $k \in \{1...n\}$ with histogram filter outputs $P(l_k = \omega)$ at a particular intersection, the probability for the intersection state is given by

$$P(I = \omega) = \frac{\prod_k P(l_k = \omega)}{\sum_{c \in \{r,y,g\}} \prod_k P(l_k = c)} \tag{6.20}$$

which treats each light as an independent measurement of the true intersection state, and so the final decision is given by $max_j\{P(I = c_j)\} \Rightarrow I = c_i$.

6.4 Experimental Results

We demonstrate the success of our techniques on Junior, Stanford University's autonomous research vehicle. Junior is equipped with an Applanix LV-420 tightly coupled GPS/IMU system that provides inertial updates and global position estimates at 200 Hz. Although we typically run an additional layer of laser-based localization to refine the GPS pose[53], for the sake of more generally applicable results we disable this refinement localization and rely only on the default GPS output in the following results. Traffic lights are seen by a Point Gray "Flea" video camera which provides 1.3 megapixel RGB frames at 15 Hz.

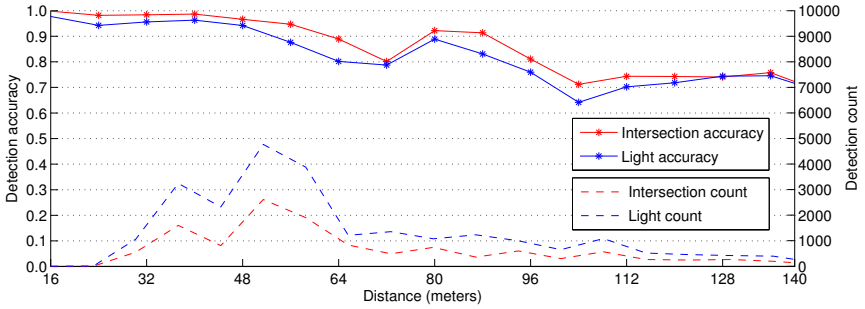


Figure 6.10: Correct detection rates of individual lights and complete intersections from sunset are depicted.

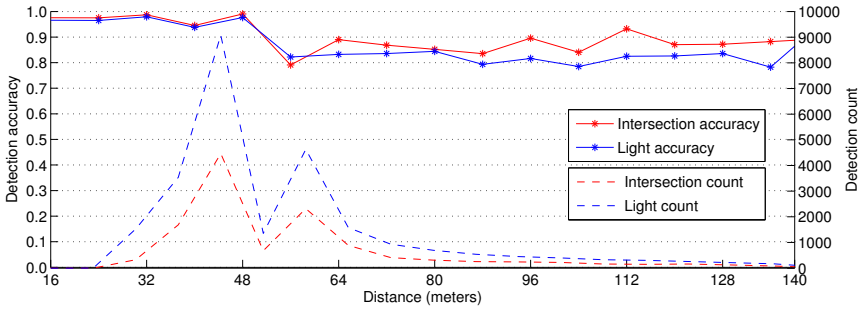


Figure 6.11: Correct detection rates of individual lights and complete intersections from night are depicted.

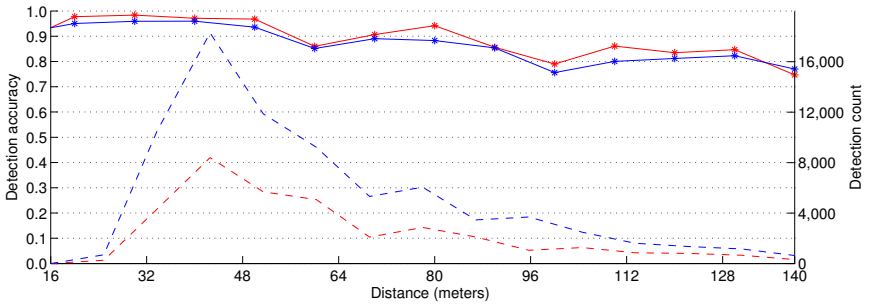


Figure 6.12: Combined results from noon, sunset and night are depicted.

To demonstrate robustness in various traffic light scenarios, and under multiple lighting conditions, we recorded a 20-minute route through Palo Alto, California at each of three different times: noon, sunset, and night. All parameters were fixed across all times of day, indicating that the exact same algorithm is effective across a wide range of lighting conditions. The route comprised 82 lights across 31 intersections, some of which were extremely challenging (e.g. Figure 6.5, bottom).

Figures 6.9-6.11 depict correct detection rates of individual lights and complete intersections under the three lighting conditions, as a function of distance to the lights. Figure 6.12 aggregates all three runs and shows our overall results. Unsurprisingly, detection accuracy drops off somewhat with distance, as the sizes of the traffic lights in the image decrease.

Comparison of individual light detections and intersection decisions suggests a distinct advantage and improved robustness with the latter, when multiple traffic lights are known to display the same color at an intersection. Indeed, using the Bayesian approach as described previously, we are frequently able to correctly identify the true state of the intersection's color even when confronted with conflicting individual detections. As Figure 6.13 shows in two separate examples, the probabilistic approach often yields the desired posterior even in extremely difficult scenes where one or more lights is incorrectly classified.

Mathematically, as long as the sources of error in each simultaneous detection are not fully dependent, a combined approach will be superior in expectation. Indeed, in Figure 6.14 we see that our algorithm's confidence in its detection label is strongly correlated with its accuracy, so that two or more lights which disagree will often be disambiguated correctly based on the detection(s) which are more confident.

A confusion matrix for individual light detections across all three times of day is shown in Figure 6.15. Similarly, a confusion matrix for intersection decisions is shown in Figure 6.16. Again, we see that intersections are meaningfully more accurate than individual lights. These results also indicate that we are very accurate at detecting red lights, and less accurate at detecting green and especially yellow lights. Fortunately for safety purposes, we are much more likely to mistakenly label a light as red than as any other color.

We calculate accuracy based on the fraction of frames containing a correct intersection state classification, out of the total number of video frames in our approximately 20-minute

Figure 6.13: Considering multiple lights with separate search windows (here, shown in blue) improves robustness. In the top frame, the window on the left reports a yellow light with 38% probability and the window on the right reports a red light with 53% probability. The final state decision of the intersection, taking into account probabilities for red, yellow and green states from both windows, yields (correctly) a red state with probability 53%. In the bottom frame, probabilities are 80% and 48% for the left and right lights and 87% for the intersection, whose state is correctly detected as red.

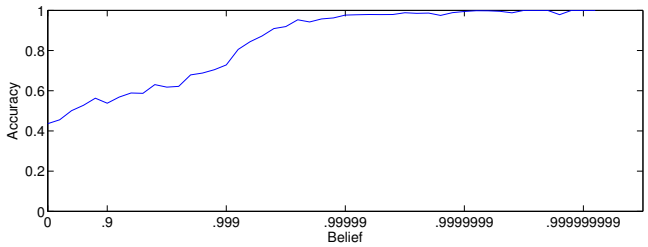


Figure 6.14: Our belief that a given light has a particular state versus the accuracy of that belief, using raw output from histogram filters. Although the filter is generally overconfident, its certainty does correlate strongly with accuracy, as desired.

		Ground Truth			
		Red	Yellow	Green	
Detected	Red	44993	402	4103	90.90%
	Yellow	401	738	936	35.57%
	Green	421	70	24246	98.02%
		98.21%	60.99%	82.79%	91.70%

Figure 6.15: Confusion matrix of light detections. Entries are number of detections. Each row and column has an associated percent accuracy. Yellow lights are more easily confused due to their shared hues with red lights and because it is more difficult to obtain a large training set for relatively rare yellow lights as compared to red and green lights.

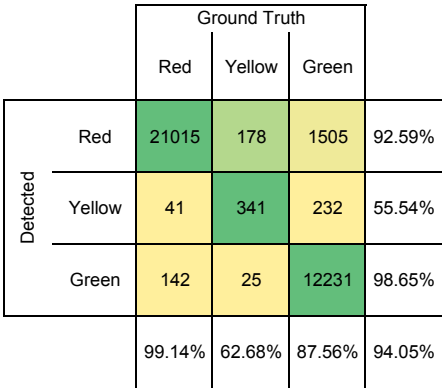


Figure 6.16: Confusion matrix of intersection decisions. Entries are number of decisions. Each row and column has an associated percent accuracy.

test sequence over three sequences for which intersections' lights are visible. Of the 76,310 individual light detections across 82 lights each at three times of day, we achieve 91.7% percent accuracy. For the associated 35,710 intersection decisions across 31 intersections, again at three times of day, we achieve 94.0% accuracy.

	Noon	Sunset	Night	Combined
Lights	92.2%	88.9%	93.8%	91.7%
Intersections	95.0%	92.3%	95.0%	94.0%

A simple extension to the standard framewise approach that is useful for practical applications is to only report traffic light state when several frames of identical color detections have occurred sequentially. Although this extension adds a fraction of a second of latency, the response time still matches or exceeds that of a human, and the results are several (absolute) percentage points higher than the framewise results above. Thus for use in an autonomous vehicle, this application of hysteresis is preferred.

In addition to these quantitative results, a significant implication of this work is that Junior is now able to autonomously drive through intersections governed by traffic lights. However, a safety driver is always present to ensure correct behavior, as we have certainly not solved traffic light detection as well as humans. In addition, if we avoid autonomous driving at intersections whose lights are especially difficult to detect, and utilize our localization system which reduces uncertainty in traffic light location, our actual performance is

significantly better than the preceding results might suggest.

6.5 Localization and Calibration Extension

The preceding algorithms and associated results assume the availability of a positioning system capable of localizing the vehicle to within several meters. However, if a more accurate positioning system is available, we can significantly improve upon the performance of traffic light mapping and state detection using two key ideas.

First, we use the high-accuracy localization data to generate more accurate mapping estimates, employing the same mapping algorithm described in section 6.3. In addition, in the detection and tracking phase, the increased localization accuracy reduces the uncertainty over the vehicle's pose and thus reduces the uncertainty of the mapped traffic light's location in the camera image.

Second, we can leverage the more accurate pose data in order to algorithmically calibrate the camera's extrinsic pose relative to the vehicle. The more accurate this six-DOF transform is, the more accurately the lights' locations can be mapped and predicted.

6.5.1 Incorporating Vehicle Localization

In principle, all of the algorithms described in this chapter are applicable to the scenario where high-accuracy vehicle localization is available; they simply ought to perform better. Indeed, as the vehicle's pose uncertainty decreases, the mapping estimates will automatically become more accurate, everything else being equal.

Consider that the goal of accurate traffic light mapping — that is, the estimation of the traffic lights' locations in global (x, y, z) coordinates — is to improve the subsequent estimation of the traffic lights' locations in image coordinates when performing state detection. If the light locations are inaccurate, then the accuracy of their calculated locations in image coordinates will almost certainly degrade.

In addition, accurately predicting the image coordinates of a previously mapped traffic light in a camera image depends both on the global light mapping error as well as the current localization error relative to the global map. When using the localization algorithm

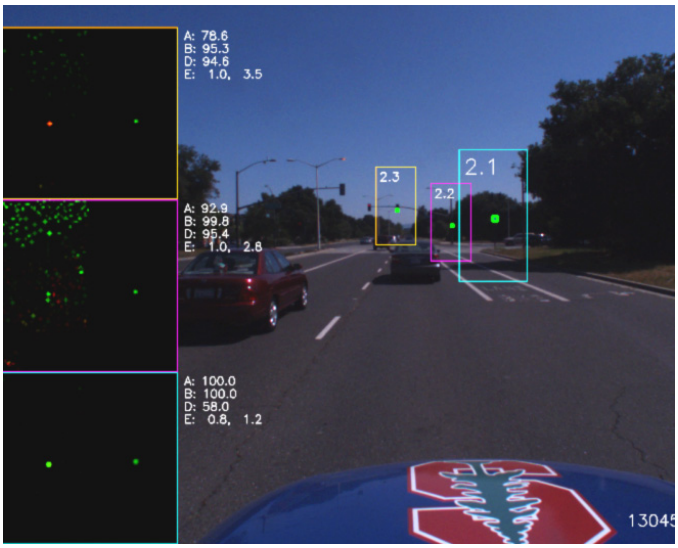


Figure 6.17: Relatively large search windows necessitated by standard GPS without laser localization

Figure 6.18: Search windows are reduced in area by 90% thanks to laser localization

detailed in Chapter 3 for both phases, we find that our uncertainty in the estimated image coordinates of reprojected mapped traffic lights is reduced by 70% in u and v, resulting in a 90% reduction in the required search window area. (See Figures 6.17 and 6.18)

6.5.2 Extrinsic Camera Calibration

Achieving accurate traffic light position estimates in the mapping phase, as well as estimating the location in image coordinates of mapped lights, depends not only on accurate vehicle localization, but also on an accurate intrinsic and extrinsic calibration of the camera itself.

We calibrate the camera's intrinsics using standard methods available in OpenCV [88]. The intrinsic calibration recovers the camera's field of view and distortion parameters.

More difficult is the estimation of the camera's external transform relative to the vehicle frame. Although measuring the camera's position on the car in (x,y,z) is straightforward enough, it is significantly more challenging to measure its three-dimensional angular rotation relative to the vehicle frame. Furthermore, whereas small errors in the translational pose of the camera make only small differences in the forward- and back-projections, small errors in camera orientation have far more significant effects, and this discrepancy increases with traffic light distance.

Consider our traffic light mapping pipeline: for each frame in which we track the light, we record the vehicle's position when the light was observed as well as the (u,v) coordinates of the light in the image. From this collection of pairs, we estimate the (x,y,z) location of the light which minimizes the projection errors. The more accurate the localization and calibration, the more accurate the mapped location, and thus the reprojection into image coordinates for each frame, will be.

It follows, then, that we can search over possible extrinsic calibrations in order to minimize the total reprojection error across all tracked frames; this idea is quite similar to the approach presented in Chapter 5. In this case, the objective function is:

$$J = \sum_{x=1}^{X} \sum_{k=1}^{K_x} (u_{k,obs} - u_{k,est})^2 + (v_{k,obs} - v_{k,est})^2 \tag{6.21}$$

where X is the total number of frames, K_x is the number of lights tracked in frame x,

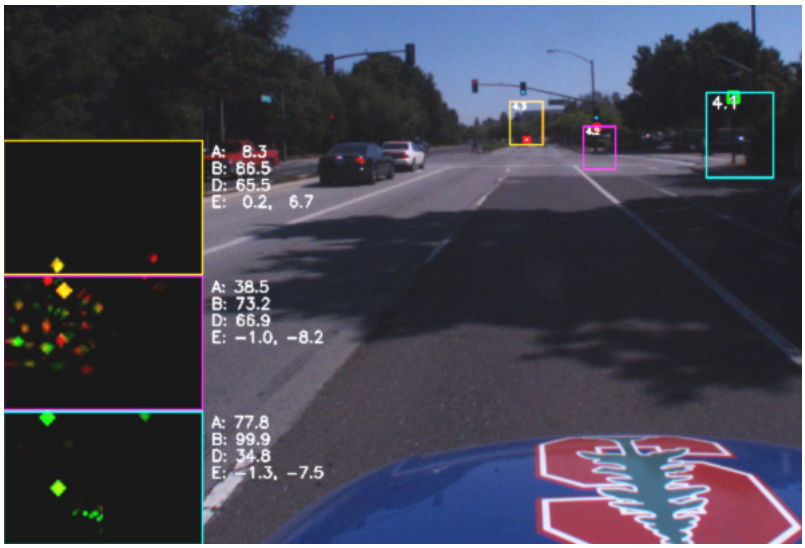


Figure 6.19: Assuming the camera was mounted exactly in line with the vehicle frame, even the best estimated mapping locations of the traffic lights do not match well with their observed locations across all frames.

and u and v of k, obs and k, est are the u and v image coordinates of the kth observed light and its corresponding projected estimate, respectively.

Importantly, u_{est} and v_{est} are always recomputed for each possible extrinsic calibration using the algorithm in Section 6.3. Thus, J penalizes a calibration to the extent that the best possible mapped light locations according to that calibration cause the reprojected image coordinates of the lights to disagree with their true observed coordinates.

J is optimized using iterative grid search. First, the camera's translation is held constant and the orientation is probed; when J is minimized, the resulting orientation is then held constant and the translation is optimized. This procedure is repeated with increasingly smaller step sizes until convergence.

Using this method, combined with the vehicle localization described in the previous section, we recovered a .5° yaw adjustment, a .1° pitch adjustment, and a .05° roll adjustment of the camera relative to the vehicle frame. With the corrected extrinsic calibration transform, the RMS reprojection errors between the new estimated image coordinates for the lights and their observed coordinates decreased from 22.7 to 4.6 pixels, or 80%. (See Figures 6.19 and 6.20)

Utilizing both vehicle localization and the corrected extrinsic calibration transform, the

Figure 6.20: Using the optimized extrinsic camera calibration transform, the new estimated mapping locations of the lights gives estimates in image coordinates which track the lights much more accurately across all frames.

decreased search window area afforded by the improved mapping and estimation accuracy and improved localization accuracy resulted in a reduction of detection errors of 60%, yielding an overall intersection accuracy score of 98%.

6.6 Conclusion

We have described a novel pipeline for traffic light state detection. We take a principled approach to collection of higher-level information from our camera image, utilizing strong constraints in template creation and weighting. Accounting for possible sources of error inherent in the traffic light detection problem, we specifically analyze those errors which contribute to uncertainty in our pipeline. And, for the first time, we have shown that the detection of multiple lights for each intersection improves robustness to noise and significantly improves performance relative to the single-light case.

There remains progress to be made on this challenging topic. Certainly, a higher resolution camera system would improve traffic light clarity at longer distances; currently, at the longer detection ranges, traffic lights often only occupy one or two pixels. In addition, with a higher resolution camera, custom templates for arrow lights could be included in our

pipeline, so that turn lanes could be handled as a special case.

An interesting extension would be to augment the camera vision system with 3D LIDAR data in order to explicitly detect traffic lights by shape in addition to color. On the other side of the cost spectrum, cheaper consumer-grade cameras could be deployed, which would necessitate additional noise filtering and perhaps explicit flare detection (see Figure 6.4). Finally, overall results may be improved if our traffic light detection algorithms were probabilistically combined with computer vision algorithms for holistic scene detection, as false positives may be more easily suppressed.

Chapter 7

Conclusions

In this dissertation, we have presented three probabilistic methods for localizing a vehicle in dynamic urban environments, several unsupervised laser calibration techniques, and a pipeline for mapping and detecting traffic lights.

Underlying all of these algorithms is a principled understanding and accounting for sources of uncertainty, whether they come from the sensor noise, inaccurate maps, changes in weather or lighting, or calibration errors. By modeling probabilistic representations of the world, we are able to repeatedly arrive at robust solutions to seemingly complex problems.

One overarching conclusion that can be drawn from the results of all of the algorithms discussed here is that intelligent energy functions and reasonable incorporation of uncertainty in probabilistic pipelines will frequently yield excellent solutions to optimization or inference problems, even when the exact nature of the uncertainty is unclear or when the objective function is not provably convex.

In particular, the massive amounts of data that are increasingly generated by modern sensors gives us an interesting situation: even a moderately inaccurate representation of uncertainty can still result in an overwhelmingly clear picture of the correct solution to a problem. For example, in the case of mapping and localization in urban terrain with a laser that generates over a million points per second, under the typical approximation of conditional independence between scans given known pose and a map, the difference in probability between the correct location and a location one meter away might well be

something resembling one in the number of elementary particles in the universe.

Consequently, it is not a lack of sufficiently noise-free data that is the impediment to a successful solution, but rather the danger of ignoring dependent sources of error or completely failing to account for a systematic bias that is destroying implicit assumptions in the model. As an example, during the Intelligent Transportation Systems 2008 World Congress autonomous driving demonstration in downtown New York City, the localization system described in Chapter 3 initially failed when we tested it in the rain, after having mapped the environment the previous day in dry weather. Ignoring the systematic darkening of road surfaces caused by the wet pavement and the attenuation of returns caused by falling rain, every localization offset seemed almost impossibly unlikely. Normalizing the average brightness of the incoming data and the map solved this problem, and the demo proceeded without a hitch.

Having massive amounts of data frequently enables new categories of algorithms that would previously have been unimaginable. For example, to our knowledge, unsupervised calibration of laser on a moving robotic platform had never been attempted before our work. Yet with the observation that multiple beams will see the same surfaces, and that incorrect extrinsic or intrinsic calibrations will blur the accumulated point clouds as the robot moves, an optimization solution naturally emerged. Now, we are able to place multiple lasers in arbitrary locations on a vehicle, collect only seconds of data, and compute with incredible precision the exact translation and orientation of the sensors relative to the vehicle's inertial frame.

Despite the new autonomous abilities our methods enable, we recognize that a sizeable gap continues to exist between our research results and a commercial system. The sensors we use are still out of the price range of almost all consumers. Our localization algorithms, while extremely robust to typical conditions, fail to perform acceptably in the face of catastrophic environment changes or snowy weather. Our calibration algorithms, while powerful and accurate, run offline and thus are not currently equipped to handle suddenly displaced sensors; an online variant that could quickly detect calibration changes is a worthy goal. Our traffic light detector, while approaching 100% accuracy, falls slightly short, and thus is not yet suitable for a production system.

Our contributions to localization, automatic sensor calibration, and traffic light detection are just part of the journey towards bringing autonomous vehicles from our imagination to our roads. Together with the rest of the Stanford Driving Team, we have made important steps towards bringing this technology to life, and we are optimistic that continued research and engineering efforts in both academia and industry will soon complete the vision.

Appendix A

Hardware and software architecture

In this appendix, we provide a brief overview of our vehicle's hardware and software architecture. For reference, we also summarize three crucial software modules used in our team's autonomous vehicle that are outside the scope of this dissertation: object detection and classification, trajectory planning, and vehicle control.

A.1 Vehicle hardware

Our research vehicle, "Junior," is a 2006 Volkswagen Passat wagon, equipped with a 4-cylinder turbo diesel injection engine. It features an electronically actuated throttle, gear shifter, parking brake, and steering system. An interface box designed in collaboration with Volkswagen provides software control over these functions as well as brake pressure and turn signals, in addition to the ability to fall back to human control of the vehicle during a software or power failure, or by manual takeover. The engine provides electric power to Junior's computing system through a high-current prototype alternator, supported by a battery-backed electronically controlled power system.

We have deployed the following sensors on our vehicle:

- A Velodyne HDL-64E S2 laser rangefinder with 64 beams, a 360-degree field of view, a horizontal angular resolution of 0.2 degrees, and a 10Hz spin rate. See Fig. A.1(a).
- Two SICK LMS 291-S14 single-beam laser rangefinders with a 90-degree field of view,

Figure A.1: Junior senses the environment with laser rangefinders (a, b, c, h), cameras (d, e), radars (f), and a positioning system with an inertial measurement unit and GPS receivers (h).

an angular resolution of 0.5 degrees, and a 75Hz spin rate. See Fig. A.1(b).

- A RIEGL LMS-Q120 single-beam laser rangefinder with a 80-degree field of view, an angular resolution of .08 degrees, and a 10Hz spin rate. See Fig. A.1(c).

- A Point Grey Flea RGB camera with 1280x960 resolution and a 15Hz frame rate. See Fig. A.1(d). item A Point Grey Ladybug-3 panoramic RGB camera with six 1600x1200 cameras and a 15Hz frame rate. See Fig. A.1(e).

- Five Bosch LRR3 radar sensors with a range of 250m and a 30-degree opening angle. See Fig. A.1(f).

- An Applanix POS-LV 420 GPS/IMU system with a GPS Azimuth Heading measurement subsystem, high-performance intertial measurement unit, wheel odemetry via a distance measurement unit (DMI), and Omnistar satellite-based Virtual Base Station service, proving pose and inertial updates at 200Hz. See Fig. A.1(g).

- Two SICK LD-LRS single-beam laser rangefinders with a 270-degree field of view, an angular resolution of 0.1 degrees, and a 10Hz spin rate. See Fig. A.1(h).

The Applanix is used by all modules which require pose data. The Velodyne is used for static and dynamic obstacle detection, object classification, and the localization system described in Chapter 3. The SICK and RIEGL LMS lasers were used for the localization systems described in Chapters 2 and 4. The Flea camera is used for the traffic light detection system described in Chapter 6. The Ladybug camera is used for camera-based obstacle detection, which is not described here. The Bosch radars are used for long-range moving obstacle detection.

Two on-board Xeon computers running Linux provide the computational power to run our software. A 12-core server runs our vision and laser algorithms while a 6-core server takes care of planning, control, and low-level communication.

A.2 Software architecture

The low-level software system is a modified version of that used in the 2005 DARPA Grand Challenge and the 2007 DARPA Urban Challenge [61]. Individual input, output, and processing modules run asynchronously, communicating over network sockets and shared memory. Modules interact via an anonymous publish/subscribe message passing protocol, based on the Inter Process Communication Toolkit (IPC) [75].

The localization module subscribes to GPS and IMU measurements as well as laser readings, and publishes Junior's global position offset. The perception module subscribes to position and localization messages, laser readings, and radar readings, and publishes a list of static and dynamic obstacles. The classification module classifies all dynamic obstacles found by the perception module. The traffic light module subscribes to localization messages and camera data, and publishes the state of the intersection when appropriate.

The planning module subscribes to the localization, perception, and classification messages and, based on an RNDF and a goal location, publishes the desired vehicle trajectory. The control module subscribes to the planner's trajectory messages and based on the current pose and desired trajectory, outputs throttle, brake, and steering torque commands which are sent to the vehicle's drive-by-wire system for actuation.

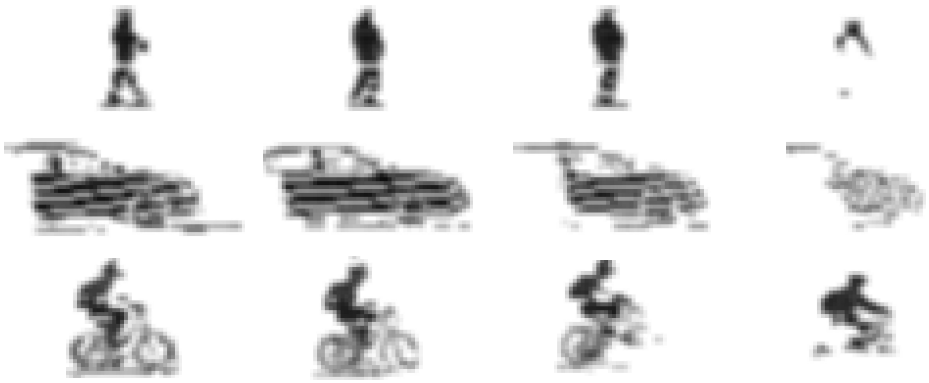


Figure A.2: Virtual orthographic camera intensity images from three tracks of objects that Junior can recognize. Depth segmentation of objects provides invariance to background clutter. Classification of depth data allows one to put objects in a canonical orientation, thus providing some measure of perspective invariance.

A.3 Object classification

In some scenarios, an autonomous vehicle requires deeper understanding of the environment to behave correctly; for example, at intersections, knowing the location of pedestrians and bicyclists can allow the car to make more sophisticated precedence decisions.

Junior's object recognition system now includes recognition of pedestrians, bicyclists, and cars. The following is a brief overview of the key points of this system; see [77] for details.

First, every Velodyne scan is segmented using depth information. This essentially amounts to local ground plane removal and connected components clustering of the remaining points, done in a 2D grid for efficiency. The segments are then fed into a standard Kalman tracker which includes (x, y) position and $(\dot{x}, \dot{y})$ velocity in its state variable. Classification of *tracks* of objects was found to be essential to high performance.

Track classification is done by applying two separate boosting classifiers: one of the shape of the object at each frame in the track, and one on motion descriptors of the entire track. These predictions are combined using a discrete Bayes filter. Example frame descriptors are shown in Figure A.2.

When tracking and segmentation are correct, the accuracy of the classifier is about 98% overall. The system is able to run in real time due to caching of shared computation in the descriptor pipeline and because classification of only segmented objects dramatically

reduces the number of possibilities that must be considered compared to sliding window type approaches. By only considering objects near the road and those that were previously recognized as pedestrians, bicyclists, and cars, we are frequently able to classify everything in every scan while maintaining real time operation.

A.4 Planning

As autonomous vehicles advance toward handling realistic road traffic, they face street scenarios where the dynamics of other traffic participants must be considered explicitly. These situations include everyday driving maneuvers like merging into traffic flow, passing with on- coming traffic, changing lanes, or avoiding other vehicles. This is where trajectory concepts come into play, which explicitly account for the time on the planning and execution level. The presented method embarks on this strategy and transfers velocity and distance control to the planning level. Additionally, the algorithm provides for reactive obstacle avoidance by the combined usage of steering and breaking/acceleration [89].

We formulate the problem of trajectory tracking in an optimal sense to take advantage of optimal control theory asserting consistency in the choice of the best feasible trajectory over time. For the car, this means that it follows the remainder of the previously calculated trajectory in each planning step. Bellman's Principle therefore asserts convergence.

While our main criterion in choosing a cost functional is compliance with Bellman's principle of optimality, trajectories minimizing it must still be close to the desired traffic behavior of the autonomous car. At the same time, the best compromise has to be found in the longitudinal direction in an analog manner. Assuming the car drives too fast or too close to the vehicle in front, it has to slow down noticeably but without excessive rush. Ease and comfort can be best described by the derivative of lateral or longitudinal acceleration: $\text{jerk}(t) := \dot{a}(t)$.

A well known approach in tracking control theory is the moving frame method [59]. Here, we use the Frenet-Serret formulation and apply the moving frame method for combining different lateral and longitudinal cost functionals for different tasks as well as to mimic human-like driving behavior. The moving reference frame is given by the tangential and normal vectors at a certain point of some curve referred to as the *center line* in the

following. This center line represents either the ideal path along the free road, or the result of a path planning algorithm for unstructured environments [91]. Rather than formulating the trajectory generation problem directly in Cartesian Coordinates, we switch to the above mentioned dynamic reference frame and seek to generate a one-dimensional trajectory for both the root point along the center line and the perpendicular offset.

A.4.1 Lateral Motion

Since we seek to maximize comfort and therefore minimize the squared jerk along the resulting trajectory, we choose the start state of our optimization according to the previously calculated trajectory. The cost functional

$$C_d = k_j \int_{t_0}^{t_1} \dddot{d}(\tau)\mathrm{d}\tau + k_t[t_1 - t_0] + k_d d_1^2 \tag{A.1}$$

with d_1 as the lateral deviation at the end state of the current planning horizon, and the weighting factors $k_j, k_t, k_d > 0$, is independent of the longitudinal movement and therefore velocity invariant. Quintic polynomials can be found to satisfy this cost functional. Instead of calculating the best trajectory explicitly and modifying the coefficients to get a valid alternative, we generate in a first step, such as in [86], a trajectory set by combining different end conditions. In a second step we can pick the valid trajectory with the lowest cost. Notice that, as we continue in each step along the optimal trajectory, the remaining trajectory will be, the optimal solution in the next step. At extreme low speeds, this strategy above disregards the non-holonomic property of the car, so that the majority of the trajectories would be rejected due to invalid curvatures. For this reason the behavioral layer can switch below a certain velocity threshold to a slightly different trajectory mode generating the lateral trajectory in dependence on the longitudinal movement; see [89] for details.

A.4.2 Longitudinal Movement

In contrast to previous works where time or travelled distance was the key criterion, we will focus here on comfort and contribute at the same time to safety at high speeds, as smooth movements adapt much better to the traffic flow. For that reason, we also take

the longitudinal jerk into account in our optimization problem. Since distance keeping, merging, and stopping at certain positions require trajectories, which describe the transfer from the current position to a longitudinal, possibly moving, target position, we generate a longitudinal trajectory set analogously to the lateral trajectories with the following cost functional:

$$C_t = k_j \int_{t_0}^{t_1} \dddot{s}(\tau)\mathrm{d}\tau + k_t[t_1 - t_0] + k_s[s_1 - s_d]^2, \tag{A.2}$$

with the distance to the leading vehicle along the center line s_d. Again, a quintic polynomials satisfies the cost functional. The movement of the leading vehicle has to be predicted within the considered time horizon. Similarly, we can define a target point which enables us to position the autonomous car next to a pair of vehicles before squeezing slowly in between during a tight merging maneuver. For stopping at intersections due to a red light or a stop sign, the target distance becomes the distance to the stop line and the target's velocity and acceleration are set to 0.

In situations without a vehicle or a stop line directly ahead, the autonomous car does not necessarily have to be at a certain position but needs to adapt to a desired velocity given by the behavioral level. For this case, the cost functional

$$C_v = k_j \int_{t_0}^{t_1} \dddot{s}(\tau)\mathrm{d}\tau + k_t[t_1 - t_0] + k_{\dot{s}}[\dot{s}_1 - \dot{s}_d]^2, \tag{A.3}$$

is satisfied by a quartic polynomial. Before combining the lateral and longitudinal trajectory sets, each one is checked against outsized curvatures and acceleration values. The remainders in each set are then brought together in every combination.

A.4.3 Combining Lateral and Longitudinal Curves

In a last step, the conjoint costs of each trajectory is calculated as the weighted sum:

$$C_{\mathrm{tot}} = k_{\mathrm{lat}} C_{\mathrm{lat}} + k_{\mathrm{lon}} C_{\mathrm{lon}} \tag{A.4}$$

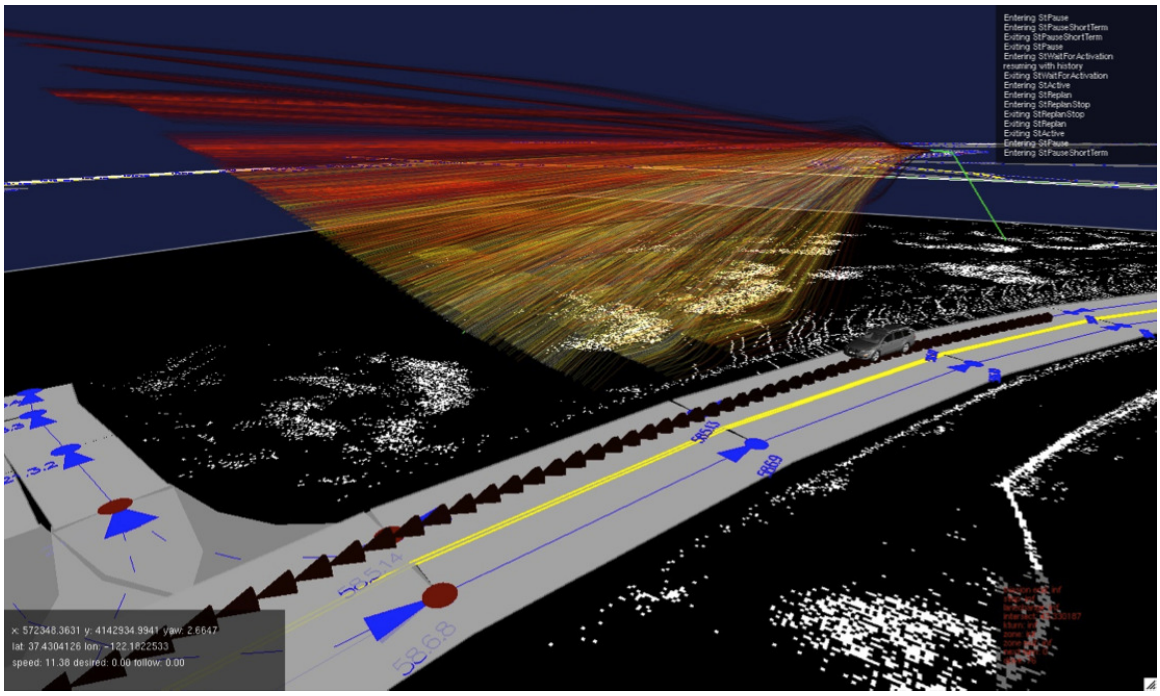


Figure A.3: Smooth trajectory set: The z axis shows the velocity; overall costs are indicated by the color.

As far as our experience goes, it is sufficient for high- way trajectory generation to classify all traffic scenarios as merging, following another car, keeping a certain velocity, stopping at a certain point, and all combinations thereof, which are conflicting most of the time. In control theory, override control is a well-known technique, which chooses among multiple control strategies according to a scheme, prevalently the most conservative one via a max or a min operator. An example for a generated smooth trajectory set is shown in Fig. A.3.

A.5 Control

The goal of Junior's control system is to take the upcoming trajectory output by the planner and generate system inputs (throttle/braking and steering torque) in order to follow this trajectory. To achieve this end, we employ a mixture of a model predictive control (MPC) strategy, based upon well-known physically based vehicle models [30], along with feedforward proportional integral derivative (PID) control for the lower-level feedback control tasks such as applying torque to achieve a desired wheel angle [42].

At the higher of these two levels, the car's state and control input are described by the vectors $\mathbf{x} \in \mathbb{R}^7$ and $\mathbf{u} \in \mathbb{R}^2$

$$\mathbf{x} = [x, y, \theta, u, v, \dot{\theta}, \delta], \quad \mathbf{u} = [\dot{u}, \dot{\delta}] \tag{A.5}$$

where x, y, and θ denote the 2D state and orientation, u and v denote the longitudinal and lateral velocities (aligned with the car frame), δ denotes the wheel angle, and the dotted variables represent the corresponding time derivatives. The equations of motion are given by a bicycle model[1]

$$\begin{aligned} \dot{v} &= \tan\delta(\dot{u} - \dot{\theta}v) + (F_{yf}/\cos\delta + F_{yr})/m - \dot{\theta}u \\ I\ddot{\theta} &= ma\tan\delta(\dot{u} - \dot{\theta}v) + aF_{yf}/\cos\delta - bF_{yr} \end{aligned} \tag{A.6}$$

where m denotes the car's mass a and b denote the distance from the center of gravity to the front and rear axles respectively, I is the moment of inertia, and the lateral tire forces are given via a linear tire model with stiffness C,

$$F_{yf} = C\left(\tan^{-1}\left(\frac{v + \dot{\theta}a}{u}\right) - \delta\right), F_{yf} = C\tan^{-1}\left(\frac{v - \dot{\theta}b}{u}\right). \tag{A.7}$$

Given this system, we interpret a trajectory output by the planner as a sequence of desired states $\mathbf{x}^\star_{1:H}$ for some time horizon H, and minimize the quadratic cost function

$$J(\mathbf{u}_{1:H}) = \sum_{t=1}^{H} \left((\mathbf{x}_t - \mathbf{x}^\star_t)^T Q(\mathbf{x}_t - \mathbf{x}^\star_t) + \mathbf{u}_t^T R\mathbf{u}_t\right) \tag{A.8}$$

subject to the system dynamics, where Q and R are cost matrices (chosen to be diagonal, in our case), that specify the trade-off in errors between the different state components. We minimize this objective by linearizing the dynamics around the target trajectory, and applying an algorithm known as the Linear Quadratic Regulator (LQR) [1] to the resulting linearized system; this optimization is done in an online fashion, each time we send a new control command.

[1] Although not usually written in this form, this is simply the model from e.g. [29] for a front wheel drive car and with the longitudinal forces set to whatever value necessary to achieve $\dot{u}$ acceleration.

Finally, after obtaining controls that approximately minimize this cost function, we integrate the dynamics forward, leading to a sequence of desired steering angles and velocities $\delta^d_{1:H}, u^d_{1:H}$ (and similarly for their velocities). To achieve these states we use feedforward PID control, for instance applying steering torque

$$\tau = k_p(\delta_t - \delta^d_t) + k_d(\dot{\delta} - \dot{\delta}^d) + k_{ff} f(\delta^d_t) \tag{A.9}$$

where k_p, k_d and k_{ff} are the feedback gains, and $f(\delta)$ is a feedforward control tuned for the car.

Bibliography

[1] B. Anderson and J. Moore. *Optimal Control: Linear Quadratic Methods.* Prentice-Hall, 1989.

[2] C. Baker, A. Morris, D. Ferguson, S. Thayer, C. Whittaker, Z. Omohundro, C. Reverte, W. Whittaker, D. Hähnel, and S. Thrun. A campaign in autonomous mine mapping. *International Conference on Robotics and Automation (ICRA)*, 2004.

[3] G. Bradski and A. Kaehler. *Learning OpenCV*, 1st ed. O'Reilly Media, Inc., 2008, p. 112.

[4] M. Bosse, P. Newman, J. Leonard, M. Soika, W. Feiten, and S. Teller. Simultaneous localization and map building in large-scale cyclic environments using the atlas framework. *IJRR*, 23(12), 2004.

[5] Bureau of Transportation Statistics U.S. Department of Transportation. Annual report, 2005.

[6] W. Burgard and R. Triebel, H. Andreasson. Improving Plane Extraction from 3D Data by Fusing Laser Data and Vision. *Intelligent Robotics and Systems*, 2005.

[7] C. D. of Transportation. *California Manual on Uniform Traffic Control Devices*, September 2006.

[8] A. Censi, L. Marchionni, and G.Oriolo. Simultaneous maximum-likelihood calibration of odometry and sensor parameters. *ICRA*, 2008.

[9] P. Cheeseman and P. Smith. On the representation and estimation of spatial uncertainty *IJR*, 5, 1986.

[10] G. Chao and J. Spletzer. On-Line Calibration of Multiple LIDARs on a Mobile Vehicle Platform. *ICRA*, 2010.

[11] Y. Chen and G. Medioni. Object Modeling by Registration of Multiple Range Images. *Proc. of the 1992 IEEE Intl. Conf. on Robotics and Automation*, pp. 2724-2729, 1991.

[12] DARPA. DARPA Grand Challenge rulebook, 2004. On the Web at http://www.darpa.mil/grandchallenge05/Rules_8oct04.pdf.

[13] DARPA. DARPA Urban Challenge rulebook, 2006. On the Web at http://www.darpa.mil/grandchallenge/docs/Urban_Challenge_Rules_ 121106.pdf.

[14] P. Debevec et al. Estimating Surface Reflectance Properties of a Complex Scene under Captured Natural Illumination. Technical Report ICTTR-06.2004, *University of Southern California Institute for Creative Technologies Graphics Laboratory*, 2004.

[15] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte Carlo localization for mobile robots. *ICRA*, 1999.

[16] F. Dellaert, W. Burgard, D. Fox, and S. Thrun. Using the condensation algorithm for robust, vision-based mobile robot localization. *ICVP*, 1999.

[17] A. Dempster, P. Laird, D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society.* Series B 39(1): 1-38, 1977.

[18] E.D. Dickmanns. Vision for ground vehicles: history and prospects. *IJVAS*, 1(1) 2002.

[19] A Doucet. On sequential simulation-based methods for Bayesian filtering. *TR CUED/F-INFENG/TR 310, Cambridge University*, 1998.

[20] B. Douillard, A. Brooks and F. Ramos. A 3D Laser and Vision Based Classifier. *International Conference in Intelligent Sensors, Sensor Networks and Information Professing (ISSNIP)*, 2009.

[21] T. Duckett, S. Marsland, and J. Shapiro. Learning globally consistent maps by relaxation. *ICRA*, 2000.

[22] H.F. Durrant-Whyte. Uncertain geometry in robotics. *IEEE TRA*, 4(1), 1988.

[23] A. Eliazar and R. Parr. DP-SLAM: Fast, robust simultaneous localization and mapping without predetermined landmarks. *IJCAI*, 2003.

[24] U. Franke, D. Gavrila, S. Görzig, F. Lindner, F. Paetzold, and C. Wöhler. Autonomous driving goes downtown. *IEEE Intelligent Systems*, vol. 13, no. 6, pp. 40–48, 1998.

[25] J. Folkesson and H. I. Christensen. Robust SLAM. *ISAV* 2004.

[26] D. Fox, J. Hightower, L. Liao, D. Schulz, and G. Borriello. Bayesian filtering for location estimation. *IEEE Pervasive Computing*, vol. 2, pp. 24–33, 2003.

[27] U. Frese, P. Larsson, and T. Duckett. A multigrid algorithm for simultaneous localization and mapping. *IEEE Transactions on Robotics*, 2005.

[28] K. Fukunaga and L. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *Information Theory, IEEE Transactions on*, vol. 21, no. 1, pp. 32–40, January 2003.

[29] J. Gerdes and E. Rosseter. A Unified Approach to Driver Assistance Systems Based on Artificial Potential Fields *Proceedings of the ASME International Mechanical Engineering Congress and Exposition*, 1999.

[30] T.D. Gillespie. *Fundamentals of Vehicle Dynamics*. SAE Publications, 1992.

[31] V. Gradinescu, C. Gorgorin, R. Diaconescu, V. Cristea, and L. Iftod. Adaptive traffic lights using car-to-car communication.*VTC Spring*. IEEE, 2007, pp. 21–25. [Online]. Available: http://dblp.uni-trier.de/db/conf/vtc/vtc2007s.html

[32] J. Guivant, E. Nebot, and S. Baiker. Autonomous navigation and map building using laser range sensors in outdoor applications. *JRS*, 17(10), 2000.

[33] J.-S. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. *CIRA*, 2000.

[34] D. Hähnel, D. Schulz, and W. Burgard. Map building with mobile robots in populated environments. *IROS*, 2002.

[35] D. Hähnel, D. Fox, W. Burgard, and S. Thrun. A highly efficient FastSLAM algorithm for generating cyclic maps of large-scale environments from raw laser range measurements. *IROS*, 2003.

[36] B. R. Harvey and D. D. Lichti. The effects of reflecting surface material properties on time-of-flight laser scanning measurements. *International Society for Photogrammetry and Remote Sensing*, 2002.

[37] M. Hebert, C. Thorpe, and A. Stentz. *Intelligent Unmanned Ground Vehicles*. Kluwer, 1997.

[38] J. F. S. Hill and S. M. Kelley, *Computer Graphics Using OpenGL (3rd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006.

[39] T.-H. Hwang, I.-H. Joo, and S. I. Cho. Detection of traffic lights for vision-based car navigation system. *PSIVT*, 2006, pp. 682–691.

[40] S.-K. Joo, Y. Kim, S. I. Cho, K. Choi, and K. Lee. Traffic light detection using rotated principal component analysis for video-based car navigation system. *IEICE Transactions*, vol. 91-D, no. 12, pp. 2884–2887, 2008.

[41] A. Kaboli, M. Bowling, and P. Musilek. Bayesian calibration for Monte Carlo localization. *AAAI*, 2006.

[42] J. Kolter, C. Plagemann, D. Jackson, A. Ng and S. Thrun. A probabilistic approach to mixed open-loop and closed-loop control, with application to extreme autonomous driving. *International Conference on Robotics and Automation*, 2010.

[43] K. Konolige. Large-scale map-making. *AAAI*, 2004.

[44] K. Konolige and M. Agrawal. FrameSLAM: From bundle adjustment to realtime visual mapping. *IEEE Transactions on Robotics*, 24(5):1066-1077, 2008.

[45] H. Kretzschmar, G. Grisetti and C. Stachniss. Lifelong map learning for graph-based SLAM in static environments. *KI - Kunstliche Intelligenz*, 2010.

[46] H. Kretzschmar, C. Stachniss and G. Grisetti. Efficient Information-Theoretic Graph Pruning for Graph-Based SLAM with Laser Range Finders. *IEEE International Conference on Intelligent Robots and Systems*, 2011.

[47] B. Kuipers and Y.-T. Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *JRAS*, 8, 1991.

[48] B. Lamond and G. Watson. Hybrid rendering - a new integration of photogrammetry and laser scanning for image based rendering. *Proc. of Theory and Practice of Computer Graphics(TPCG)*, 2004.

[49] K. Lee, B. Kalyan, S. Wijesoma, M. Adams, F. Hover, and N. Patrikalakis. Tracking random finite objects using 3D-LIDAR in marine environments. *Proceedings of the 2010 ACM Symposium on Applied Computing.*

[50] S. Lenser and M. Veloso. Sensor resetting localization for poorly modelled mobile robots. *ICRA*, 2000.

[51] J. Leonard, J.D. Tardós, S. Thrun, and H. Choset, editors. *ICRA Workshop W4*, 2002.

[52] J. Levinson, M. Montemerlo, and S. Thrun. Map-based precision vehicle localization in urban environments. *Proceedings of Robotics: Science and Systems*, Atlanta, GA, USA, June 2007.

[53] J. Levinson and S. Thrun. Robust vehicle localization in urban environments using probabilistic maps. *2010 IEEE International Conference on Robotics and Automation*. IEEE, May 2010, pp. 4372–4378.

[54] J. Levinson and S. Thrun. Unsupervised Calibration for Multi-Beam Lasers. *International Symposium on Experimental Robotics*, 2010.

[55] J. Levinson, J. Askeland, J. Dolson and S. Thrun. Traffic Light Localization and State Detection. *International Conference on Robotics and Automation*, 2011.

[56] J. Levinson, J. Askeland, J. Becker, J Dolson, D. Held, S. Kammel, J. Kolter, D. Lnger, O. Pink, V. Pratt, M. Sokolsky, G. Stanek, D. Stavens, A. Teichman, M. Werling and S. Thrun. Towards Fully Autonomous Driving: Systems and Algorithms. *IEEE Intelligent Vehicles*, 2011.

[57] F. Lindner, U. Kressel, and S. Kaelberer. Robust recognition of traffic signals. *Intelligent Vehicles Symposium, 2004 IEEE*, June 2004, pp. 49–53.

[58] J. Liu and R. Chen. Sequential monte carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93. 1998.

[59] P. Martin, P. Rouchon and J. Rudolph. Invariant tracking. *ESAIM: Control, Optimisation and Calculus of Variations*, 2004, 1(10) pp. 1-13.

[60] M. Montemerlo, S. Thrun, H. Dahlkamp and S. Strohband. Winning the DARPA Grand Challenge with an AI robot. *Proc. of the Association for the Advancement of Artificial Intelligence (AAAI)*, 2006.

[61] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, A. Petrovskaya, M. Pflueger, G. Stanek, D. Stavens, A. Vogt, and S. Thrun. Junior: The stanford entry in the urban challenge. *Journal of Field Robotics*, 2008.

[62] D. Mount and S. Arya. ANN: A Library for Approximate Nearest Neighbor Searching. available on-line at http://www.cs.umd.edu/ mount/ANN/ updated January, 2010.

[63] N. Muhammad and S. Lacroix. Calibration of a rotating multi-beam Lidar. published on-line at http://www.pges.fr/2rt3d/SortedDocs/Publications/ CalibrationOfA-MultiBeamLidar2.pdf 2009.

[64] J.-H. Park and C. sung Jeong. Real-time signal light detection. *FGCNS '08: Proceedings of the 2008 Second International Conference on Future Generation Communication and Networking Symposia*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 139–142.

[65] M.A. Paskin. Thin junction tree filters for simultaneous localization and mapping. *IJCAI*, 2003.

[66] J.L. Rodgers and W.A. Nicewander. Thirteen ways to look at the correlation coefficient. *The American Statistician*, 42(1):59-66, February 1988.

[67] A. Petrovskaya and S. Thrun. Model based vehicle detection and tracking for autonomous urban driving. *Autonomous Robots*, Volume 26 Issue 2-3. April 2009.

[68] M. Pitt and N. Shephard. Filtering via simulation: auxiliary particle filter. *Journal of the American Statistical Association*, 94, 1999.

[69] D. Pomerleau. *Neural Network Perception for Mobile Robot Guidance*. Kluwer, 1993.

[70] A. Segal, D. Haehnel, and S. Thrun Generalized-ICP. *Robotics Science and Systems*, 2009.

[71] J. Shackleton, B. VanVoorst, J. Hesch. Tracking People with a 360-degree Lidar. *7th IEEE Conference on Advanced Video and Signal Based Surveillance*, 2010.

[72] T. Shioyama, H. Wu, N. Nakamura, and S. Kitawaki. Measurement of the length of pedestrian crossings and detection of traffic lights from image data. *Measurement Science and Technology*, vol. 13, no. 9, pp. 1450–1457, 2002. [Online]. Available: http://stacks.iop.org/0957-0233/13/1450

[73] G. Slabaugh, R. Schafer, and M. Livingston. Optimal ray intersection for computing 3d points from n-view correspondences. pp. 1–11, 2001.

[74] M. Sheehan, A. Harrison and P. Newman. Automatic Self-Calibration Of A Full Field-Of-View 3D n-Laser Scanner. *International Symposium on Experimental Robotics*, 2010.

[75] R. Simmons and D. Apfelbaum. A task description language for robot control. *Proceedings of the Conference on Intelligent Robots and Systems (IROS)*, 1998.

[76] D. Steinhauser, O. Ruepp and D. Burschka. Motion segmentation and scene classification from 3D LIDAR data. *Intelligent Vehicles Symposium, IEEE*, 2008.

[77] A. Teichman, J. Levinson and S. Thrun. Towards 3D Object Recognition via Classification of Arbitrary Object Tracks. *International Conference on Robotics and Automation*, 2011.

[78] C. Thorpe and H. Durrant-Whyte. Field robots. *ISRR*, 2001.

[79] S. Thrun, W. Burgard and D. Fox. Probabilistic Robotics. *MIT Press*, 2005.

[80] S. Thrun. What we're driving at. *The Official Google Blog*, 2010. At http://googleblog.blogspot.com/2010/10/what-were-driving-at.html

[81] S. Thrun and M. Montemerlo. The GraphSLAM algorithm with applications to large-scale mapping of urban structures. *IJRR*, 25(5/6), 2005.

[82] N. Tomatis, I. Nourbakhsh, and R. Siegwart. Hybrid simultaneous localization and map building: closing the loop with multi-hypothesis tracking. *ICRA*, 2002.

[83] I. Ulrich and I. Nourbakhsh. Appearance-based obstacle detection with monocular color vision. *AAAI*, 2000.

[84] J. Underwood, A. Hill, and S. Scheding. Calibration of range sensor pose on mobile platforms. *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Diego, CA, October 2007.

[85] C. Urmson, J. Anhalt, M. Clark, T. Galatali, J.P. Gonzalez, J. Gowdy, A. Gutierrez, S. Harbaugh, M. Johnson-Roberson, H. Kato, P.L. Koon, K. Peterson, B.K. Smith, S. Spiker, E. Tryzelaar, and W.L. Whittaker. High speed navigation of unrehearsed terrain: Red Team technology for the Grand Challenge 2004. *TR CMU-RI-TR-04-37*, 2004.

[86] C. Urmson, J. Anhalt, D. Bagnell, et al. Autonomous driving in urban environments: Boss and the Urban Challenge. *Journal of Field Robotics*, 25(8) pp. 425-466, 2008.

[87] C.-C. Wang, C. Thorpe, and S. Thrun. Online simultaneous localization and mapping with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas. *ICRA*, 2003.

[88] Willow Garage. Camera Calibration and 3d Reconstruction http://opencv.willowgarage.com/documentation/cpp/camera_calibration_and_3d_reconstruction.html

[89] M. Werlind, J. Ziegler, S. Kammel and S. Thrun. Optimal trajectory generation for dynamic street scenarios in a Frenet Frame. *International Conference on Robotics and Automation*, 2010, pp. 987-993.

[90] World Health Organization. Road traffic deaths. http://www.who.int/gho/road_safety/mortality/traffic_deaths_number/en/index.html updated 2011.

[91] J. Ziegler, M. Werlind and J. Schroder. Navigating car-like robots in unstructured environments using an obstacle sensitive cost function. *2008 IEEE Intelligent Vehicles Symposium*, pp. 787-791.