ROBOTIC LOCALIZATION AND PERCEPTION IN STATIC TERRAIN AND DYNAMIC URBAN ENVIRONMENTS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Isaac Thomas Miller

January 2009

ROBOTIC LOCALIZATION AND PERCEPTION IN STATIC TERRAIN AND DYNAMIC URBAN ENVIRONMENTS

Isaac Thomas Miller, Ph.D.

Cornell University 2009

This dissertation presents a complete, real-time, field-proven approach to robotic localization and perception for full-size field robots operating outdoors in static terrain and dynamic urban environments. The approach emphasizes formal probabilistic yet efficient frameworks for solving salient problems related to robotic localization and perception, including 1) estimating robot position, velocity, and attitude by fusing GNSS signals with onboard inertial and odometry sensors, 2) aiding these navigation solutions with measurements from onboard landmark sensors referencing a pre-surveyed map of environmental features, 3) estimating the locations and shapes of static terrain features around the robot, and 4) detecting and tracking the locations, shapes, and maneuvers of dynamic obstacles moving near the robot. The approach taken herein gives both theoretical and data-driven accounts of the localization and perception algorithms developed to solve these problems for Cornell University's 2005 DARPA Grand Challenge robot and 2007 DARPA Urban Challenge robot.

The approach presented here is divided into four main components. The first component statistically evaluates variants of an Extended Square Root Information Filter fusing GNSS signals with onboard inertial and odometry sensors to estimate robot position, velocity, and attitude. The evaluation determines the filter's sensitivity to map-aiding, differential corrections, integrity monitoring, WAAS augmentation, carrier phases, and extensive signal black-

outs. The second component presents the PosteriorPose algorithm, a particle filtering approach for augmenting robotic navigation solutions with vision-based measurements of nearby lanes and stop lines referenced against a known map. These measurements are shown to improve the quality of the navigation solution when GNSS signals are available, and they keep the navigation solution converged in extended signal blackouts. The third component presents a terrain estimation algorithm using Gaussian sum elevation densities to model terrain variations in a planar gridded elevation model. The algorithm is validated experimentally on the 2005 Cornell University DARPA Grand Challenge robot. The fourth component presents the LocalMap tracking algorithm, a real-time solution to the joint estimation problem of data assignment and dynamic obstacle tracking from a potentially moving robot. The algorithm is validated in controlled experiments with full-size vehicles, and on data collected at the 2007 DARPA Urban Challenge.

BIOGRAPHICAL SKETCH

Isaac Thomas Miller received his Bachelor of Science degree in Mechanical Engineering from the California Institute of Technology in 2003. While at Caltech, Isaac was awarded the Chester A. Boggs Scholarship for Academic Merit in 2000 and the Achievement Reward for College Scientists, California Chapter, from 2000 to 2003. Isaac was also awarded membership to the Tau Beta Pi Engineering Honor Society, California Beta Chapter, in 2003. During his time at Caltech, Isaac worked as a software engineer for Reynolds & Reynolds and The Relizon Company. Isaac also worked as a research fellow for Qualia Computing, Inc. and CADx Systems, developing pattern recognition software for mechanical, financial, and medical detection and classification systems.

Isaac received his Master of Science degree in Mechanical Engineering from Cornell University in 2006. While pursuing his Master of Science degree, Isaac was awarded Cornell University's McManus Fellowship in 2003 and a National Science Foundation Graduate Research Fellowship from 2004 to 2007. Isaac was also a finalist in the Fannie and John Hertz Foundation's Hertz Fellowship competition in 2004. Much of Isaac's Master of Science degree is a representation of his contributions as a central member of Cornell University's 2005 DARPA Grand Challenge team, for which he developed a position, velocity, and attitude estimator, a real-time terrain estimator, and a full-size robotic control and simulation scheme. Isaac's research ultimately helped Cornell's 2005 team qualify as one of twenty-three teams invited to the final 2005 DARPA Grand Challenge competition in Primm, Nevada.

Since receiving his Master of Science Degree, Isaac has been pursuing a Doctor of Philosophy degree in Mechanical Engineering from Cornell University. Work for this degree is closely related to Isaac's field research for Cornell Uni-

versity's 2007 DARPA Urban Challenge team. Isaac developed several critical systems for Cornell University's Urban Challenge robot, including a position, velocity, and attitude estimator, a multitarget tracking algorithm, and a map-aiding algorithm. Isaac also developed several supporting technologies for Cornell University's Urban Challenge robot, including a road tracking algorithm, a parameter estimation and calibration algorithm, and a vehicle steering, engine, and transmission simulation. Isaac's research helped Cornell University's Urban Challenge robot become one of eleven robots invited to the final 2007 DARPA Urban Challenge in Victorville, California, and one of only six robots to complete the challenge successfully. Isaac was also recently awarded the 2008 IEEE / ION Position Localization and Navigation Symposium's best student paper award for a sensitivity analysis of the position, velocity, and attitude estimator he developed for Cornell University's Urban Challenge robot.

Isaac's current research interests include robotic perception and planning in the context of Bayesian estimation and probabilistic representations, sensor fusion and Bayesian estimation, position, velocity, and attitude estimation, real-time implementation of robotic systems and algorithms, control, and dynamical modeling and simulation.

For my mother, who gave me life, fed me and raised me, taught me to keep going, and never asked for a single thing in return.

ACKNOWLEDGEMENTS

The author would like to thank the members of the Cornell University DARPA Grand Challenge and DARPA Urban Challenge teams, especially Jason Catlin, Frank-Robert Kline, Mike Kurdziel, Sergei Lupashin, Peter Moran, Aaron Nathan, Brian Schimpf, Alan Turnquist, and Noah Zych, for providing support during data collection, development, and testing. The author also thanks Marc Emond and Peter Moran for their photography work.

The author would like to thank Cornell University's corporate team sponsors, especially Septentrio Satellite Navigation and Singapore Technologies Kinetics, for sponsoring Cornell University and for hardware support.

The author also graciously acknowledges the support of his committee members: Mark Campbell, Mark Psiaki, Ephrahim Garcia, Bart Selman, and Dan Huttenlocher, for facilitating and aiding in the research documented herein. The author would also specifically like to thank his adviser, Mark Campbell, for his support and enthusiasm for taking on new projects.

The author also acknowledges the tireless educational efforts of Mark Brooks-Hedstrom, John Loomis, Mark Lutz, and Susan Taylor. The author especially acknowledges the mind-opening, paradigm-shifting enrichment of Bill and Carolyn Mowry, who have the patience of ten saints wrapped into two.

Finally, the author would like to thank his motley family: James Wilbur Wright, Mary-Lou Wright, Barbara Wright, Charlie Neal, Grace Miller, Daniel Lazarz, and Emily Runnells, for everything else.

This material is based upon work supported under a National Science Foundation Graduate Research Fellowship. This work is also supported by the DARPA Urban Challenge program (contract no. HR0011-06-C-0147), with Dr. Norman Whitaker as Program Manager.

TABLE OF CONTENTS

Biographical Sketch . . . iii
Dedication . . . v
Acknowledgements . . . vi
Table of Contents . . . vii
List of Tables . . . ix
List of Figures . . . x

1 Introduction **1**

2 Tightly-Coupled GPS / INS System Design for Autonomous Urban Navigation **6**
2.1 Introduction . . . 6
2.2 Cornell University's 'Skynet' . . . 8
2.2.1 Skynet's System Architecture . . . 9
2.2.2 Skynet's Localization Hardware . . . 12
2.3 Cornell's Attitude And Position Estimation Algorithm . . . 15
2.3.1 A Brief Overview Of The Square Root Information Filter . . . 15
2.3.2 The Extended Square Root Information Filter . . . 19
2.3.3 The Attitude And Position Estimation Algorithm . . . 20
2.4 Sensitivity Analysis In The Design Of The Pose Estimator . . . 24
2.4.1 Sensitivity To Map Aiding . . . 29
2.4.2 Sensitivity To Differential Corrections . . . 31
2.4.3 Sensitivity To Filter Integrity Monitoring . . . 34
2.4.4 Sensitivity To WAAS Corrections . . . 37
2.4.5 Sensitivity To GPS Carrier Phases . . . 38
2.4.6 Sensitivity To Signal Blackouts . . . 41
2.5 Conclusion . . . 46

3 Particle Filtering for Map-Aided Localization in Sparse GPS Environments **49**
3.1 Introduction . . . 49
3.2 Problem Description . . . 51
3.3 The PosteriorPose Algorithm . . . 52
3.3.1 A Brief Review Of Recursive Particle Filtering . . . 52
3.3.2 Generating The Proposal Distribution Via Onboard Inertial Navigation . . . 56
3.3.3 Updating Particle Weights With Absolute And Relative Sensor Cues . . . 58
3.4 Implementation And Hardware Setup . . . 67
3.5 Experimental Results . . . 71
3.5.1 Course 1: Dense Road Information . . . 73
3.5.2 Course 2: Sparse Road Information . . . 80
3.6 Conclusion . . . 85

4 A Mixture-Model Based Algorithm for Real-Time Terrain Estimation 87
4.1 Introduction . . . 87
4.2 Terrain Estimation Algorithm . . . 89
4.2.1 Statistical Treatment of Sensor Measurements . . . 90
4.2.2 Measurement Association . . . 97
4.2.3 In-cell Terrain Measurement Fusion . . . 99
4.2.4 Algorithm Benefits and Real-Time Implementation . . . 103
4.3 A Simple Example . . . 106
4.4 Real World Application: A Moving Ground Vehicle Equipped with Multiple Sensors . . . 116
4.5 Conclusions . . . 129

5 Stable and Efficient Tracking of Multiple Dynamic Obstacles Under Large Viewpoint Changes 131
5.1 Introduction . . . 131
5.2 The LocalMap Tracking Algorithm . . . 135
5.2.1 The Discrete Data Assignment Problem . . . 137
5.2.2 The Continuous Tracking Problem . . . 144
5.3 Experimental Performance . . . 158
5.3.1 Experiment 1: Perpendicular Intersection Encounter . . . 162
5.3.2 Experiment 2: Parallel Head-On Encounter . . . 168
5.3.3 Experiment 3: Multiple Obstacle Tracking In Heavy Traffic 173
5.4 Summary Of Performance In The DARPA Urban Challenge . . . 177
5.5 Conclusion . . . 178

6 Conclusion 180
6.1 Summary of Contributions . . . 183

Bibliography 186

LIST OF TABLES

2.1 Summary of Statistical Similarity Between Pose Estimator Variants 48

LIST OF FIGURES

2.1 Cornell University's 'Skynet,' an autonomous Chevrolet Tahoe that completed the 2007 DARPA Urban Challenge. 8
2.2 System architecture of Skynet. 11
2.3 An overhead view of the DARPA Urban Challenge NQE Area C course. Skynet's ground track, reconstructed from data logs taken at the Urban Challenge, is plotted in blue. Rectification and registration errors have not been entirely removed from the image. 26
2.4 A bird's eye view of the DARPA Urban Challenge NQE Area C course. The 20 minute compact course includes tree cover, low buildings, and power lines, making it ideal for unit tests against the pose estimator. Rectification and registration errors have not been entirely removed from the image. 27
2.5 Skynet's position when the baseline pose estimator with map aiding produces a statistically different pose estimate than the pose estimator without. Significant differences tend to occur after sharp turns, where map aiding provides positioning cues along new directions. 31
2.6 When denied differential corrections, the pose estimator produces a ground track with a mean error of 1.19 m. Although the mean error is approximately one vehicle width, the pose solution has no sudden discontinuities that would be devastating to autonomous driving. 33
2.7 Although denied differential corrections (DC), the magnitude of the pose estimator's updates are still very small. The position solution remains precise and robust, two features critical for autonomous driving. 34
2.8 When filter integrity monitoring (FIM) is turned off, the pose estimator suffers from large discontinuities. 36
2.9 Incorporating Doppler shifts and carrier phases (CP) without online validation yields a less robust pose solution than ignoring those measurements entirely. 40
2.10 Square roots of singular values of the covariance matrices for the baseline pose estimator and one in which differential corrections have been blacked out at mission time 5539.0. Although the pose solution degrades rapidly when differential corrections are first lost, the degradation tapers to a slow pace. 43
2.11 Euclidean error from baseline for two pose variants in a GPS blackout: one using ABS wheel speed measurements (Blackout Pose) and one not using ABS wheel speeds (No ABS Blackout Pose). All external satellite positioning signals have been blacked out at mission time 5539.0. 44

2.12 Square roots of singular values of the covariance matrices for pose estimators with and without wheel speed measurements. All external satellite positioning signals have been blacked out at mission time 5539.0. 45

3.1 The PosteriorPose algorithm is implemented in real-time on Cornell University's 'Skynet,' an autonomous 2007 Chevrolet Tahoe that completed the 2007 DARPA Urban Challenge. 68
3.2 Variants of the PosteriorPose algorithm run with no GPS blackouts on a course with dense road information. The PosteriorPose algorithm produces lower errors on average than the GPS / INS solution, even with GPS fully available. When additionally estimating GPS biases, the PosteriorPose algorithm uses road information to significantly reduce the effects of time-correlated GPS errors. 74
3.3 Differences in error between the fused GPS / INS solution and the PosteriorPose algorithm estimating GPS biases. The PosteriorPose algorithm uses road information to significantly reduce the effects of time-correlated GPS errors. 75
3.4 The PosteriorPose algorithm's model-based approach to resolving position ambiguity correctly estimates time-correlated GPS biases. 77
3.5 The PosteriorPose algorithm remains converged in a 30 minute extended GPS blackout using relative sensor cues. In contrast, the integrated INS solution drifts due to numerical integration errors. 78
3.6 Skynet's ground track during a 30 minute extended GPS blackout according to the truth data, the PosteriorPose algorithm, and the integrated INS solution. The PosteriorPose algorithm is able to use relative sensor cues to eliminate drift in the INS solution caused by numerical integration errors. 79
3.7 Variants of the PosteriorPose algorithm run with no GPS blackouts on a course with sparse road information. The PosteriorPose algorithm produces lower errors on average than the GPS / INS solution, and performance improvements are largely unaffected by the changes in the availability of road cues. 81
3.8 The PosteriorPose algorithm estimates GPS biases on a course with sparse road information. Errors in these biases are similar to those accrued on a course with dense road information. 82
3.9 The PosteriorPose algorithm remains converged in a 30 minute extended GPS blackout despite sparse relative sensor cues. An integrated INS algorithm does not remain converged, even when initialized from moving vehicle data. 83

3.10 Skynet's ground track during a 30 minute extended GPS black-out on a course with sparse road information. The PosteriorPose algorithm is able to remain converged using road cues, even on a course with less stop lines and half the turns. 84

4.1 Problem geometry of the simulated one-dimensional cart and rangefinder. 107
4.2 True terrain, terrain estimate $\hat{U}_{GM}$, and $\pm 2\sigma_{GM}$ bounds for a one-dimensional terrain example with cart moving at 5 m/s. 111
4.3 (Left) Terrain estimate measurement probability mass levels and number of measurements assigned for one-dimensional terrain example. High probability mass in this example indicates the presence of a vertical or near-vertical face. (Right) Terrain estimate measurement probability mass level and total number of measurements accumulated over time for terrain cell $j = 50$. . . . 111
4.4 (Left): True terrain, terrain estimate $\hat{U}_{GM,j}$, $\pm 2\sigma_{GM,j}$ bounds, and minimum and maximum measurements accumulated over time for terrain cell $j = 50$ with vehicle moving at 5 m/s. (Middle): The same quantities with vehicle moving at 0.5 m/s. (Right): The same quantities with vehicle moving at 25 m/s. 114
4.5 (Left): True terrain, terrain estimate $\hat{U}_{GM,j}$, $\pm 2\sigma_{GM,j}$ bounds, and minimum and maximum measurements near a simulated ditch with vehicle moving at 5 m/s. (Middle): The same quantities with vehicle moving at 0.5 m/s. (Right): The same quantities with vehicle moving at 25 m/s. 115
4.6 (Left): True terrain, terrain estimate $\hat{U}_{GM,j}$, $\pm 2\sigma_{GM,j}$ bounds, and minimum and maximum measurements near a simulated wall with vehicle moving at 5 m/s. (Middle): The same quantities with vehicle moving at 0.5 m/s. (Right): The same quantities with vehicle moving at 25 m/s. 116
4.7 The Spider Light Strike Vehicle used to test the terrain estimation algorithm. 118
4.8 The Spider's three laser rangefinders (LIDARs) and two-axis gimbal platform. 119
4.9 Example sensor axes and ENU reference axes that determine the transformation from sensor to terrain measurements. 121
4.10 Sample $\hat{U}_{GM} + 2\sigma_{GM}$ elevation map resulting from the Spider's sensor fusion scheme. 124
4.11 (Left) Final $\hat{U}_{GM}$ map near the two 0.8 m tall trash cans. (Right) Final σ_{GM} map near the two 0.8 m tall trash cans. 125

4.12 (Left) Total association probability and $\hat{U}_{GM} \pm 2\sigma_{GM}$ bounds over time for a particular terrain cell containing a portion of a trash can. (Right) Vehicle ground track during real-time terrain experiment. The vehicle intersects the line connecting the trash cans at $t \approx 2358$ sec., $t \approx 2374$ sec. and $t \approx 2395$ sec. 125
4.13 (Left) Vehicle speed during real-time terrain experiment. (Right) Vehicle heading during real-time terrain experiment. 127
4.14 'Phantom' walls in the $\hat{U}_{GM} + 2\sigma_{GM}$ map arising from high variance due to incorrectly-modeled gimbal encoder noise. 128

5.1 Coordinate frames used for tracking a single obstacle (a moving vehicle) under known measurement assignments. In the LocalMap, an obstacle's position is defined by the x and y location of a reference point p_0 fixed to the obstacle. 147
5.2 Locus of point cloud centers of mass observed as a moving vehicle passes in front of a stationary laser rangefinder. The apparent drift in the observed center of mass results from dramatic changes in the sensed shape of the moving vehicle as it passes in front of the laser rangefinder. This sensitivity, appearing in even mildly dynamic environments, causes instability in feature extraction algorithms such as center of mass, edge, and corner detectors applied to laser rangefinder data. 152
5.3 The LocalMap is implemented in real-time on Cornell University's 'Skynet,' an autonomous 2007 Chevrolet Tahoe that completed the 2007 DARPA Urban Challenge. 159
5.4 Sensor placement and coverage diagram for Skynet's laser rangefinders. Skynet sits in the center of the diagram, facing right. Redundant laser rangefinders cover the front of Skynet's field of view, where it encounters the most dangerous moving obstacles. 160
5.5 Sensor placement and coverage diagram for Skynet's radars. Skynet sits in the center of the diagram, facing right. Radars are placed to detect oncoming cars in opposing lanes, from the left and right in merging situations, and from the rear in passing situations. 160
5.6 In the first experiment, the LocalMap tracks a moving target vehicle as it crosses the ego vehicle's path at an intersection. 163
5.7 LocalMap range tracking errors to the closest point on a moving target vehicle over 11 perpendicular intersection encounters with the target approaching from the right at 15 mph. Error statistics are calculated across encounters according to the true bearing of the target's leading edge. Of the 284 average range errors considered, 251 are within 20 cm of zero at the 5% significance level. 164

5.8 LocalMap ground speed tracking errors over 11 perpendicular intersection encounters with a moving target vehicle approaching from the right at 15 mph. 166
5.9 LocalMap relative heading tracking errors over 11 perpendicular intersection encounters with a moving target vehicle approaching from the right at 15 mph. The LocalMap's point cloud representation is able to estimate relative heading correctly despite significant changes in target vehicle viewpoint. 167
5.10 In the second experiment, the LocalMap tracks a moving target vehicle as it approaches the moving ego vehicle from the opposite direction. Both vehicles travel at approximately 15 mph for these parallel head-on encounters. 168
5.11 LocalMap range tracking errors to the closest point on a moving target vehicle over 11 parallel head-on encounters at 30 mph closing speeds. Error statistics are calculated across encounters according to the true range to the target vehicle. In the 284 true ranges to the target vehicle considered, 153 are statistically equal to zero, and 233 are within 20 cm of zero at the 5% significance level. 169
5.12 LocalMap ground speed tracking errors over 11 parallel head-on encounters at 30 mph closing speeds. 171
5.13 LocalMap relative heading tracking errors over 11 parallel head-on encounters at 30 mph closing speeds. 172
5.14 In the third experiment, variants of the LocalMap are run on excerpts of data from a DARPA Urban Challenge qualifying round. In this data, Skynet autonomously completes multiple merges into and out of moving traffic across a lane of oncoming vehicles. 174
5.15 Sample cumulative distribution function of errors in the number of obstacles tracked in a heavy traffic scenario in variants of the LocalMap run with 1, 2, 5, 10, and 20 particles. Errors are calculated against a LocalMap variant run with 50 particles. The LocalMap algorithm's convergence to a common number of tracked obstacles as the number of particles increases shows stability in selection of data assignments. 175

CHAPTER 1

INTRODUCTION

Human attrition incurred in a recent seemingly-endless series of military occupations has given full-size field robotics public and political attention it has not seen since the early days of artificial intelligence. News of roadside bombings during routine supply missions, aircraft downed during routine patrols, and hand-to-hand attacks during routine village walk-throughs continue to dot headlines around the world- and continue to reach the ears of an increasingly-critical public each day. Few would argue the problem, that guerrilla tactics have emerged as the primary means for non-governmental groups to lay waste to infrastructure in an unconventional bid for power. Yet many would argue that the cure, which rebuilds the infrastructure on the backs of volunteering soldiers, is far from optimal. With so many lives lost on routine patrols, supply missions, and other repetitive tasks, many look to full-size ground robots to take their turn in the field.

Politicians have readily embraced the idea of removing humans from direct line of fire in some of the more repetitive and dangerous military tasks. The United States in particular has been interested in automated and remote control warfare, setting a goal to have one-third of all ground combat vehicles be unmanned by 2015 [1]. This and other political inspiration, coupled with ambitious foresight, led the Department of Defense's Defense Advanced Research Projects Agency (DARPA) to hold the DARPA Grand Challenge in 2004. The 2004 Grand Challenge was appropriately named, requiring competitors to build full-size robots to navigate 142 miles of difficult desert terrain between Barstow, California and Primm, Nevada without human intervention [35]. Un-

fortunately, the challenge was too grand: of the planned 142 miles, only 7.4 were ever visited by a robot.

The difficulty in the 2004 DARPA Grand Challenge wasn't caused by a lack of technology, as robots wielding primarily the same pieces of hardware returned to the Mojave desert to complete a 132 mile course in the 2005 DARPA Grand Challenge [73]. No, the 2004 Grand Challenge served as a wake-up call: it revealed a significant disconnect between the state of the art of algorithms developed for academic research and those developed for action in the field. Whereas the former had developed to focus on elegant probabilistic mapping formulations and provably-optimal planning algorithms, the latter still struggled with fast, *ad hoc*, greedy algorithms wrapped around off-the-shelf components [90, 74, 14]. In fact, many root causes of robot failure in the 2004 DARPA Grand Challenge could be traced to minor hardware and software integration mistakes [14].

In contrast, the 2005 DARPA Grand Challenge showcased progress made toward unifying theoretical and practical robotics. Sebastian Thrun and the Stanford University racing team, winners of the 2005 Grand Challenge, designed many of their higher-level processing algorithms from scratch [93]. Instead of purchasing an off-the-shelf position, velocity, and attitude estimator, Thrun *et al.* elected to write their own to include problem-specific constraints not available in off-the-shelf systems. Rather than blindly trust sensor output and *ad hoc* decision criteria to discover obstacles, Thrun *et al.* developed a probabilistic representation of their sensor information and used machine learning techniques to find optimal decision boundaries. And finally, instead of implementing a canned greedy local path planner, Thrun *et al.* learned driving parameters

from human exemplars and checked paths online against dynamic vehicle constraints. In total, the solution adopted a theoretical view of field robotics, emphasizing a clear understanding of sensor and hardware uncertainty without sacrificing simplicity.

In a larger sense, the 2005 DARPA Grand Challenge identified three primary barriers to a successful field deployment: localization, perception, and planning [57]. The localization problem is, most generally, the task of determining the robot's current configuration with respect to its goals. In full-size ground robots, localization primarily refers to determining the robot's position, velocity, and attitude in a map that includes the robot's goals. Successful localization in this context mandates a clear understanding and correct treatment of sensing errors, particularly when the map is incorrect. The perception problem is the task of making sense of the robot's surroundings from noisy and incomplete sensor data. Implicitly, solutions to the perception problem must understand and model sensor errors in order to identify static obstacles, dynamic and possibly adversarial or cooperative agents, and other environmental features which may relate to or hinder the robot's goals. It also includes the task of modeling the environment efficiently to facilitate the robot achieving its goals without wasting resources. Finally, the planning problem is the task of determining and executing a series of actions to achieve the robot's goals. Successful planning, in general, implies a clear understanding of the effects of the robot's actions on itself and its environment.

This dissertation presents a complete, real-time, field-tested solution to two of these problems: robotic localization and perception, for full-size field robots operating outdoors in static terrain and dynamic urban environments. Chapter

2 begins by presenting a tightly-coupled position, velocity, and attitude (pose) estimator used as a position feedback signal for autonomous navigation in Cornell University's 2007 DARPA Urban Challenge robot, 'Skynet.' A rigorous statistical sensitivity analysis is performed on the pose estimator as different inputs and design features are removed. All pose estimator variants are scrutinized both in a statistical sense and in a practical sense, by comparing each variant's performance on logged data recorded at the 2007 DARPA Urban Challenge. The relative performance between the pose estimator variants offers new insight into the effectiveness of pose estimator design decisions in relation to full-size autonomous navigation. Chapter 3 expands upon the localization solution by presenting a map-aiding algorithm for augmenting a robotic pose estimator with additional positioning cues obtained by observing landmarks referenced from a pre-surveyed map of environmental features. The algorithm is implemented as a particle filter; it fuses GPS with lane and stop line measurements taken from optical cameras to improve the overall quality of the pose solution. These measurements are incorporated with careful hypothesis testing and error modeling to account for non-Gaussian and multi-modal errors committed by GPS and vision-based detection algorithms. Accompanying experimental data shows the map-aiding algorithm outperforms a traditional pose estimator and remains converged even in 30 minute GPS blackouts. Chapter 4 switches gears to the robotic perception problem by presenting a grid-based algorithm to efficiently estimate and map terrain in a static outdoor environment traversed by a full-size robot. The algorithm utilizes a formal probabilistic analysis to incorporate and account for multiple sources of sensing error, both in-plane and out of plane, in a rigorous manner. The approach is constructed such that terrain estimates and estimation error statistics can be calculated in real-time without

maintaining a history of sensor measurements, and it is validated experimentally on Cornell University's 2005 DARPA Grand Challenge robot. Chapter 5 completes the approach to the robotic perception problem by presenting a computationally feasible, real-time algorithm that solves the joint estimation problem of data assignment and dynamic obstacle tracking from a potentially moving robotic platform. The algorithm utilizes a Bayesian factorization to separate the joint estimation problem into 1) a data assignment problem solved via particle filter, and 2) a multiple dynamic obstacle tracking problem solved by efficient parametric filters developed specifically for tracking full-size vehicles in a dense traffic environment. The algorithm is validated in controlled experiments with full-size vehicles, and on data collected at the 2007 DARPA Urban Challenge. Chapter 6 concludes with a summary of the localization and perception algorithms presented in this dissertation, along with a specific discussion of the contributions these algorithms have made to field robotics.

CHAPTER 2

TIGHTLY-COUPLED GPS / INS SYSTEM DESIGN FOR AUTONOMOUS URBAN NAVIGATION

2.1 Introduction

Global navigation satellite system signals such as GPS and dead reckoning inertial navigation systems (INS) are well known to be an ideal pair for sensor fusion and temporal filtering. The resulting fused position, velocity, and attitude (pose) solution has seen a staggering number of academic, commercial, and defense applications, including navigation, surveying, and guidance [88, 72, 3, 6, 68, 11]. Such widespread and varied use of the GPS / INS sensing pair has resulted in a multitude of approaches for fusing the available information, depending on the type and characteristics required of the outputs. Three common integration strategies, loose coupling, tight coupling, and ultra tight coupling, fuse GPS with INS at various levels of GPS processing [3]. New emerging strategies seek to augment this basic level of coupling with application-based motion constraints and measurements of nearby landmarks, either from maps known *a priori* or generated in real-time [17, 23, 55, 31].

One area where GPS / INS fusion is becoming increasingly important is robotics, where the fused GPS / INS pose solution is often used as a feedback signal for path tracking and decision making [88, 31, 58, 55]. The requirements of the pose solution in this application differ substantially from requirements in other applications. Here, precision and robustness are far more important than accuracy, and emphasis is typically placed on obtaining a statistically consistent pose solution rather than an accurate one [88]. In no place was this more

apparent than the 2005 DARPA Grand Challenge, a United States government-sponsored 130 mile desert race in which the only competitors were robots. There, brittleness in Cornell University's pose estimator resulted in instability and failure after only 9 miles of travel [58].

Two years later, the United States government sponsored a new robotic race, the 2007 DARPA Urban Challenge. This race was to take place in an urban setting, with each full size robot to complete a 60 mile set of mock supply missions without human intervention, in a city filled with traffic. Cornell's 2005 pose estimator was rewritten for the Urban Challenge, under full awareness of the lessons learned in the Grand Challenge and the new difficulties present in the urban environment. Ultimately the pose estimator was successful, and Cornell's robot was one of only six to complete the entire 60 mile Urban Challenge [15, 56].

The remainder of this paper explains and analyzes the design decisions made in developing the pose estimator for Cornell University's 2007 DARPA Urban Challenge robot. Section 2.2 introduces the robot, called 'Skynet,' with a brief discussion of its system architecture and hardware. Section 2.3 gives an overview of Cornell's pose estimator algorithm, a tightly-coupled Extended Square Root Information Filter, and the important facets of its design. Section 2.4 examines the pose estimator's performance through a sensitivity analysis, whereby variants of the pose estimator are run on logged Urban Challenge sensor data to discern the effects of each critical design decision. Section 2.5 concludes with a statistical summary of the pose estimator's sensitivity to its inputs, with emphasis placed on requirements for successful autonomous navigation.

Figure 2.1: Cornell University's 'Skynet,' an autonomous Chevrolet Tahoe that completed the 2007 DARPA Urban Challenge.

2.2 Cornell University's 'Skynet'

Cornell University's 'Skynet,' shown in Figure 2.1, is a full size 2007 Chevrolet Tahoe built to compete as Team Cornell's entry in the 2007 DARPA Urban Challenge. Skynet is fully autonomous: completely actuated, filled with computers and a network architecture, equipped with sensors and probabilistic perception algorithms, and outfitted with navigation and traffic planning algorithms. It is capable of navigating and interacting with a populated urban environment, obeying California state driving laws without human intervention [15, 56].

Between October 25, 2007 and October 31, 2007, Skynet participated in the Urban Challenge National Qualifying Event (NQE), a set of short 25-minute

autonomous urban missions designed by DARPA to evaluate candidates for selection into the Urban Challenge Event (UCE) itself. Completing these events safely and successfully, Skynet was selected as one of eleven robots allowed to compete in the 2007 DARPA Urban Challenge. On November 3, 2007, after approximately 6 hours and 60 miles of autonomous urban driving, Skynet successfully completed the DARPA Urban Challenge. It was one of only six robots to finish.

2.2.1 Skynet's System Architecture

Skynet interacts with the world according to the action, sensing, and planning subsystems shown as general system blocks in Figure 2.2. The vehicle block, denoted 'vehicle' in Figure 2.2, is the lowest level of these subsystems. It comprises both the hardware and software aspects of Skynet's steering, brake, throttle, and transmission actuation systems. These actuation systems, consisting of aerospace servomotors and low level software tracking loops, allow Skynet's steering wheel, brake pedal, electronic throttle, and transmission cable to achieve commands issued by the onboard computers.

The localization algorithm, denoted 'pose estimator' in Figure 2.2, lies above the vehicle block. It fuses information from external satellite signals with internal vehicle odometry to produce an attitude and position solution. This solution is supplied to Skynet's intelligent planner, where it is used both as a feedback signal for basic path tracking, and as an absolute localization signal for dictating higher level vehicle behaviors in traffic. The localization algorithm also produces a relative attitude and position solution, which relates Skynet-fixed

coordinate frames from previous times to the vehicle's current attitude and position.

The perception algorithms, denoted 'local map' and 'scene estimator' in Figure 2.2, operate in parallel with the localization algorithm. The perception algorithms process information from Skynet's onboard sensors: laser rangefinders, millimeter-wave radar, and optical cameras to interpret Skynet's surroundings for the intelligent planner. The local map fuses this incoming sensor information with relative attitude and position solutions from the localization algorithm to detect and track obstacles in a Skynet-fixed coordinate frame. The scene estimator fuses these tracked targets with the absolute attitude and position solution from the localization algorithm to assign absolute positions and lane occupancy probabilities to each tracked target. It also fuses the absolute solution with its own camera-based lane estimates to improve the final attitude and position solution that is passed to the planner.

The intelligent planning algorithm, denoted 'behavioral layer,' 'tactical layer,' and 'operational layer' in Figure 2.2, considers all low level feedback, localization, and obstacle information provided by the other subsystems to plan and follow routes to complete its assigned mission. This task consists of three levels of planning. First, the behavioral layer uses Skynet's current position and a graph search algorithm to plan an initial path through the navigable streets defined by DARPA's Route Network Definition File (RNDF) and Mission Data File (MDF). Second, the tactical layer guides Skynet along this high level route by selecting an efficient and law-abiding path over a short segment of the route. Finally, the path planned by the tactical layer is handed to the operational layer, which uses a constrained nonlinear optimization package to smooth the first 15

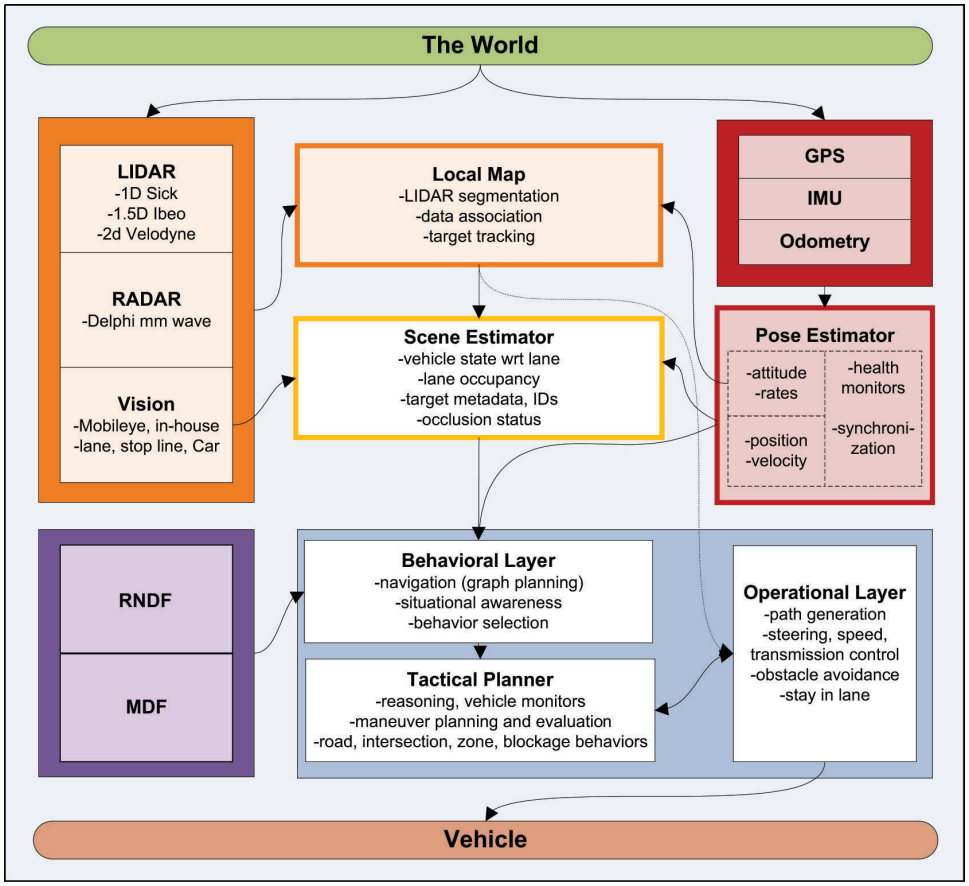

Figure 2.2: System architecture of Skynet.

to 30 m of the path into one that is physically achievable. The path generated over this small planning horizon is then discretized into speed and curvature commands to be achieved by the low level actuation controllers. This small path is tracked over time in a Skynet-fixed coordinate frame using the relative attitude and position solution from the localization algorithm. The path is discarded at the next planning cycle, when a new high level planned path takes its place.

2.2.2 Skynet's Localization Hardware

Skynet's position, velocity, and attitude (pose) solution, generated by the 'pose estimator' system block in Figure 2.2, is used in all higher level reasoning and perception components to facilitate safe autonomous urban driving. As the only source of absolute position information on Skynet, the pose estimator's accuracy, precision, robustness, and stability all directly influence how Skynet interprets and responds to its surroundings. Cornell University's pose estimator has therefore been designed with consideration for these four abstract qualities while satisfying the more concrete requirements of the Urban Challenge: staying in designated lanes, coming to a stop within 1 m of each stop line, parking in waypoint-defined parking spaces, and obeying other location-dependent traffic laws. The pose estimator has also been designed with regard to the difficulties of the urban GPS signal environment, including heavy multipath and frequent signal obstruction.

Skynet's pose estimator satisfies these basic design requirements by fusing information from reliable and high quality localization sensing hardware. Four sensors are fused: a Litton LN-200 inertial measurement unit (IMU), Skynet's antilock brake (ABS) wheel encoders, a Septentrio PolaRx2e@ three-antenna GPS receiver, and a Trimble Ag252 single-antenna high precision GPS receiver. The LN-200 is a combined three-axis fiber optic rate gyroscope and a three-axis silicon accelerometer. This tactical grade IMU is mounted on the floor of Skynet, along its centerline, just above the rear axle. The LN-200 integrates its rate gyros and accelerometers to produce estimates of differential vehicle rotation and motion at 400 Hz, and has superb 1 deg. / hr. gyro bias repeatability and 300 μg accelerometer bias repeatability [67]. Intuitively, the IMU is used to inte-

grate Skynet's pose solution forward in time with a dead reckoning numerical integration scheme. The IMU is augmented with Skynet's ABS wheel encoders, 64 count / rev. encoders mounted to each of Skynet's tires. Information from these encoders is obtained from Skynet's stock GM CAN network at 30 Hz data transmission rates. The information is used to calculate an average speed at the center of Skynet's rear axle, which is subsequently used to slow the rate of error growth in IMU integrations.

The two GPS receivers, the Septentrio PolaRx2e@ and the Trimble Ag252, are both used as absolute position signals to correct errors that grow over time in dead reckoning. The Septentrio PolaRx2e@, a three-antenna, single-clock, 48-channel GPS receiver provides raw pseudorange, Doppler shift, and carrier phase measurements of all visible GPS satellites on all its antennas at 5 Hz synchronized measurement intervals [83]. Its antennas are mounted in a characteristic 'L' pattern on Skynet's roof, spaced as far apart as possible to promote greater observability in the differential signals between the antennas. The PolaRx2e@ decodes the WAAS signal at the same 5 Hz synchronized measurement rate to permit the use of more accurate signal models. The PolaRx2e@ also reports raw pseudoranges to WAAS satellites, though those are not used in Skynet's pose estimator. In contrast, the Trimble Ag252 is a single-antenna GPS receiver, mounted on the centerline of Skynet's roof, just above the rear axle. In Skynet's pose estimator, the Ag252 is used solely to decode high precision (HP) and virtual base station (VBS) OmniSTAR differential corrections. These corrections are decoded at a rate of 5 Hz, synchronized on the GPS second according to the Ag252's clock [94]. With an advertised accuracy of ± 10 cm, these differential correction signals are the primary source of satellite-based sub-meter positioning information in Skynet's pose estimator.

The sensors chosen for the pose estimator complement each others' weaknesses to improve overall robustness and reliability in Skynet's pose estimator. The LN-200 provides accurate measurements of differential vehicle motion at the high data rates necessary to produce a smooth pose solution, but the rate measurements must be numerically integrated. The resulting IMU dead reckoning scheme is therefore subject to integration errors that increase over time, with no absolute position or orientation information to correct the integration errors [68]. Skynet's ABS wheel encoders help slow the growth rate of these accumulating integration errors by providing speed measurements, which are used to adjust estimates of biases in the LN-200 accelerometer. Although these measurements improve the quality of the pose solution during dead reckoning, they are subject to measurement errors and the effects of wheel slip. The PolaRx2e@ provides data at a much slower rate than the IMU, but its GPS- and WAAS-based absolute attitude and position information can be used to prevent IMU integration errors from growing without bound. These three sensing modalities cannot consistently achieve sub-meter accuracy, even when fused, but they can when updated with infrequent measurements of the differential correction signal made by the Ag252. The differential correction signal tends to be brittle, as it is often biased and difficult to track for extended periods of time. However, statistical hypothesis tests can be used to determine which differential correction measurements are most accurate. These tests, which depend on historical data accumulated from all four sensing modalities, make the differential correction signal beneficial even in challenging urban signal environments.

2.3 Cornell's Attitude And Position Estimation Algorithm

Skynet's four localization sensing modalities, described in section 2.2.2, are statistically fused via an Extended Square Root Information Filter (ESRIF) into Skynet's pose estimate. This section develops The ESRIF pose estimator at a high level, first describing the statistical formulation of the filter, and finishing with its algorithmic components.

2.3.1 A Brief Overview Of The Square Root Information Filter

The Square Root Information Filter (SRIF) is a numerically stable implementation of the Kalman filter (KF) [12]. Like the KF, the SRIF is a recursive filter designed to estimate the *a posteriori* density of a state vector $x(k)$ at time index k conditioned on an observed time series of sensor measurements $Z(k) = [z(1), z(2), \ldots, z(k)]$. Also like the KF, the SRIF operates under the assumption that $x(k)$ evolves in time according to the linear dynamics model:

$$x(k+1) = F(k)\,x(k) + G(k)\,u(k) + \Gamma(k)\,v(k) \tag{2.1}$$

where $F(k)$ is the *state transition matrix*, $G(k)$ is the *control input matrix* for known control input $u(k)$, and $\Gamma(k)$ is the *noise input matrix* for the unknown process noise vector $v(k)$. Similarly, the KF and SRIF assume that sensor measurements $z(k)$ are linear functions of the state:

$$z(k) = H(k)\,x(k) + w(k) \tag{2.2}$$

where $H(k)$ is the *measurement matrix*, and $w(k)$ is the unknown measurement noise vector. Most importantly, the KF and SRIF assume that the initial *a priori*

state density $p(x(0))$, the process noise density $p(v(k))$, and the measurement noise density $p(w(k))$ are all Gaussian:

$$p(x(0)) \sim \mathcal{N}(\hat{x}(0), P(0)) \tag{2.3}$$

$$p(v(k)) \sim \mathcal{N}(0, Q(k)) \tag{2.4}$$

$$p(w(k)) \sim \mathcal{N}(0, R(k)) \tag{2.5}$$

where $\hat{x}(0)$ is the initial *a priori* state estimate, $P(0)$ is the initial *a priori* state covariance matrix, and $Q(k)$ and $R(k)$ are the possibly time-varying covariance matrices of the process and measurement noise vectors, respectively. Both the process noise and measurement noise are assumed to be white noise random processes. Here they are restricted to be zero mean, mutually uncorrelated, and uncorrelated with the initial state estimate, though those restrictions need not be imposed in general [11].

Under the aforementioned assumptions, both the *a priori* state probability density $p(x(k)|Z(k-1))$ and the *a posteriori* state probability density $p(x(k)|Z(k))$ are Gaussian:

$$p(x(k)|Z(k-1)) \sim \mathcal{N}\left(\bar{x}(k), \bar{P}(k)\right) \tag{2.6}$$

$$p(x(k)|Z(k)) \sim \mathcal{N}(\hat{x}(k), P(k)) \tag{2.7}$$

allowing the KF to maintain only the mean and covariance matrix, sufficient statistics of the state probability density [11]. This compact form leads to the KF's familiar *prediction step*, where the system dynamics in equation 2.1, the current state estimate $\hat{x}(k)$, and the current state covariance matrix $P(k)$ are used to generate the predicted state estimate $\bar{x}(k+1)$ and covariance matrix $\bar{P}(k+1)$ conditioned on the measurement set $Z(k)$, and the KF's familiar *update step*, where the measurement function in equation 2.2, the predicted state

estimate $\bar{x}(k+1)$, and the predicted state covariance $\bar{P}(k+1)$ are used to generate the updated state estimate $\hat{x}(k+1)$ and covariance matrix $P(k+1)$ conditioned on the measurement set $Z(k+1)$. KF estimation continues recursively in this fashion as time advances, and as new measurements become available [11].

Although the SRIF makes exactly the same linear-Gaussian assumptions as the KF, it differs in its representation of the state probability density. Rather than representing the state probability density $p(x(k)|Z(k))$ by its mean $\hat{x}(k)$ and covariance matrix $P(k)$, the SRIF maintains a *square root information state* $\hat{y}(k)$ and a *square root information matrix* $R_{xx}(k)$ [12]. At all times the square root information state $\hat{y}(k)$ and matrix $R_{xx}(k)$ relate to the state mean $\hat{x}(k)$ and covariance $P(k)$ by the following definitions:

$$\hat{y}(k) = R_{xx}(k)\,\hat{x}(k) \tag{2.8}$$

$$R_{xx}^{T}(k) \cdot R_{xx}(k) = P^{-1}(k) \tag{2.9}$$

Analogous definitions extend to the prediction mean $\bar{x}(k)$ and covariance matrix $\bar{P}(k)$, and their square root information forms, $\bar{y}(k)$ and $\bar{R}_{xx}(k)$.

The SRIF maintains the state probability density as square root information variables $\hat{y}(k)$ and $R_{xx}(k)$ primarily for numerical reasons [12]. In particular, the square root information state $\hat{y}(k)$ effectively stores a version of the state vector $x(k)$ normalized by the uncertainty in that state. This form is especially useful in estimation problems in which large scaling discrepancies exist between state variables, as it attenuates numerical errors that may arise when inverting badly-scaled matrices during the KF update step. In addition, representing the state covariance matrix in square root information form effectively doubles numerical precision, as all matrix inversions performed on full covariance matrices

in the KF are performed on square root matrices in the SRIF. Furthermore, since equations 2.8 and 2.9 do not specify a particular matrix square root, stable and efficient factorization algorithms such as the Cholesky decomposition may be used in lieu of more expensive and potentially less stable eigenvector decompositions.

In addition to numerical stability, a second benefit to representing the state probability density as square root information variables $\hat{y}(k)$ and $R_{xx}(k)$ is the ability to initialize state estimates with infinite covariance. Because infinity cannot be properly represented on a digital computer, traditional KF implementations often approximate this *naïve Bayes* initialization by an initial covariance matrix $P(0)$ with 'sufficiently large' entries along the diagonal. This choice yields suboptimal state estimates while the KF converges, which may be particularly detrimental if one or more elements of the state vector are reset frequently. The SRIF does not suffer from this problem, as the *naïve Bayes* initialization is a situation in which one or more of the state variables have no *a priori* information. Such a situation is easily reflected in the square root information matrix $R_{xx}(k)$ by zeroing appropriate rows and columns.

Like the KF, the SRIF has analogous prediction and update steps. However, because these steps operate to maintain the information variables, they have a different form. Whereas the KF is a direct least squares solution to the exponents of its constituent Gaussians, the SRIF applies a QR-factorization to solve the same least squares problem with square root techniques [12].

2.3.2 The Extended Square Root Information Filter

Both the KF and the SRIF operate under the assumptions that the dynamics and measurement functions given in equations 2.1 and 2.2 are linear, and all random variables in the estimation problem are Gaussian. When either of those two assumptions are violated, an approximation to the KF, called the Extended Kalman Filter (EKF), is often used to perform suboptimal estimation [11]. In the EKF, the dynamics describing the evolution of the state need not be linear:

$$x\left(k+1\right) = f\left(x\left(k\right), u\left(k\right), v\left(k\right)\right) \tag{2.10}$$

Similarly, the measurement equation describing the sensor model also does not need to be linear:

$$z\left(k\right) = h\left(x\left(k\right), w\left(k\right)\right) \tag{2.11}$$

although the measurement noise $w\left(k\right)$ is often assumed to enter additively into the measurement. The *a priori* state probability density $p\left(x\left(0\right)\right)$, the process noise density $p\left(v\left(k\right)\right)$, and the measurement noise $p\left(w\left(k\right)\right)$ also do not need to be Gaussian, although in the EKF they are represented by their first and second moments as if they were.

One common variant of the EKF assumes the appropriate state estimation error, $\bar{e}\left(k\right) = x\left(k\right) - \bar{x}\left(k\right)$ or $\hat{e}\left(k\right) = x\left(k\right) - \hat{x}\left(k\right)$, is small. Under this assumption, the nonlinear dynamics of equation 2.10 and the nonlinear measurement of equation 2.11 are linearized about the state estimate $\bar{x}\left(k\right)$ or $\hat{x}\left(k\right)$ to yield linearized error dynamics:

$$\bar{e}\left(k+1\right) \approx \frac{\partial f}{\partial x}\hat{e}\left(k\right) + \frac{\partial f}{\partial v}v\left(k\right) \tag{2.12}$$

$$\nu\left(k+1\right) \approx \frac{\partial h}{\partial x}\bar{e}\left(k+1\right) + \frac{\partial h}{\partial w}w\left(k+1\right) \tag{2.13}$$

where $\nu(k) = z(k) - \bar{z}(k)$ is the *measurement innovation* [11]. Equations 2.12 and 2.13 restore linearity to the problem, allowing the EKF to utilize the KF's familiar linear matrix equations during predictions and updates. The price paid for the ease of linearity is optimality: the EKF is not guaranteed to produce minimum mean squared error (MMSE) estimates, unlike the KF.

The approximations made in equations 2.12 and 2.13 can also be used to create an Extended Square Root Information Filter (ESRIF) equivalent to the EKF [72]. Like the EKF, the ESRIF largely resembles its linear counterpart, with partial derivative Jacobian matrices replacing the state transition matrix $F(k)$, the noise input matrix $\Gamma(k)$, and the measurement matrix $H(k)$. The only notable differences are two small correction terms that arise in the information state prediction and update equations. These terms correct for the fact that equations 2.12 and 2.13 are linear in the error variables $\hat{e}(k)$ and $\bar{e}(k)$, not $\hat{x}(k)$ and $\bar{x}(k)$.

The ESRIF retains all the numerical advantages over the EKF that the linear SRIF possesses over the KF. The only additional downside to the ESRIF is the fact that any nonlinear elements of the dynamics function $f(\cdot)$ and the measurement function $h(\cdot)$ must be linearized about the estimated state $\hat{x}(k)$ or $\bar{x}(k)$. As a consequence, any states evolving according to the nonlinear dynamics or entering into the nonlinear measurement equations cannot be initialized with infinite covariance (zero information).

2.3.3 The Attitude And Position Estimation Algorithm

Skynet's pose estimator applies an ESRIF to statistically fuse its IMU, wheel encoders, and GPS receivers into a real-time feedback signal for autonomous

driving in an urban environment. At a high level, the pose estimator consists of 17 'core' states estimated at all times, plus numerous 'adjunct' states depending on the current GPS satellite constellation. The 17 core states, always present in the pose estimator, estimate attitude, position, velocity, bias, and time. Attitude is stored as the $ZY'X''$ Euler angles yaw, pitch, and roll required to orient the local geocentric East-North-Up (ENU) coordinate frame with respect to a coordinate frame aligned with IMU axes. Position is stored as the three-element vector position of the IMU in an Earth-Centered, Earth-Fixed (ECEF) coordinate frame. Velocity is stored as the three-element vector velocity of the IMU in the ECEF frame. Bias stores the six-element vector of slowly-drifting rate gyroscope and accelerometer biases; separate biases are stored for each axis of the rate gyro and each axis of the accelerometer. Time stores the PolaRx2e@ GPS clock bias and clock drift rate.

The pose estimator's adjunct states estimate quantities related to individual and paired GPS satellites. Two types of adjunct states are estimated: residual satellite biases, and carrier phase biases. Residual satellite biases (RSBs) estimate time correlated biases in the calculated line-of-sight distance from Skynet's GPS antennas to each visible GPS satellite. One bias is estimated for each visible satellite. Carrier phase biases (CPBs) estimate the unknown but constant carrier phase ambiguity obtained when carrier phase measurements from two satellites are double differenced across two PolaRx2e@ antennas [71]. In Skynet's pose estimator, one satellite is selected as the reference satellite against which all double differences are made. Two double differences are created for each remaining satellite: one using the PolaRx2e@'s main and first auxiliary antennas, and one using the PolaRx2e@'s main and second auxiliary antennas. If the same set of N satellites is tracked across all three PolaRx2e@ antennas, this scheme of

RSBs and CPBs results in a total of $N + 2\,(N - 1)$ adjunct states estimated in the pose estimator. As old satellites drop below the horizon and new satellites appear, the number of adjunct states changes to accommodate.

The prediction step of Skynet's ESRIF pose estimator performs a discrete numerical integration using one IMU measurement. The method used is a dual-rate integration method similar to that proposed by Savage [78], [79], and [80]. In Skynet's pose estimator, the faster of the dual-rate integrations is performed on the IMU itself, which reports integrated incremental changes in orientation and velocity at 400 Hz. The slower of the dual-rate integrations is performed in software. At a high level, the incremental changes in orientation and velocity measured by the IMU are first corrected by the biases, similar to the model discussed in Ohlmeyer [68]. After correction, the corrected measurements are used to integrate the state of the pose estimator. During integration, additional corrections are made for incremental rotation of the Earth, and for apparent centripetal and coriolis accelerations arising from the rotating ECEF coordinate frame. Gravity, as measured by the accelerometers, is also subtracted out using the EGM-96 Earth gravity potential model with a numerically stable formulation of Legendre polynomial recursions [44], [46]. Other elements of the pose estimator state vector: rate gyro and accelerometer biases, clock drift rate, and all CPBs, are modeled as random walk processes. The RSBs are modeled as autocorrelated random processes with characteristic correlation times of 600 sec. to approximate the time correlation effects of biases introduced by multipath [11].

Measurement updates in Skynet's ESRIF pose estimator are performed asynchronously; that is, when they become available from the sensors. All localiza-

tion sensors except the IMU, which is used solely for predictions, are used to update the pose estimate. A scalar speed measurement is generated from the average of the ABS wheel encoders on the rear wheels. This measurement is compared to the velocity magnitude calculated from the current state estimate $\hat{x}(k)$ to generate an innovation for the linearization described in equation 2.13. The Ag252 differential correction measurements are similar to the ABS wheel speeds: the position of the Ag252 antenna as measured by the differential correction signal is compared to the position of the Ag252 antenna as predicted by the current pose solution to generate an innovation.

Because the PolaRx2e@ provides raw GPS measurements from three antennas simultaneously, its measurement updates provide richer information than the wheel encoders or the Ag252. Rather than extract an innovation from the GPS solution reported by the PolaRx2e@ directly, individual innovations are instead extracted for all pseudoranges, Doppler shifts, and carrier phase double differences measured on the PolaRx2e@'s three antennas. The result is a large set of PolaRx2e@ measurements: one pseudorange for each tracked satellite on each antenna tracking it, one Doppler shift for each tracked satellite on each antenna tracking it, and two double differenced carrier phases for each satellite except the reference satellite across the PolaRx2e@ antenna pairs. Thus, for the same N satellites tracked by each PolaRx2e@ antenna, a total of $3N + 3N + 2(N - 1)$ measurements are formed at each PolaRx2e@ update. For each of these measurements, the ESRIF implements the full GPS software interface specified in the GPS interface control document [32]. In other words, the ESRIF computes pseudorange, Doppler shift, and carrier phase innovations by comparing raw GPS measurements made at the PolaRx2e@ with the expected raw measurements implied by the current pose solution. The broadcast GPS satellite orbit and sig-

nal propagation models are also augmented with WAAS signals, as specified in the WAAS interface control document [95]. The signal propagation models are further refined with tropospheric corrections based on current atmospheric conditions [65], [75], [18], [13].

Utilization of the raw GPS measurements rather than the filtered PolaRx2e@ GPS solution within the ESRIF creates a *tightly-coupled* pose estimator. Such a configuration is particularly suited to urban environments, as it allows GPS information to be incorporated as it is acquired, even when fewer than four satellites are visible [6]. In addition, direct utilization of raw GPS observables permits errors to be modeled on a satellite by satellite basis [81]. Skynet's pose estimator takes advantage of this fact to perform statistical hypothesis tests to validate each measurement against the current pose estimate before updating the ESRIF. These tests are more powerful than those that could be done on the GPS receiver itself, as they incorporate historical information from Skynet's other three localization sensors. Similar hypothesis tests are performed on ABS wheel encoder and Ag252 differential correction measurements to ensure robustness and consistency in the resulting pose solution.

2.4 Sensitivity Analysis In The Design Of The Pose Estimator

Cornell's pose estimator algorithm was field-proven in the 2007 DARPA Urban Challenge, where it served as the positioning signal Skynet used to complete 60 miles of autonomous urban driving [56]. This success validates the design decisions described in section 2.3 as a whole, but gives little insight into the effects of each individual decision on the performance of the final algorithm.

In order to gain that insight, the pose estimator must be deconstructed. Each critical design decision is therefore systematically reversed, to determine the pose estimator's sensitivity to its design in a practical setting.

While the Urban Challenge Event (UCE) provided an excellent system level endurance test for Skynet, its sheer magnitude makes it unwieldy for individual unit tests against the pose estimator. In particular, the UCE's 6-hour duration makes it computationally prohibitive to use for algorithm evaluation in its entirety, and it is unclear which, if any, portion of the UCE might be better or worse to use for an evaluation. More importantly, exogenous environmental stimuli in the UCE frequently resulted in unimaginably complex interactions between the competing robots. What appeared *prima facie* to be algorithmic faults were often mere manifestations of differing interpretations of the UCE rules, and are left to other papers [56, 26].

Rather than contend with such complex and unpredictable interactions, this study uses an excerpt of data taken from Skynet's logs of the DARPA Urban Challenge National Qualifying Event (NQE). Unlike the UCE, which consisted of 6 hours of largely uninterrupted autonomous driving, the NQE was a set of 3 short courses designed to test individual components of urban driving behavior. NQE Area C, shown from an overhead view in Figure 2.3 and from a bird's eye view in Figure 2.4, provides the ideal setting for unit tests against the pose estimator. The 20 minute course is compact, requiring robots to loop back over its three city blocks and two intersections several times during the prescribed mission. The course also contains features that often challenge satellite navigation: trees, low buildings, and power lines, but these features are not so dense that GPS is unavailable. Finally, Skynet completed NQE Area C flawlessly, yielding

Figure 2.3: An overhead view of the DARPA Urban Challenge NQE Area C course. Skynet's ground track, reconstructed from data logs taken at the Urban Challenge, is plotted in blue. Rectification and registration errors have not been entirely removed from the image.

a log of synchronized raw sensor data, processed pose data, and video records for use as an externally-validated ground truth solution.

In the sections that follow, the design decisions comprising Skynet's pose estimator are systematically reversed. The pose estimator's design elements are removed sequentially and cumulatively; each section removes one *additional* design element from the remainder of the pose estimator. This approach is taken

Figure 2.4: A bird's eye view of the DARPA Urban Challenge NQE Area C course. The 20 minute compact course includes tree cover, low buildings, and power lines, making it ideal for unit tests against the pose estimator. Rectification and registration errors have not been entirely removed from the image.

primarily for the sake of brevity, to avoid an exponential walk through the pose estimator's design space. Design elements are removed in the order the authors believe most insightful: the selection of each feature to remove is made in an attempt to magnify the differences between successive variations of the pose estimator algorithm.

To statistically evaluate the variants of the pose estimator algorithm gener-

ated below, each pose estimator variant is run on the NQE Area C data log. The output of each pose estimator variant is then compared to the baseline output, recorded on the day Skynet completed NQE Area C. For this comparison, the baseline pose estimator output is treated as a field-verified ground truth, where the reported estimation mean $\hat{x}(k) = R_{xx}^{-1}(k)\,\hat{y}(k)$ and covariance $P(k) = R_{xx}^{-1}(k)\,R_{xx}^{-T}(k)$ describe the probability density function of Skynet's true pose:

$$x(k) \sim \mathcal{N}(\hat{x}(k), P(k)) \tag{2.14}$$

In other words, Skynet's true pose $x(k)$ at time index k is henceforth taken as a Gaussian random variable with mean $\hat{x}(k)$ and covariance $P(k)$ as calculated when Skynet completed NQE Area C on October 29, 2007. This baseline data is then used to cast the sensitivity analysis as a statistical hypothesis test, by questioning whether each variant of the pose estimator algorithm produces output that is likely to have been drawn from this true pose probability density. If so, the variant will have produced a solution that is statistically indistinguishable from the logged truth data, revealing the algorithm's insensitivity to the changes made in that variant.

The output of each pose estimator variant $x^{\ddagger}(k)$ is first tested to see whether it is likely to have been drawn from the true pose density given in equation 2.14. This comparison is made by noting that if $x^{\ddagger}(k) \sim \mathcal{N}(\hat{x}(k), P(k))$ at time index k, then the test statistic

$$\tau(k) = \left[x^{\ddagger}(k) - \hat{x}(k)\right]^{T} P^{-1}(k) \left[x^{\ddagger}(k) - \hat{x}(k)\right] \tag{2.15}$$

will be distributed as a chi-squared random variable with n degrees of freedom, where n is the length of the vectors $x^{\ddagger}(k)$ and $\hat{x}(k)$ [4]. To evaluate specifically the pose estimator's final position solution, the test is conducted only on the

in-plane East and North components of the pose estimators' position estimates. These in-plane components are generated by a common projection into an East-North plane tangent to the WGS-84 ellipsoid Earth model at a point near the center of the Urban Challenge course. The test statistic $\tau(k)$ is therefore distributed as a chi-squared random variable with two degrees of freedom.

Seven other secondary statistical quantities are also computed to aid in the comparison of pose estimator variants. To quantitatively evaluate the proximity of each variant to the baseline solution, the symmetric Kullback-Leibler (K-L) divergence is computed between each variant and the baseline. For computational feasibility across the entire data set, this divergence is computed under the approximation that the pose solution at each time step is independent of the pose solution at every other time step. To aid with physical insight, the mean position and attitude errors are also computed for each variant, along with their standard deviations. Finally, to evaluate each variant's practical utility, the mean and standard deviation of the discontinuity in the ESRIF position update are also calculated. These statistical results are summarized in Table 2.1, and are interpreted in detail in sections 2.4.1 through 2.4.6.

2.4.1 Sensitivity To Map Aiding

The sensitivity analysis of the pose estimator begins by removing its map aiding algorithm. Cornell's map aiding algorithm utilizes the DARPA-provided Route Network Definition File (RNDF) and onboard computer vision algorithms to search for surveyed landmarks such as lane lines and stop lines to aid in localization. These landmark-based onboard sensing cues are then fused with the

output of the pose estimator in a bootstrap particle filter to create a posterior estimate of Skynet's position and heading, called its 'posterior pose' [55]. This posterior pose estimate is sent to the intelligent planner at a 10 Hz rate as the preferred localization signal for planning and path tracking.

The pose estimator is thus run on the Area C log data with map aiding removed. Hypothesis testing this variant at the one-sided 5% significance level (the 95% confidence level) reveals statistically significant pose estimates in only 532 of the 12020 iterations in the NQE Area C log data. The average difference between the two pose variants across the entire data set is 8.7 cm, with a standard deviation of 4.9 cm. Although the two variants produce slightly different pose solutions, it is evident that the pose estimator is relatively insensitive to map aiding. This result is consistent with the findings of Miller and Campbell, who show small (but statistically significant) improvements with map aiding in a full GPS environment, and large improvements with map aiding in GPS blackouts [55].

Although the pose estimator is relatively insensitive to map aiding augmentations in a full GPS environment, it is instructive to examine Skynet's position when map aiding produces a significantly different pose solution. These locations are plotted in Figure 2.5. Most significant differences occur during shallow turns, or immediately after sharp turns. Some of these differences are likely due to disagreement between the surveyed waypoints defining the course and the painted road lines. Differences occurring immediately after sharp turns most likely represent the information gained when Skynet acquires new lane lines perpendicular to its former direction of travel. In these cases, the newly-acquired lane lines provide information along a new axis, allowing further re-

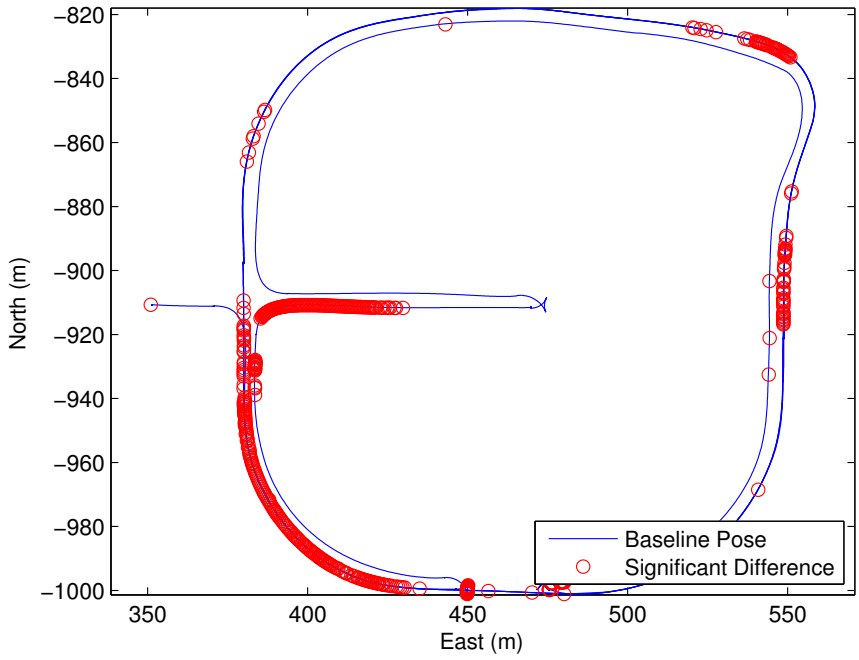

Figure 2.5: Skynet's position when the baseline pose estimator with map aiding produces a statistically different pose estimate than the pose estimator without. Significant differences tend to occur after sharp turns, where map aiding provides positioning cues along new directions.

finement to Skynet's pose estimate.

2.4.2 Sensitivity To Differential Corrections

With map aiding removed from the pose estimator, the next design element to be removed is the differential correction signal. As mentioned in section 2.2.2, the high precision (HP) and virtual base station (VBS) differential correction signal decoded from the Ag252 are the primary source of sub-meter absolute positioning information available to Skynet's pose estimator. With map aiding already removed in section 2.4.1, the resulting pose estimator variant relies on

raw GPS observables and WAAS corrections as its remaining sources of absolute positioning information.

Without its primary source of sub-meter absolute positioning information, the pose estimator suffers considerably. Not surprisingly, this pose estimator variant yields statistically significant pose estimates at the 5% significance level in all 12020 time steps contained within the NQE Area C data log. As shown in Table 2.1, these differences yield a substantial factor of 10 increase in the K-L divergence between this pose estimator variant and the baseline recorded at the NQE. In physical terms, this corresponds to a mean difference of 1.19 m between this variant and the baseline, with a standard deviation of 0.33 m. Figure 2.6 plots the ground track of the variant against the baseline, showing the estimation errors committed by the variant throughout the log data.

Although the pose estimator's absolute accuracy suffers greatly when denied its most important source of sub-meter positioning information, its precision remains intact. Figure 2.7 quantifies the algorithm's precision with a plot of magnitudes of the discontinuities in the pose estimator's updates when it is denied differential corrections. In this variant of the pose estimator, these updates have a mean discontinuity of 1.5 cm with a standard deviation of 3.3 cm. The fact that these discontinuities are small is not guaranteed by the EKF or the ESRIF, which place no constraints on the continuity of the pose solution during measurement updates. Rather, it is the result of a deliberate effort to design the system for robustness even before accuracy, as robustness is most essential for autonomous navigation. In particular, large discontinuous jumps in the pose solution translate directly into large discontinuous steps in path tracking error. At worst, an aggressive path tracking controller will respond to such errors

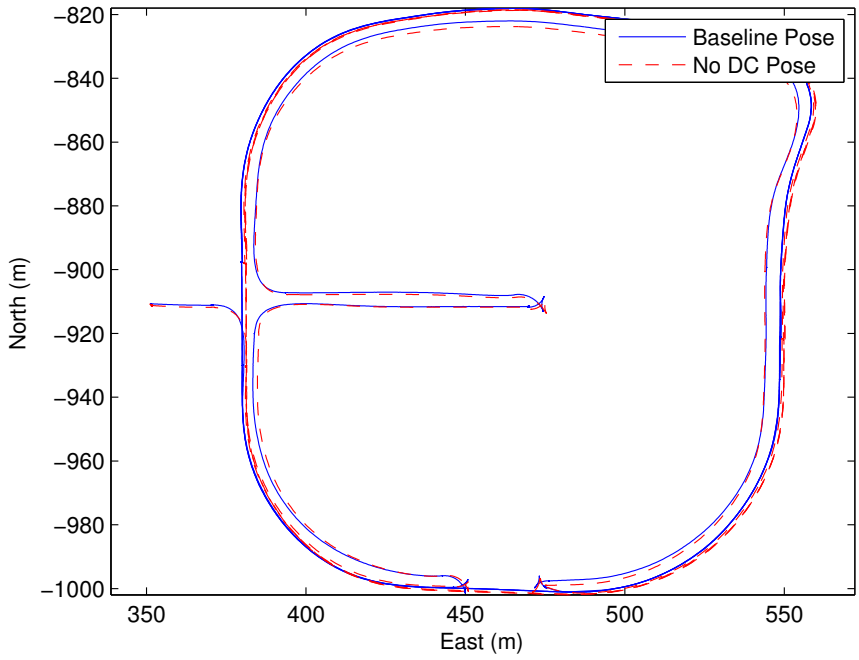

Figure 2.6: When denied differential corrections, the pose estimator produces a ground track with a mean error of 1.19 m. Although the mean error is approximately one vehicle width, the pose solution has no sudden discontinuities that would be devastating to autonomous driving.

with unsafe and possibly catastrophic swerving. At best, the path tracking controller's bandwidth must be lowered considerably if these discontinuous errors cannot be removed from the positioning algorithm itself.

In addition to precision, the pose estimator's attitude solution retains its accuracy even without differential corrections. In particular, the pose estimator variant's heading (yaw) solution, used for Skynet's heading measurement, differs from the baseline solution by an average of only $-0.19°$ with a standard deviation of $3.27°$. It is thus apparent that Skynet's pose estimator receives very little attitude information from differential corrections, despite the fact that they are its most important source of absolute positioning information.

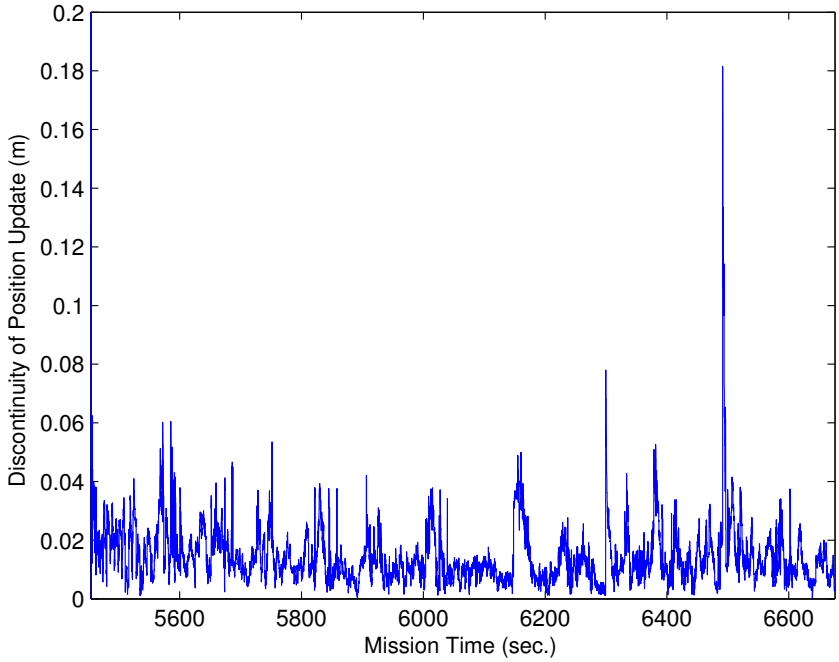

Figure 2.7: Although denied differential corrections (DC), the magnitude of the pose estimator's updates are still very small. The position solution remains precise and robust, two features critical for autonomous driving.

2.4.3 Sensitivity To Filter Integrity Monitoring

After differential corrections, the next design element removed from the pose estimator is its suite of filter integrity monitoring (FIM) hypothesis tests. In general Bayesian estimation, these FIM tests are meant to detect when the filter has become *inconsistent*; that is, when one or more of the linear, Gaussian, or correlation assumptions of the EKF or ESRIF have been violated [11]. In the context of pose estimation for autonomous navigation, however, such *ex post facto* FIM tests are not helpful. For the same reasons discussed in section 2.4.2, it is far better to maintain filter consistency by identifying and rejecting faulty measurements before they corrupt the fused information in the pose es-

timator [88]. Skynet's pose estimator therefore performs two online FIM tests. The first test, meant to detect (but not identify) faulty measurements, is based on a global chi-squared statistic formed over the entirety of each measurement vector. If the measurement is found to be faulty, a second test is performed to attempt to identify offending elements of the measurement vector. The second test is similar to the cycle slip detection technique used by Mohiuddin and Psiaki, which tests each satellite for faults individually by evaluating the integrity of all measurements *except* those from the satellite in question [60]. Skynet's pose estimator performs a substantially cheaper variant of this test by pairing each satellite with the most consistent satellite in an attempt to identify and reject carrier phase cycle slips and large multipath distortion without discarding the entire measurement vector.

With the FIM tests turned off, the pose estimator blindly accepts all measurements without question. Since the filter still has no map aiding or differential corrections, it fails the 5% significance test in 11856 of 12020 time steps. Interestingly, these results suggest that this variant of the pose estimator is actually statistically *closer* to the baseline pose estimator than the differential corrections variant tested in section 2.4.2. This result is further corroborated with K-L divergences in Table 2.1, and with the position output itself, which differs from the baseline algorithm by an average of 83.1 cm with a standard deviation of 52.2 cm.

Unfortunately, the slight improvement to accuracy comes at the price of robustness. Aside from the larger standard deviation between this pose variant and the baseline, it also has average update discontinuities of 2.2 cm with a standard deviation of 4.5 cm. More importantly, the variant is less stable, with

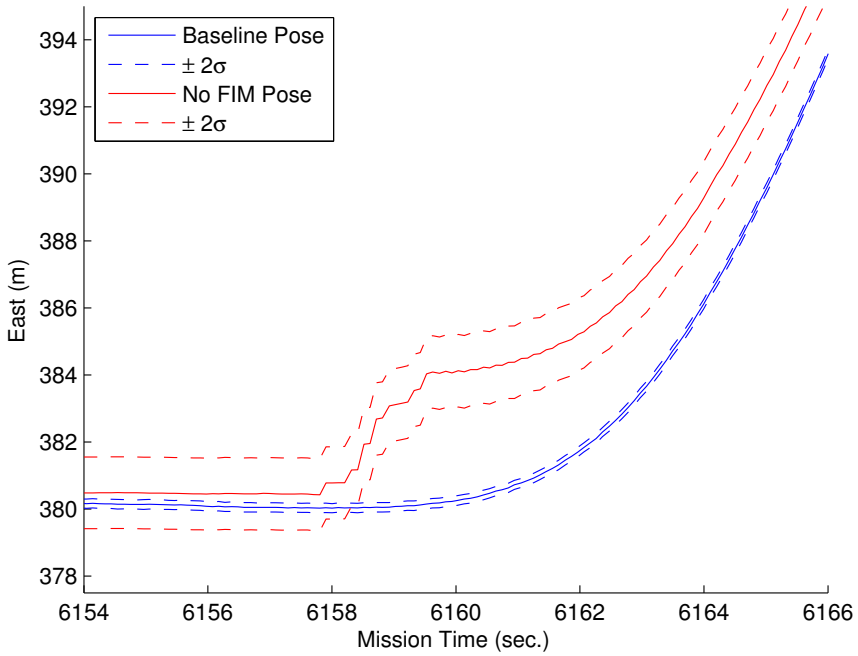

Figure 2.8: When filter integrity monitoring (FIM) is turned off, the pose estimator suffers from large discontinuities.

11 separate instances of discontinuities exceeding 20 cm and one exceeding 70 cm. These discontinuities are most commonly due to the appearance of new GPS satellites at low elevations, whose measurements tend to suffer more from ionospheric, tropospheric, and multipath distortion. Paradoxically, these error-prone satellite measurements also tend to offer position information along new axes, thereby affecting the pose estimate more than measurements from old satellites. A time history of the largest of these discontinuities is shown in Figure 2.8. If it were used as a feedback signal, the path tracking controller would likely make an unsafe maneuver.

2.4.4 Sensitivity To WAAS Corrections

The fourth design element removed from the pose estimator is the WAAS signal. The WAAS signal nominally includes improved positioning, pseudorange, and ionospheric corrections for each individual satellite [95]. More importantly for Skynet's pose estimator, the WAAS corrections include uncertainty estimates and models of their degradation over time. These uncertainty parameters allow the satellites to be validated more accurately within the pose estimator, ensuring each satellite's information is incorporated with a fair weighting.

When WAAS corrections are removed, the pose estimator defaults to a very conservative weighting of satellite information. Satellites near the horizon are assumed to have large variances, and thus contribute almost no information to the final pose solution. Even without differential corrections or integrity monitoring, this conservative approach produces some useful results. At the 5% significance level, this variant is statistically different from the baseline pose estimator in 10741 of 12020 time steps, and Table 2.1 indicates that this conservative approach is actually closer to the baseline pose estimator than many other variants. Intuitively, the result makes sense: by adopting a skeptical view of the information coming from low-elevation satellites, this variant is on average less sensitive to the most common signal errors that caused difficulties for the no-FIM variant examined in section 2.4.3. This variant has an average error of only 48.5 cm with a standard deviation of 37.3 cm compared to the baseline.

This result should not be taken as evidence against the usefulness of WAAS. Although this variant is statistically closer to the baseline truth on average, it still suffers from the same acute problems as the FIM variant tested in section 2.4.3. The discontinuities in this variant's ESRIF updates have an average of 2.2

cm with a standard deviation of 5.5 cm, indicating that this variant experiences more extreme discontinuities than the one using WAAS corrections. The result is corroborated by examining the individual update discontinuities: there are 23 separate instances of discontinuities exceeding 20 cm, and there are 2 instances where the variant differs from the baseline solution by more than 3 m. These results suggest that the pose estimator's accuracy is relatively unaffected by WAAS corrections, though WAAS corrections do aid in the algorithm's overall robustness.

To make the analysis more physical, this variant can be compared statistically with the no-FIM variant examined in section 2.4.3. Hypothesis tests at the 5% significance level indicate that this variant differs from the no-FIM variant in section 2.4.3 in only 1089 of 12020 time steps, and the symmetric K-L divergence between the variants is only 25672.8. Comparing this value to other much larger divergence values in Table 2.1 supports the previous conclusion: WAAS primarily aids in the robustness of the pose estimator, not in its absolute accuracy.

2.4.5 Sensitivity To GPS Carrier Phases

The final external satellite measurement to be removed from the pose estimator are measured carrier phases of GPS satellites. These carrier phases are used in two places in Skynet's pose estimator: as measurements of satellite Doppler shifts with respect to the receiver, and for double differencing signals between Skynet's antennas. These carrier phase signals are an important source of attitude information. Intuitively, each Doppler shift measurement yields velocity

information along the line of sight between Skynet and the satellite to which the measurement corresponds. Similarly, each carrier phase double difference yields relative positioning information of one of Skynet's antennas with respect to another, a measurement made extremely accurate due to the noise canceling properties of the double difference operator applied to antennas in close proximity.

The pose estimator's performance is further degraded from the no-WAAS variant examined in section 2.4.4 when both these measurements are removed. At the 5% significance level, this variant produces significantly different output from the baseline in 11847 of 12020 time steps. In addition to a larger K-L divergence, shown in Table 2.1, algorithmic errors are larger, averaging at 69.1 cm with a standard deviation of 26.8 cm. Interestingly, the variant achieves surprisingly low attitude errors, averaging only -1.1° with a standard deviation of 0.5°.

Although the pose estimator still retains the conservative assumptions discussed in the no-WAAS variant examined in section 2.4.4, it is evident that the benefits of incorporating carrier phase measurements trade off heavily with algorithm performance. In particular, since these measurements are often modeled as being much more accurate than pseudoranges, the pose estimator is very sensitive to common non-Gaussian errors such as cycle slips. In section 2.4.4, this resulted in a number of large discontinuities in the pose solution. In the carrier phase variant discussed here, those measurements are ignored entirely, resulting in a maximum discontinuity of less than 5 cm over the entire data set.

Figure 2.9 compares baseline errors between the no-carrier-phases pose variant and the variant including carrier phases studied in section 2.4.4. On average

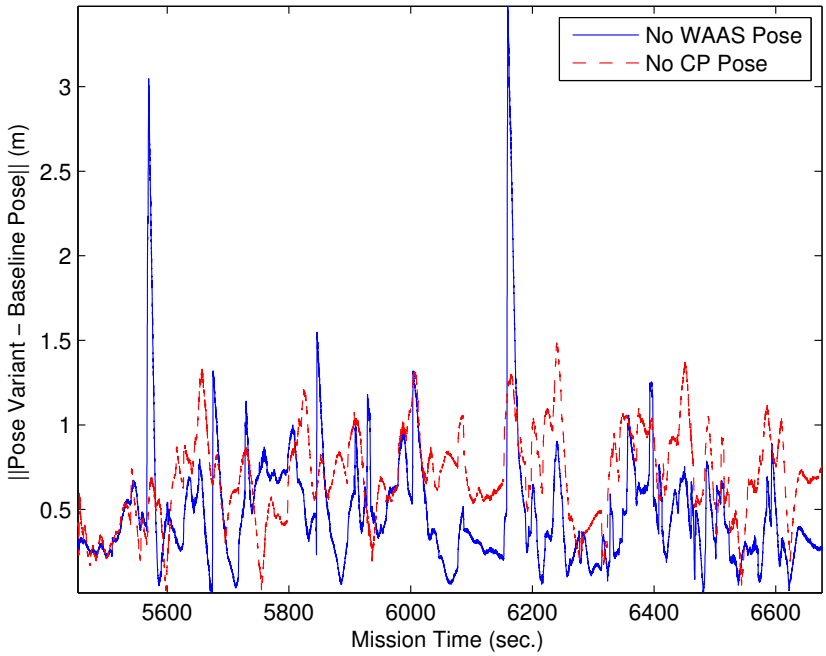

Figure 2.9: Incorporating Doppler shifts and carrier phases (CP) without online validation yields a less robust pose solution than ignoring those measurements entirely.

the two variants are quite similar: at the 5% significance level they differ from each other in only 159 of 12020 time steps, and their symmetric K-L divergence is only 17551.5. Yet despite the similarities, the conservative pose variant using only pseudoranges is more precise and robust, making it more suitable for autonomous navigation in the absence of statistical measurement validation. In light of the slow drift rates in Skynet's tactical grade IMU, the price paid in degraded attitude accuracy is outweighed by the more conservative filter's precision and robustness, assuming statistical measurement validation is not performed. If measurement validation is to be performed, however, variants including carrier phase measurements are preferred, since biases in vehicle yaw translate directly into steady state errors in path tracking.

2.4.6 Sensitivity To Signal Blackouts

Sections 2.4.1 through 2.4.5 have deconstructed the pose estimator, removing all sources of absolute positioning information except the most basic treatment of GPS pseudoranges themselves. In essence, these sections have considered the ramifications of pose estimator design decisions on autonomous robotics in an ideal GPS signal environment. All satellite signals have been treated as readily available, leaving the user to decide which information to use and which to ignore. Such an experimental assumption disregards two events with significant impact on the practical deployment of autonomous field robots: complete signal blackouts, and (less intuitively) signal reacquisitions.

From a strict Bayesian perspective, the EKF and ESRIF naturally handle blackouts and reacquisitions: the pose solution is predicted when no new measurements are available, and it is updated when they are. Yet this statistically rigorous solution carries the same practical risks first discussed in section 2.4.2. Not only does the pose solution degrade over time as numerical integration errors mount, but signal reacquisition after a long measurement drought tends to produce and even magnify discontinuous jumps in the pose solution. The authors addressed these issues as part of Cornell's 2005 DARPA Grand Challenge entry [58], and more advanced solutions are not pursued here.

In keeping with the sensitivity analysis conducted so far, the baseline pose estimator is instead examined in a statistical sense in two types of signal blackouts: loss of differential corrections, and full loss of signal. Both of these events are tested by denying the baseline pose estimator access to the appropriate signals in the data log, as well as external map aiding information. Aside from these signal omissions, other design elements are retained: FIM, WAAS, and

carrier phases. The mission time of the signal loss will be the same in each experiment: at 5539.0 sec. Skynet remains stationary in the data log up until this time, giving the pose estimator approximately 86 seconds of full signal visibility for initialization purposes. Once dropped, the signals remain unavailable to the end of the data log, approximately 19 minutes later.

The first experiment drops only differential corrections. This variant's K-L divergence in Table 2.1 reveals a surprising proximity to the baseline solution with full information. The result suggests that full and continuous coverage from a differential corrections signal may not be necessary, due to the temporal 'memory' of the ESRIF and the quality of Skynet's IMU. Figure 2.10 quantifies this claim by plotting the minimum and maximum 1σ position uncertainties, $\sigma_{\min}(k)$ and $\sigma_{\max}(k)$, as defined by the pose estimator's covariance matrix. These values are obtained via singular value decomposition of the pose estimator's covariance matrix:

$$P(k) = U(k) \cdot \begin{bmatrix} \sigma^2_{max}(k) & 0 \\ 0 & \sigma^2_{min}(k) \end{bmatrix} \cdot V^T(k) \tag{2.16}$$

where $U(k)$ and $V^T(k)$ are unitary matrices. Intuitively, $\sigma_{\min}(k)$ and $\sigma_{\max}(k)$ correspond to 1σ error bounds (in meters) in the most and least uncertain directions of the pose solution. Although the pose solution degrades rapidly when differential corrections are first lost, the rate of degradation slows over time. Eventually, the pose solution would degrade to the no-differential-corrections variant analyzed in section 2.4.2. More importantly, if sub-meter accuracy is a design goal as it was in the Urban Challenge, then these results indicate that the pose estimator can tolerate long stretches of time without differential corrections.

The second experiment drops all external satellite signals. Two variants of

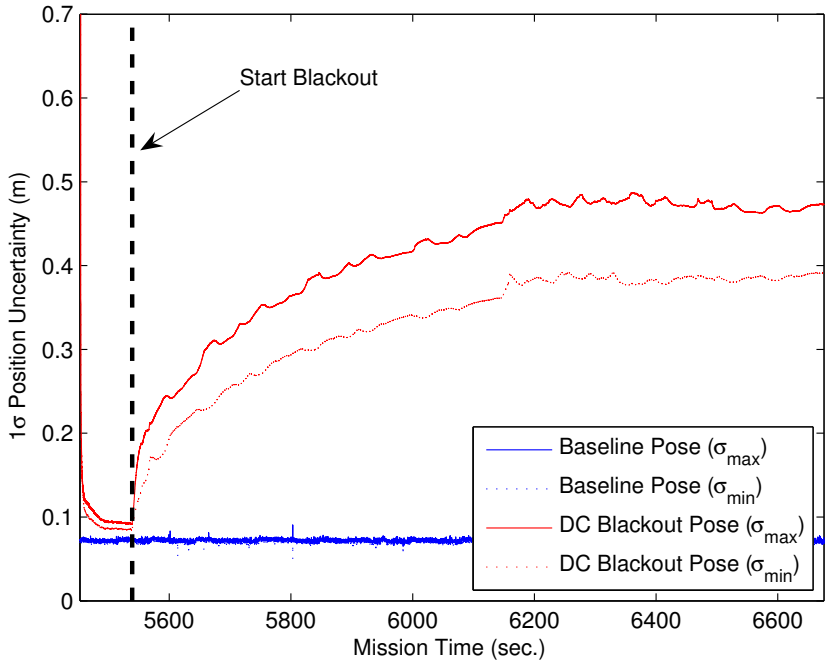

Figure 2.10: Square roots of singular values of the covariance matrices for the baseline pose estimator and one in which differential corrections have been blacked out at mission time 5539.0. Although the pose solution degrades rapidly when differential corrections are first lost, the degradation tapers to a slow pace.

the pose estimator are run in the total signal blackout: one just integrating the IMU, and one updating IMU integrations with wheel speed measurements. The experiment results in significantly degraded pose solutions: with no absolute position information available after the signal blackout, the effects of steadily-growing numerical integration errors quickly overwhelm information obtained when the external navigation signals were present. Figure 2.11 plots the Euclidean distances between the two pose estimator variants and the baseline pose solution, to quantify the rate at which information leaves the estimator during the blackout. The rate is rapid: both variants reach 1 m error levels approximately 25 seconds after the beginning of the blackout.

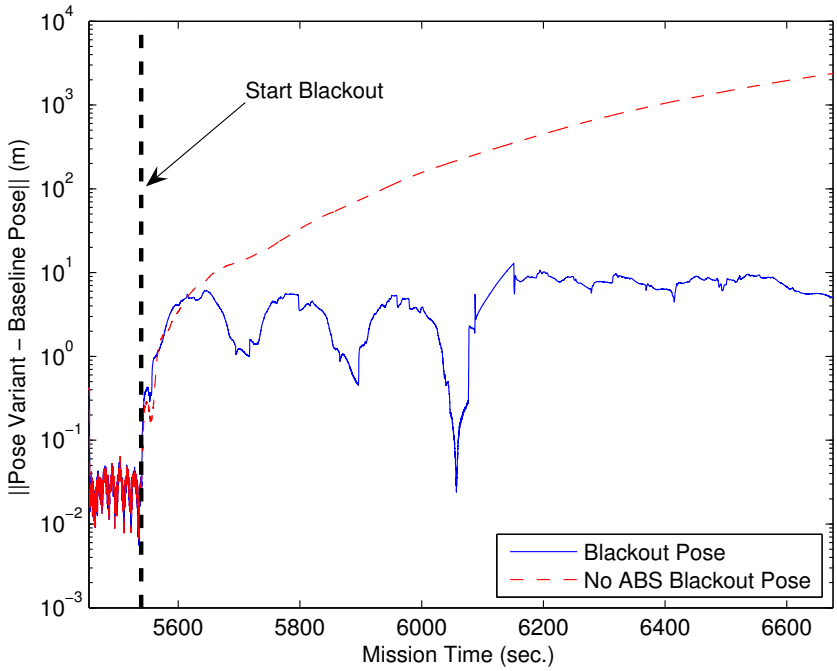

Figure 2.11: Euclidean error from baseline for two pose variants in a GPS blackout: one using ABS wheel speed measurements (Blackout Pose) and one not using ABS wheel speeds (No ABS Blackout Pose). All external satellite positioning signals have been blacked out at mission time 5539.0.

The K-L divergences in Table 2.1 highlight the importance of using wheel speed sensors to slow the rate of growth of numerical integration errors. Although both pose estimator variants experience similar errors for the first 25 seconds of signal blackout, the variant utilizing wheel speeds experiences final errors less than two orders of magnitude lower than the variant without wheel speeds. This result is due to the ESRIF's capability to aggregate information; it learns correlation models between state estimation errors when external navigation signals are present. When the signals are blacked out, the ESRIF uses the correlation model to update its entire state estimate from wheel speed measurements. This allows the variant utilizing wheel speeds to occasionally decrease its position uncertainty, as shown in Figure 2.12. In contrast, the variant

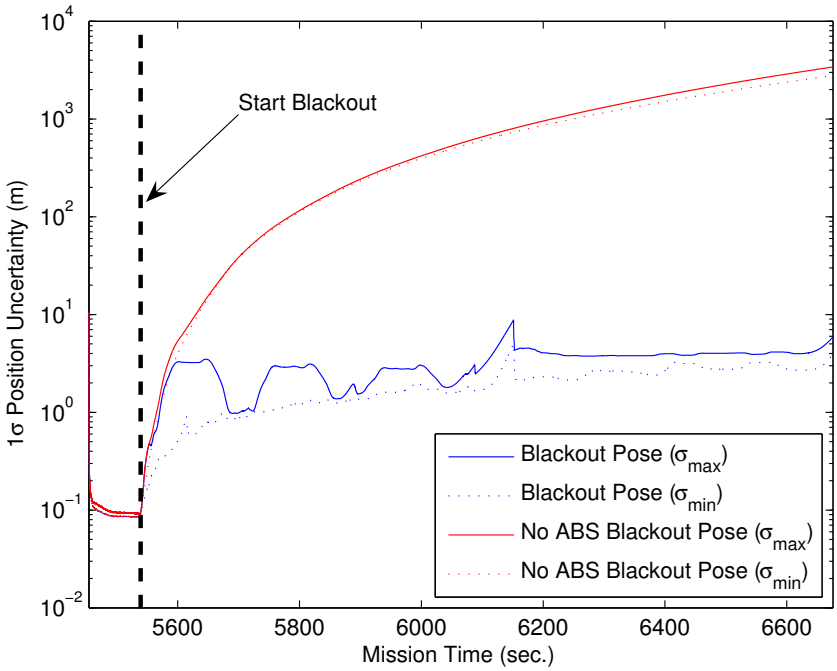

Figure 2.12: Square roots of singular values of the covariance matrices for pose estimators with and without wheel speed measurements. All external satellite positioning signals have been blacked out at mission time 5539.0.

ignoring wheel speeds makes no measurement updates, so its estimated errors grow much faster. Strictly speaking, both variants are unusable for autonomous navigation due to their large positioning errors and discontinuities. Nevertheless, the relatively slow rate of error growth in the pose variant utilizing wheel speeds makes it a promising candidate for applying map aiding and rate limiting techniques to improve its accuracy and precision [55, 58].

2.5 Conclusion

This paper has performed an in-depth sensitivity analysis of the tightly-coupled pose estimator built as part of Cornell University's 2007 DARPA Urban Challenge robot, Skynet. Skynet's pose estimator fuses information from a Litton LN-200 IMU, Skynet's ABS wheel encoders, a Septentrio PolaRx2e@ three-antenna GPS receiver, and a Trimble Ag252 differential corrections GPS receiver. Information is fused in an Extended Square Root Information Filter (ESRIF), a numerically robust implementation of an Extended Kalman Filter (EKF). The pose estimator also fuses map-based information from onboard vision sensors detecting pre-surveyed lane lines and stop lines, though these are incorporated separately via particle filter [55].

The paper has investigated the sensitivity of Skynet's pose estimator to each of its inputs by deconstructing the pose estimator: analyzing the effects of reversing each of its critical design components. These pose estimator variants have been tested on logged data taken at the DARPA Urban Challenge National Qualifying Event, with the logged output of the pose estimator used as a baseline solution. A summary of the sensitivity analysis is presented in Table 2.1, which lists for each pose estimator variant the percent of iterations at which the variant is significantly different from the baseline (% Diff.), the symmetric Kullback-Leibler divergence between the variant and the baseline (K-L), the mean and standard deviation of the positioning error between the variant and the baseline (Avg. Pos. and Std. Pos.), the mean and standard deviation of the heading error between the variant and the baseline (Avg. Att. and Std. Att.), and the mean and standard deviation of the ESRIF update discontinuities in each variant (Avg. Discont. and Std. Discont.), where all variants have been

compared to the baseline pose solution recorded when Skynet competed in the DARPA Urban Challenge. Table 2.1 indicates that the pose estimator is most sensitive to its differential corrections, and it becomes increasingly less effective as integrity monitoring and other design decisions are reversed as well.

Although the pose estimator is statistically most sensitive to differential corrections, it is important to remember that precision and robustness are more important than absolute accuracy in autonomous urban navigation. With that in mind, many tested pose estimator variants yielded intolerably large discontinuities in their pose solutions. In fact, only variants including filter integrity monitoring were found suitable in this respect, aside from one conservative variant using only pseudoranges.

Similar results were found in variants experiencing signal blackouts. A 19 minute differential corrections blackout revealed that although the pose estimator is rather sensitive to differential corrections, it does not need to receive them continuously. This result makes 60 mile autonomous drives like the DARPA Urban Challenge possible, as differential corrections are often only received intermittently. Two pose estimator variants were further tested in a 19 minute total signal blackout: one variant incorporating Skynet's wheel speed measurements, and one only integrating the IMU. Both variants were found unsuitable for autonomous navigation, though related research has suggested that the variant incorporating wheel speeds is potentially robust enough to be repaired with map aiding techniques [55].

Table 2.1: Summary of Statistical Similarity Between Pose Estimator Variants

Pose Variant	**% Diff.**	**K-L**	**Avg. Pos.**	**Std. Pos.**
No Map Aiding	4.4	278780.4	0.087 m	0.049 m
No Differential Corrections	100.0	3389571.1	1.188 m	0.326 m
No Integrity Monitoring	98.6	2711135.1	0.831 m	0.522 m
No WAAS	89.4	2063970.1	0.485 m	0.373 m
No Carrier Phases	98.6	2399262.1	0.691 m	0.268 m
Differential Corrections Blackout	85.9	1093691.6	0.535 m	0.391 m
Full Blackout	91.6	59060232.5	4.977 m	3.095 m
Full Blackout, No Speeds	92.6	4.7×10^{12}	557.625 m	674.091 m
Pose Variant	**Avg. Att.**	**Std. Att.**	**Avg. Discont.**	**Std. Discont.**
No Map Aiding	-0.21°	3.27°	0.0046 m	0.0318 m
No Differential Corrections	-0.19°	3.27°	0.0149 m	0.0331 m
No Integrity Monitoring	-0.17°	3.27°	0.0223 m	0.0446 m
No WAAS	-0.14°	3.26°	0.0220 m	0.0553 m
No Carrier Phases	-1.08°	0.50°	0.0095 m	0.0391 m
Differential Corrections Blackout	-0.19°	3.27°	0.0125 m	0.0329 m
Full Blackout	-1.87°	0.45°	N/A	N/A
Full Blackout, No Speeds	-1.91°	0.41°	N/A	N/A

CHAPTER 3

PARTICLE FILTERING FOR MAP-AIDED LOCALIZATION IN SPARSE GPS ENVIRONMENTS

3.1 Introduction

As the military and civilian driving experience moves toward remote and even autonomous operation, the need for extremely accurate vehicle localization in urban environments grows ever larger. The Global Positioning System (GPS) and its variants have largely met that need for areas with a clear view of the sky, but signal blockages and reflections due to buildings and trees continue to cause problems in the deepest urban canyons. At best, the GPS signal is not available, so navigation solutions rely on dead reckoning in an inertial navigation system (INS) to maintain a position solution which diverges after a few minutes. At worst, the GPS solution is available but corrupted by multipath, leading to gross errors in the overall navigation solution. In either case, the solution is often not accurate enough to localize a vehicle precisely within a lane.

Emerging research in the area of Map-Aided Localization addresses the urban localization problem by incorporating information from a known map of environmental features such as roads and lanes. Cui and Ge combine raw GPS pseudoranges with a known map of roads in a Kalman filter to improve vehicle positioning under the assumption that the vehicle is constrained to move along the roads in the proper direction [17]. The technique is quite successful, though the authors resort to a complex Interacting Multiple-Model Kalman Filter (IMMKF) to determine the vehicle's path as it approaches branches or intersections. Lee *et al.* implement a localization system under the same road con-

straints, but they utilize a Rao-Blackwellized particle filter to incorporate landmarks detected via laser rangefinder to aid in positioning [43]. This technique avoids using the IMMKF by sampling the vehicle navigation solution with a particle filter and assigning higher weights to particles that are closer to the known road map. The algorithm is extended in Ref. [99] with a Voronoi-based technique to associate each particle with a segment of the road map, but still assumes that the vehicle is constrained to the road. A third approach, that of Cheng and Singh, applies similar road constraints, but assumes vehicle odometry is unknown, as in a tracking scenario [16]. This approach uses particles to approximate the distribution over which road segment the target vehicle is on, which is not well-modeled in a linear-Gaussian Kalman filter. In contrast, El Najjar and Bonnifait treat the road as a noisy measurement in a Kalman filter [23]. This approach has the benefit that the vehicle is not constrained to move on the road, though the vehicle's predicted motion in the filter is influenced by the modeled accuracy of the road network.

The primary problem in the aforementioned approaches is that they assume the vehicle of interest generally moves along the road, which does not hold for autonomous systems. Autonomous systems are subject to sensor mistakes, and may very well drive off the road or on the wrong side of the road without a second thought. With that in mind, this study presents a particle filtering approach that captures the benefits of map-aided positioning while removing the assumption that the vehicle's motion is in any way influenced by the road. This study extends the authors' previous work, which showed the approach capable of sustaining a single 8 minute GPS blackout on a single limited course [55]. Here, statistical evidence is given for the algorithm's long-term stability in a variety of challenging sensor environments, including several 30 minute GPS

blackouts. Section 3.2 introduces the 2007 DARPA Urban Challenge, which provides the framework and motivation for the approach. Section 3.3 describes the particle filtering solution to the urban localization problem. Section 3.4 describes the implementation of the algorithm, including techniques that enable it to be run in real-time. Section 3.5 shows the ability of the algorithm to improve upon a fused GPS / INS system, especially in extended GPS blackouts.

3.2 Problem Description

The algorithm introduced here, called the *PosteriorPose* algorithm, addresses the difficult urban localization problem without assuming that the vehicle remains on the road. The problem is motivated by the DARPA Urban Challenge (DUC), a 60-mile set of mock supply missions to be completed by an autonomous vehicle operating in a closed urban environment [19]. In the DUC, entrants are given 24 hours to process a Route Network Definition File (RNDF), which contains surveyed information about the network of roads to be navigated. Within the RNDF, each viable lane is enumerated with a set of GPS waypoints surveyed down the lane's center. Special waypoints designating stop lines are also marked in the RNDF and correspond to painted white lines on the ground at which the vehicle must stop. Each lane may also optionally be designated with a nominal width and left and right line marking styles, such as 'double yellow,' 'broken yellow,' and 'broken white.' While the nominal lane widths are only approximate values, the line markings, when specified, correspond with physical markings on the ground. The problem is further complicated, however, by the possibility that markings not designated in the RNDF, such as unsurveyed lane lines, crosswalks, or turn arrows, may also be present.

After the RNDF is processed, each Urban Challenge vehicle is given a Mission Data File (MDF) indicating an ordering to a subset of the waypoints the vehicle must visit to complete its supply missions [19]. Once the vehicle is given its MDF and 5 minutes to process, it must operate autonomously in the urban environment: the vehicle must drive in the available lanes, stop at stop lines, and obey traffic laws, all without receiving external guidance. Localization and orientation within the RNDF are key elements for facilitating such precise autonomous navigation, but the challenging urban canyon signal environment and the possibility of longterm signal obstruction means that GPS, and even GPS fused with inertial navigation, may not be accurate enough for a vehicle to obey all rules of the road.

Fortunately, visual landmarks corresponding to the surveyed information in the RNDF can be used as additional position and orientation cues to aid a GPS-based navigation solution. The PosteriorPose algorithm, presented in the sequel, takes advantage of these cues to improve its estimate of the vehicle state $x(k) = [e(k)\,; n(k)\,; h(k)]^T$ at time k, where $e(k)$ and $n(k)$ are the vehicle's Easting and Northing with respect to the RNDF's center, and $h(k)$ is the vehicle's heading.

3.3 The PosteriorPose Algorithm

3.3.1 A Brief Review Of Recursive Particle Filtering

We introduce the PosteriorPose algorithm first with a brief review of recursive particle filtering. The general filtering problem begins with a desire to estimate

the posterior density of the state vector $x(k)$ at time k given a time series of sensor measurements $\{z(0), \ldots, z(k)\} = Z(k)$. The particle filtering approach to the traditional filtering problem seeks to approximate the posterior density with a set of N particles $\left\{\chi^1(k), \ldots, \chi^N(k)\right\}$ and weights $\left\{w^1(k), \ldots, w^N(k)\right\}$ in a discretized form:

$$p\left(x(k)\,|Z(k)\right) \approx \sum_{i=1}^{N} w^i(k) \cdot \delta\left(x(k) - \chi^i(k)\right) \tag{3.1}$$

where $\delta(\cdot)$ represents the Dirac delta-function, and the weights $w^i(k)$, for $i \in [1, \ldots, N]$, sum to unity [7]. Intuitively, each particle $\chi^i(k)$ represents a possible value of the unknown true state $x(k)$, and its weight $w^i(k)$ is related to the likelihood of that value being correct.

The relation between the particles and their weights determines whether (and how well) the discrete particle density approximates the true posterior. Since the weights cannot be chosen with impunity, the technique of *importance sampling* is most commonly employed to assign weights once a given set of particles is selected [7]. Importance sampling begins with a given set of particles, chosen randomly according to a *proposal density* $\bar{p}\left(x(k)\,|Z(k)\right)$. The weights are then assigned to each particle according to the likelihood ratio between the true and the proposal densities:

$$w^i(k) = \frac{p\left(\chi^i(k)\,|Z(k)\right)}{\bar{p}\left(\chi^i(k)\,|Z(k)\right)} \tag{3.2}$$

This choice of weights has several important properties. The first is convergence: the moments of the discrete particle density approach the moments of the true posterior density as the number of particles $N \to \infty$, and the discrete density itself can be shown to approach the true posterior under the same conditions [7]. The second benefit is in feasibility: if it is relatively inexpensive to draw particles from $\bar{p}(\cdot)$ compared to $p(\cdot)$, then the importance sampled

density represents a practical and viable way to generate samples and various measures of the true posterior density. Note, however, that these benefits depend on the proposal density having a domain and shape 'similar' to the true density in order to discourage $w^i(k) = 0$ and $w^i(k) = \infty$; in fact, many of the problems that classically plague the particle filter in practical settings arise as weights approach either of these two extremes [7].

Recursive particle filtering arises naturally from the importance sampling framework as the consequence of the following choice of the proposal density:

$$\begin{aligned} \bar{p}\left(\chi^i(k)\,|Z\,(k)\right) &= q\left(\chi^i(k)\,|\chi^i(k-1)\,,Z(k-1)\right) \\ &\cdot \quad \bar{p}\left(\chi^i(k-1)\,|Z(k-1)\right) \end{aligned} \tag{3.3}$$

Combining this with Bayes's rule applied to $p\,(\chi^i(k)\,|Z(k))$ from Equation 3.2 yields the popular recursive weight update [7]:

$$\begin{aligned} w^i(k) &\propto w^i(k-1)\cdot p\left(z(k)\,|\chi^i(k)\right) \\ &\cdot \quad \frac{p\,(\chi^i(k)\,|\chi^i(k-1))}{q\,(\chi^i(k)\,|\chi^i(k-1)\,,Z(k-1))} \end{aligned} \tag{3.4}$$

where '$\propto$' is used because a normalizing constant has been dropped from the formulation. Often, $q\,(\cdot)$ is further restricted when the filtering problem is accurately modeled as first order Markov:

$$q\left(\chi^i(k)\,|\chi^i(k-1)\,,Z(k-1)\right) = p\left(\chi^i(k)\,|\chi^i(k-1)\right) \tag{3.5}$$

That is, the particles are chosen according to the predicted state density. This reduces the recursive weight update to a relatively manageable form:

$$w^i(k) \propto w^i(k-1)\cdot p\left(z(k)\,|\chi^i(k)\right) \tag{3.6}$$

which only requires knowledge and evaluation of the measurement likelihood function $p\left(z(k)\,|\chi^i(k)\right)$ to update the particles' weights with new sensor measurements. Unfortunately, this weight update will often cause all but a few particles' weights to tend to zero after repeated updating, even with the most carefully-chosen proposal distribution [7]. To circumvent this *degeneracy* problem, the particle filter is occasionally resampled to refresh the set of particles and their weights. During a resampling step, a new set of N particles is chosen from the existing set of particles by sampling the existing particles with replacement, using the particles' weights as their selection probabilities. The old particles are then replaced with the new set, and the weights of all the new particles are set to $1/N$. The weight reset is performed because the new particle set itself is already distributed according to the existing approximation of the posterior density $p\left(x(k)\,|Z(k)\right)$ encoded by the old particle set [7]. Resampling therefore renews the particle set as it tends to make multiple copies of particles with high weights and remove particles with very low weights. Intuitively, resampling concentrates more particles in the regions of state space that are likely to contain the true state $x(k)$.

Resampling within the particle filter should be performed frequently to prevent particle degeneracy. One metric often employed to detect the need for resampling is the estimate of the effective number of particles $\hat{N}$ [7]:

$$\hat{N} = \frac{1}{\sum_{i=1}^{N}\left(w^i(k)\right)^2} \tag{3.7}$$

which lies in the range $1 < \hat{N} < N$ and intuitively gives a measure of how many of the particles are actually making significant contributions to the estimate of the posterior density. This measure $\hat{N}$ can then be thresholded as a test of whether resampling is needed at any given time.

The particle filter presented in this section, which uses Equation 3.5 to draw particles, Equation 3.6 to update particle weights, and Equation 3.7 to test for resampling, is used to generate vehicle state estimates in the PosteriorPose algorithm. The only notational difference in the discussion that follows is that all terms are conditioned on surveyed RNDF map information M brought to bear on the estimation problem in addition to available sensor measurements.

3.3.2 Generating The Proposal Distribution Via Onboard Inertial Navigation

The recursive particle filtering technique discussed in Section 3.3.1 utilizes the predicted state density $p\left(\chi^i(k)\,|\chi^i(k-1)\,,M\right)$ as the proposal density $q\left(\chi^i(k)\,|\chi^i(k-1)\,,Z(k)\,,M\right)$ to draw a set of particles at time k from an existing set at time $k-1$. Note the explicit conditioning on surveyed RNDF map information M, which is added to estimate a map-conditioned posterior state density $p\left(x(k)\,|Z(k)\,,M\right)$ as explained in Section 3.3.1.

For a vehicle driven by a rational intelligent agent such as a human, map information M provides important constraints on the predicted state density. Section 3.1 argues that these constraints are not present in autonomous systems, which depend heavily on noisy sensors and lack the 'common sense' to understand when to ignore them. The PosteriorPose algorithm therefore argues that the predicted state density is independent of map information:

$$p\left(\chi^i(k)\,|\chi^i(k-1)\,,M\right) = p\left(\chi^i(k)\,|\chi^i(k-1)\right) \tag{3.8}$$

and when this density is selected as the proposal density $q\left(\chi^i(k)\,|\chi^i(k-1)\,,Z(k)\right)$,

it too is independent of map information. Following this argument, the PosteriorPose algorithm defines the proposal density independent of map information according to the following vehicle dynamics model:

$$\chi^i(k) = \chi^i(k-1) + \Delta\chi^i\,(k-1) + v^i\,(k-1) \tag{3.9}$$

where $\Delta\chi^i\,(k-1) = [\Delta e^i(k-1)\,, \Delta n^i(k-1)\,, \Delta h^i(k-1)]^T$ are the measured values of the state's transition and $v^i\,(k-1) = [v_e^i(k-1)\,, v_n^i(k-1)\,, v_h^i(k-1)]^T$ are the errors in those measurements, commonly called the *process noise*. Measurements of the state's transition $\Delta\chi^i(k-1)$ are commonly obtained by integrating onboard odometry sensors such as an inertial measurement unit or wheel encoders. While the errors in these integrations are not known exactly, their statistics are modeled in light of the Central Limit Theorem as jointly Gaussian and white:

$$\begin{aligned} E\left[v^i(k)\right] &= 0, \forall i, k \\ E\left[\left(v^i(j)\right)\left(v^i(k)\right)^T\right] &= Q(k)\cdot\delta_{jk}, \forall i, j, k \end{aligned} \tag{3.10}$$

where $Q(k)$ is the process noise covariance matrix, and δ_{jk} is the Kronecker delta function [11]. Conveniently, this Gaussian model also makes the task of drawing $v(k) \sim \mathcal{N}\,(0, Q(k))$ quite practical, as zero-mean jointly Gaussian variables with arbitrary covariance can be drawn from the standard normal density through an affine transformation:

$$v^i(k) = C^T(k)\cdot y^i(k) \tag{3.11}$$

where $C^T(k)\cdot C(k) = Q(k)$ is the Cholesky factorization of $Q(k)$, and $y^i(k) \sim \mathcal{N}\,(0, \mathcal{I})$ is drawn from the standard normal density. This technique is used in the PosteriorPose filter to draw the proposed set of particles at time k given

those at time $k-1$ according to the transition density $p\left(\chi^i(k)\,|\chi^i(k-1)\right) \sim \mathcal{N}\left(\chi^i\left(k-1\right)+\Delta\chi^i\left(k-1\right), Q\left(k-1\right)\right)$.

3.3.3 Updating Particle Weights With Absolute And Relative Sensor Cues

The recursive particle filtering technique discussed in Section 3.3.1 evaluates the measurement likelihood function $p\left(z(k)\,|\chi^i(k), M\right)$ in Equation 3.6 to update particle weights $w^i(k)$ with a sensor measurement $z(k)$ in light of RNDF map information M. This brings the new sensor information to bear on the particles by fusing the information into the posterior density estimate $p\left(x(k)\,|Z(k), M\right)$. In the PosteriorPose algorithm, two pieces of distinct sensor information $z(k)$ are used: absolute measurements provided by GPS or other navigation sensors, and relative measurements of visual cues provided by onboard cameras. While these measurements are plentiful, the possibility of spurious signals and false detections means that careful modeling, monitoring, and validation are necessary to avoid updating particle weights with wildly incorrect measurements.

Absolute Sensor Cues

Absolute measurements $z_a(k)$ are the most straightforward to fuse in the particle filtering framework, because they are direct measurements of the state. If the measurements are generated by one or more GPS receivers in a non-recursive least squares scheme, the measurement likelihood function is well-modeled by

Gaussian white noise of mean $\chi^i(k)$:

$$p\left(z_a(k)\,|\chi^i(k)\,,M\right) \sim \mathcal{N}\left(\chi^i(k)\,,R_{aa}(k)\right) \tag{3.12}$$

where $R_{aa}(k)$ is the covariance matrix of the least squares estimate [11]. If, however, the absolute measurements are generated as the result of a recursive estimator, perhaps by integrating GPS velocities or an inertial navigation system, then the autocorrelation in that estimator's error must also be modeled. In such a case, the measurements can be whitened by augmenting the state $x(k)$ with East, North, and Heading biases $\beta(k) = [\beta_e^i(k)\,,\beta_n^i(k)\,,\beta_h^i(k)]^T$ obeying the following dynamics:

$$\beta_*^i(k) = \lambda \cdot \beta_*^i(k-1) + v_{\beta_*}^i(k-1) \tag{3.13}$$

where $\lambda = \exp\left(-dt/T_b\right)$ accounts for the bias's autocorrelation time T_b during the elapsed time dt between time indices $k-1$ and k, and v_β^i is a zero-mean Gaussian white noise process [11]. With this addition, the measurement noise is once again Gaussian and white, but the particle filter will no longer be overly-confident of the absolute position measurements it receives:

$$\begin{aligned} &p\left(z_a(k)\,|\chi^i(k)\,,\beta^i(k)\,,M\right) \\ &\qquad \sim \mathcal{N}\left(\chi^i(k)+\beta^i(k)\,,R_{aa}(k)\right) \end{aligned} \tag{3.14}$$

Note that although equations 3.12 and 3.14 are written as being conditioned on RNDF map information M, neither assumes any map-based driving constraints. The PosteriorPose algorithm thus treats absolute measurements as independent of map information:

$$p\left(z_a(k)\,|\chi^i(k)\,,M\right) = p\left(z_a(k)\,|\chi^i(k)\right) \tag{3.15}$$

This maintains the framework developed in Section 3.1, where the vehicle to be localized is not constrained to the road.

Even with the addition of biases in equation 3.14 to model the autocorrelation of absolute positioning sensors, the threat of significant signal multipath in the urban environment makes it possible for these sensors to occasionally produce grossly inaccurate measurements that are not modeled by the likelihoods in equations 3.12 and 3.14. For that reason, all absolute measurements are first validated using an approximate hypothesis test similar to the *gating* procedure commonly used in Kalman filtering [11]. First, the minimum mean squared error (MMSE) estimate $\hat{x}(k)$ is generated from the particle filter along with its mean squared error matrix (MSE) $P_{xx}(k)$:

$$\hat{x}(k) = \sum_{i=1}^{N} w^i(k) \cdot \left(\chi^i(k) + \beta^i(k)\right) \tag{3.16}$$

$$P_{xx}(k) = \sum_{i=1}^{N} w^i(k) \cdot \left[\Delta x^i(k)\right] \left[\Delta x^i(k)\right]^T \tag{3.17}$$

where $\Delta x^i(k) = \chi^i(k) + \beta^i(k) - \hat{x}(k)$ is the difference between the i^{th} particle's predicted vehicle location and the MMSE vehicle location. Next, an approximate innovation statistic $\epsilon_a(k)$ is formed:

$$\begin{aligned} \epsilon_a(k) &= (z_a(k) - \hat{x}(k))^T \\ &\cdot (P_{xx}(k) + R_{aa}(k))^{-1} \\ &\cdot (z_a(k) - \hat{x}(k)) \end{aligned} \tag{3.18}$$

which, if the distribution of particles and the measurement noise are truly Gaussian, will be a χ^2 random variable with three degrees of freedom [11]. Thresholding against a sensible value, such as the 95% bound of $\epsilon_a(k)$, effectively tests the hypothesis that the measurement $z_a(k)$ is modeled correctly. In the PosteriorPose algorithm, the test is only approximate since the particles need not be

distributed according to a Gaussian. Nevertheless, the test still provides a sensible neighborhood for defining which measurements are or are not reasonable.

Relative Sensor Cues

Unlike absolute measurements from the navigation sensors, relative measurements from optical sensors used in the PosteriorPose algorithm are not direct measurements of the state $x(k)$. Instead, these sensors generate two types of information: measurements of nearby lanes and measurements of nearby stop lines. These types of information are measured in a camera-centric or vehicle-centric coordinate frame. Though these relative sensor cues are very weak positioning signals by themselves, they provide high accuracy relative measurements that combine well with weak or intermittent absolute position fixes.

The challenge in incorporating relative measurements from optical sensors lies in modeling measurement errors. These errors are not Gaussian, since the measurements are generated by high-level image processing algorithms instead of raw image data. Such algorithms rarely feature linear or near-linear mappings from pixels to detections. Therefore even if pixel intensity errors were Gaussian, errors in the high-level algorithms still would not be. Without an accurate error model, difficulties resulting from these non-Gaussian errors would dilute the utility of relative sensor cues and the overall quality of the position estimate.

A key realization in the PosteriorPose algorithm is that the errors, though non-Gaussian, are not entirely unpredictable. In particular, *a priori* information available in the RNDF can be used to predict likely false positives, which are

often closely-related to a small number of nearby features present in the environment and marked in the RNDF. The PosteriorPose algorithm enumerates these false positives explicitly in its model of errors in relative sensor cues, using them as modes of multimodal measurement likelihoods.

The first type of relative sensor modeled in this manner measures nearby visible lanes, generating lane measurements $z_l(k)$ of the camera's perpendicular distance from each of a lane's boundaries and also the camera's heading with respect to the lane. Predictably, errors in this type of lane finding algorithm are not Gaussian; common segmentation and convolution approaches to vision-based lane finding occasionally detect the wrong lane, identify lanes where none exist, or combine multiple lanes into a single large lane measurement [30]. In the PosteriorPose algorithm, the RNDF is used to model these types of sensor error explicitly. In particular, the measurement likelihoods conditioned on the j^{th} sensor error μ_{lj} are modeled as Gaussian:

$$\begin{aligned} & p\left(z_l(k)\,|\chi^i(k)\,,\mu_{lj},M\right) \\ & \quad \sim \mathcal{N}\left(\bar{z}_l\Big(k,\chi^i(k)\,,\mu_{lj},M\Big)\,,R_{ll}(k,\mu_{lj})\right) \end{aligned} \tag{3.19}$$

where $\bar{z}_l(k,\chi^i(k)\,,\mu_{lj},M)$ is the expected lane measurement calculated using the i^{th} particle's state in the j^{th} camera detection mode with RNDF map information M, and $R_{ll}(k,\mu_{lj})$ is the measurement covariance matrix conditioned on the j^{th} camera detection mode. This model is applied to six different modes of detection: (1) detecting the correct lane, detecting the lane to the left (2) or right (3) of the correct lane, (4) detecting the left and correct lane as one lane, (5) detecting the right and correct lane as one lane, and (6) detecting the left, right, and correct lanes as a single lane. A seventh sensing mode (7), representing a false lane detection where no lane exists, is modeled with uniform likelihood over a

feasible range of lane distances and headings:

$$p\left(z_l(k)\,|\chi^i(k)\,,\mu_{l7},M\right)\sim\mathcal{U}\Big(\bar{z}_l\Big(k,\chi^i(k)\,,\mu_{l7},M\Big)\Big) \tag{3.20}$$

The measurement likelihood is then calculated as a mixture of each of these modes of detection:

$$\begin{aligned} &p\left(z_l(k)\,|\chi^i(k)\,,M\right)\\ &\qquad=\sum_{j=1}^{7}p\left(\mu_{lj}\right)\cdot p\left(z_l(k)\,|\chi^i(k)\,,\mu_{lj},M\right) \end{aligned} \tag{3.21}$$

where $p\left(\mu_{lj}\right)$ is a tuning parameter representing the prior probability of the sensor making the j^{th} type of detection.

As with the absolute positioning sensors, hypothesis tests are applied to relative lane measurements to further prevent spurious measurements from corrupting the particles' weights. First, the particle filter computes the probability that no lane exists by summing the weights of the particles that believe no lane should be visible. This probability is then thresholded to a desired false positive rate, so that lane measurements are not applied if the particle filter is confident that the lanes being measured are not designated in the RNDF. Second, the particle filter computes an MMSE estimate $\hat{z}_l(k)$ and MSE matrix $P_{ll}(k)$ of the expected lane measurements assuming the sensor measures the correct lane:

$$\hat{z}_l(k)=\sum_{i=1}^{N}w^i(k)\cdot\bar{z}_l\left(k,\chi^i(k)\,,\mu_{l1},M\right) \tag{3.22}$$

$$P_{ll}(k)=\sum_{i=1}^{N}w^i\left(k\right)\cdot\left[\Delta z_l^i(k)\right]\left[\Delta z_l^i(k)\right]^T \tag{3.23}$$

where $\Delta z_l^i(k)=\bar{z}_l(k,\chi^i(k)\,,\mu_{l1},M)-\hat{z}_l(k)$ is the difference between the i^{th} particle's expected lane measurement and the MMSE lane measurement. An innovation statistic $\epsilon_l(k)$ is then formed similar to Equation 3.18 using $z_l(k)$, $\hat{z}_l(k)$,

$P_{ll}(k)$, and $R_{ll}(k, \mu_{l1})$ and then thresholded as if it were a χ^2 random variable with two degrees of freedom: one for perpendicular lane distance and one for lane heading.

The second type of relative sensor utilized in the PosteriorPose algorithm classifies the types of line marks that bound each lane as to whether they are single lines, double lines, dashed lines, or virtual lines. This information is weaker than the lane distance and heading measurements, but helpful in determining which lane the vehicle is in on a multi-lane road. The line mark classifier is used to update particle weights using Equation 3.6 like other sensors, except the measurement likelihood is a discrete probability mass function:

$$p\left(z_m(k)\,|\chi^i(k)\,, M\right) = p\left(z_m(k)\,|\bar{z}_m\Big(k, \chi^i(k)\,, M\Big)\right) \tag{3.24}$$

where $\bar{z}_m(k, \chi^i(k)\,, M)$ is the vector of expected line mark classifications given the vehicle state $\chi^i(k)$ and the RNDF map information M, and $p\left(z_m(k)\,|\bar{z}_m(k, \chi^i(k)\,, M)\right)$ is the probability of the line mark classifier generating the measurement $z_m(k)$ given the actual lane line marks are of type $\bar{z}_m(k, \chi^i(k)\,, M)$. Intuitively, Equation 3.24 expresses the measurement likelihood in terms of the line classifier's classification accuracy and false positive rates.

Two hypothesis tests are performed to validate the line classifier's measurements before incorporating them into the PosteriorPose algorithm. The first hypothesis test applies a threshold to the probability that no lane exists, which is calculated in the same manner as with the relative lane measurements. The second test calculates the probability that the lane marks have been classified correctly by summing across all particles whose expected line mark classifications $\bar{z}_m(k, \chi^i(k)\,, M)$ match the actual measurement $z_m(k)$. A threshold is ap-

plied to this probability as well, ensuring that line mark classifications do not update the particle weights unless the filter is reasonably certain that the line marks were classified correctly.

The final sensor providing relative measurements is the stop line sensor, which measures the distance to any stop line visible within its camera's field of view. The likelihood function for this sensor is modeled like the lane sensor by enumerating two modes of detection: μ_{s1} and μ_{s2}. These mode-conditional likelihoods, which model the sensor's behavior when measuring an actual stop line vs. a false detection, are then combined in a mixture similar to Equation 3.21 to calculate the likelihood of a particular stop line measurement. Like the lane sensor, the stop line sensor's measurement likelihood conditioned on a correct detection is modeled as Gaussian:

$$p\left(z_s(k)\,|\chi^i(k)\,,\mu_{s1},M\right) \sim \\ \mathcal{N}\left(\bar{z}_s\Big(k,\chi^i(k)\,,\mu_{s1},M\Big)\,,R_{ss}(k,\mu_{s1})\right) \tag{3.25}$$

where $\bar{z}_s(k,\chi^i(k)\,,M)$ is the expected stop line measurement given a particular vehicle state $\chi^i(k)$ and RNDF map information M, and $R_{ss}(k,\mu_{s1})$ is the stop line measurement covariance matrix. Similarly, the stop line sensor's measurement likelihood conditioned on a false detection is modeled as uniform over a feasible range of stop line distances:

$$p\left(z_s(k)\,|\chi^i(k)\,,\mu_{s2}\right) \sim \mathcal{U}\left(\bar{z}_s\Big(k,\chi^i(k)\,,\mu_{s2},M\Big)\right) \tag{3.26}$$

Equations 3.25 and 3.26 are then combined to create a mode-dependent likelihood similar to equation 3.21:

$$p\left(z_s(k)\,|\chi^i(k)\,,M\right)$$

$$= \sum_{j=1}^{2} p\left(\mu_{sj}\right) \cdot p\left(z_s(k) \left| \chi^i(k), \mu_{sj}, M \right.\right) \tag{3.27}$$

where the stop line sensor's two detection modes are mixed according to tuning parameters p_{s1} and p_{s2}, which intuitively represent the stop line sensor's *a priori* probability of correct detection and false positive rate, respectively.

As with the other sensors, hypothesis tests are applied to the stop line measurements to gate them before they are used update particle weights in the PosteriorPose algorithm. First, the PosteriorPose particles are used to compute the probability that a stop line is visible by summing across the weights of all particles that expect to see a stop line. This probability is then thresholded to some desired false positive rate on the stop line sensor. Next, an MMSE estimate of the distance to the stop line is produced along with an MSE matrix using equations analogous to 3.22 and 3.23 for the stop line measurement. Finally, the MMSE stop line estimate and its MSE matrix are used to form an innovation statistic similar to Equation 3.18, which is then thresholded as a χ^2 random variable with 1 degree of freedom.

Each sensor discussed in this section is used to update the PosteriorPose algorithm with the likelihood weighting method of Equation 3.6, provided the sensor's measurements pass the associated hypothesis tests. In this way, the extra information generated by the local sensors is aggregated into a unified vehicle state estimate without sacrificing accuracy through significant sensor mistakes. Intuitively, each relative sensor cue discussed in this section provides position and orientation information by comparing sensor measurements generated from the nearby environment with expected sensor measurements generated from the surveyed road network. If the sensing vehicle should travel off

the road or outside the area surveyed in the road network, the relative sensing cues will be unable to provide any information to improve the vehicle's pose estimate. With no relative sensing updates to apply, the PosteriorPose algorithm simply filters the absolute information provided by its onboard GPS / INS sensing. In this case the algorithm naturally reverts to a secondary layer of GPS filtering, effectively reporting a filtered version of the available absolute positioning information.

3.4 Implementation And Hardware Setup

The PosteriorPose algorithm described in Section 3.3 has been implemented in real-time on Cornell University's 'Skynet,' an autonomous 2007 Chevrolet Tahoe shown in Figure 3.1. Skynet is equipped with sensors similar to those described in Section 3.3: a Septentrio PolaRx2e@ tri-antenna GPS receiver providing GPS position, velocity, and attitude, a Litton LN-200 tactical grade inertial measurement unit providing inertial navigation, a MobilEye SeeQ camera and vision module providing road lane and line estimates, and Cornell's own texture and edge-based vision algorithms providing redundant lane line and stop line estimates [56]. The GPS antennas are mounted on Skynet's roof, and they are tightly-coupled with the IMU and wheel odometry in a Square Root Information Filter to produce the filtered GPS / INS navigation solution [59]. This navigation solution provides vehicle position and odometry measurements at 100 Hz. The MobilEye and Cornell lane and line detection systems are run on images generated by a forward-looking Basler A622F camera, mounted in the center of Skynet's roof. The MobilEye SeeQ system is commercially available as a lane departure warning system, and runs at approximately 15 Hz. It reli-

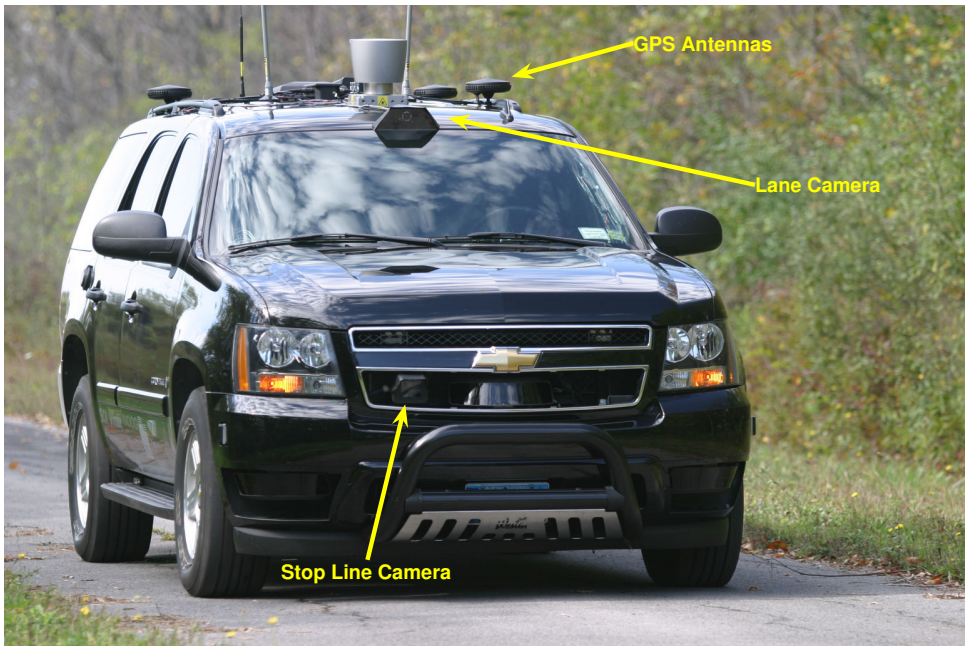

Figure 3.1: The PosteriorPose algorithm is implemented in real-time on Cornell University's 'Skynet,' an autonomous 2007 Chevrolet Tahoe that completed the 2007 DARPA Urban Challenge.

ably detects painted lines, though it requires several seconds of uninterrupted tracking to become confident in its estimates. The Cornell lane detection system is designed as a slower but more accurate complement to the MobilEye; it uses Felzenszwalb and Huttenlocher's image segmentation algorithm to detect lanes from texture boundaries in single image frames, even when there are no painted road lines [25]. The Cornell lane detection system runs at approximately 2 Hz. The Cornell stop line detection system is run on images generated by a Basler A311F camera, mounted in Skynet's front bumper. The algorithm detects stop lines by applying Canny edge detection to each camera image. The resulting edge image is searched for pairs of oppositely-oriented edges corresponding to upper and lower boundaries of stop line candidates. Potential stop line candidates are broadcasted at approximately 17.5 Hz.

The PosteriorPose algorithm is implemented in C++ and runs in real-time on a single core of a 2.0 GHz Intel Core 2 Duo machine running Microsoft Windows Server 2003. The PosteriorPose algorithm receives all sensor measurements via UDP over Skynet's internal network, and each measurement is timestamped by synchronized microcontrollers prior to transmission. In order to ensure the measurements are applied in the correct time order within the PosteriorPose algorithm, a queue of sensor measurements is implemented at the receiving end. This queue stores all sensor measurements in order of ascending timestamp, and only allows the PosteriorPose algorithm to apply a measurement update when all expected measurements for a particular window of time arrive. This window of time is set to 500 ms on Skynet, which is slightly longer than the delay in the slowest sensor to ensure that all measurements are applied to the PosteriorPose algorithm in temporal order. Unfortunately, this technique also introduces a 500 ms delay in the PosteriorPose estimate, since measurements are not applied until they are at least 500 ms old. This delay is removed prior to reporting a PosteriorPose estimate by temporarily predicting the PosteriorPose particles forward from the timestamp of the last applied measurement to the present time using vehicle odometry. These predicted particles are then used to generate a vehicle state estimate with no delay.

The surveyed road network, provided in the RNDF file format in advance, includes waypoint and stop line locations as well as lane descriptions [19]. Since the PosteriorPose algorithm must access this information repeatedly for each particle, searching the road network efficiently is critical for a real-time implementation of the PosteriorPose algorithm. On Skynet, the road network is preprocessed into an efficient graphical data structure, called the *road graph*, where each pair of connected waypoints is a vertex. Pointers to these vertices are

stored in arrays hashed both by unique waypoint and lane identifiers. A special pointer is maintained at Skynet's current MMSE position to permit rapid acquisition of the current road segment at any time. Lists of vertices nearby and adjacent to each vertex are also precomputed and stored in the road graph, allowing the PosteriorPose algorithm to access nearby lane descriptions without searching the entire road graph at each measurement update. Storing lists of edges in this manner is ideally suited to real world road networks, which have large numbers of vertices with few neighbors. Lists of nearby stop lines are also precomputed for each vertex in a similar manner.

Despite computational shortcuts enabled by neighborhood definitions in the road graph, it is occasionally necessary to find the closest vertex to an arbitrary point in the road network without prior information. This need may arise if the vehicle travels too far off the road network, endures a long GPS blackout without significant road information, or if the PosteriorPose algorithm is initialized with poor prior position information. These cases arise frequently enough in practice to warrant implementation of an efficient closest vertex searching algorithm. On Skynet's implementation of the PosteriorPose algorithm, the road network is overlaid with a bounding grid. The grid boundaries are assumed large enough that Skynet will never leave the grid. Prior to the start of the PosteriorPose algorithm, a list is computed for each grid cell that contains all vertices that could possibly be closest to any point in that grid cell. Arguments relating to the triangle inequality are used to ensure the list is exhaustive. When a closest vertex is required at run-time, the list of possible closest vertices is retrieved for the grid cell containing the queried position. The resulting list of vertices is then searched exhaustively to find the closest vertex. The overall search procedure runs on average in constant time, similar to a lookup operation in a hash table.

The computational effort spent up front in creating the road graph and vertex grid makes a significant difference: without it, the implementation of the PosteriorPose algorithm barely maintains 50 Hz update rates with 400 particles on a single road loop with 100 waypoints. With the road graph and vertex grid, the implementation achieves update rates exceeding 100 Hz with 2000 particles on arbitrarily-sized road networks. At these settings the algorithm almost fully loads the processor core on which it is run, with most time spent resampling the particle filter and drawing Gaussian random numbers.

3.5 Experimental Results

The capabilities of the PosteriorPose algorithm are demonstrated in driving experiments at the Seneca Army Depot in Romulus, NY. This closed road network includes several tens of miles of surveyed roads, accurately painted road lines and stop lines, and a variety of GPS signal environments. The Depot also includes a variety of features challenging to sensing, including paved and unpaved roads, some weathered road lines, potholes, railroad tracks, considerable short and tall vegetation, and storage buildings. Skynet is shown on a portion of this course in Figure 3.1. Experiments with the PosteriorPose algorithm are performed only in areas with an unobstructed view of the sky, to permit an OmniSTAR high precision (HP) differential corrections service, nominally accurate to 10 cm, to be used as a source of truth data. GPS signals are then artificially withheld in the PosteriorPose algorithm to simulate more difficult GPS environments. Experimental data is collected at normal city driving speeds, up to 30 mph (13.4 m/s).

The PosteriorPose algorithm is tested on two driving courses at the Depot. On both courses, Skynet's positioning algorithms are initialized for 5 minutes before driving multiple loops of the specified course for 30 minutes. Logged sensor data from these driving experiments is then presented to the PosteriorPose algorithm under different levels of GPS signal availability. The results are compared using the OmniSTAR HP GPS signal as a ground truth. Comparisons are made between variants of the PosteriorPose algorithm by computing East-North position errors $E(k) = \left\| [\hat{e}(k), \hat{n}(k)]^T - [e(k), n(k)]^T \right\|$ using MMSE position estimates calculated as in equations 3.16 and 3.17. Errors from these MMSE estimates are evaluated at a rate of 1 Hz.

The MMSE estimate used to evaluate the PosteriorPose algorithm in the following experiments is only one type of estimate that can be produced from the flexible representation of the particle filter. The MMSE estimate is convenient for evaluation purposes because it produces a single position estimate from the hypotheses represented by the particles. The estimate also happens to be particularly stable in practice, since it is a function of all the particles rather than one or a small set. These properties make the MMSE estimate a suitable feedback signal for large-scale robotic path planning, as demonstrated with Skynet's 60 mile autonomous drive during the 2007 DARPA Urban Challenge [56]. Nevertheless, the MMSE estimate can be a poor solution when the particle set is strongly bimodal or multimodal. Such cases happen frequently in very sparse GPS environments, where relative sensor cues might localize the vehicle to the road without clearly identifying which lane it occupies. Such issues remain open problems in the correct use of probabilistic information for robotic path planning, especially as rich multimodal representations of the posterior state density become more feasible to maintain in real-time. Here, the MMSE esti-

mate is used to evaluate the PosteriorPose algorithm in all driving experiments due to its ubiquitous use in practical robotics.

3.5.1 Course 1: Dense Road Information

The first driving course is a loop approximately 2.5 km long consisting of dense road features: 8 stop lines, 16 turns, 2.1 km of painted solid lines, 0.2 km of painted dashed lines, and 0.2 km of unpainted roads. A data log is collected on this course consisting of approximately 5 minutes of stationary GPS data for initialization, followed by 30 minutes of driving. Skynet's lane and stop line algorithms are activated at the end of the stationary initialization period and remain active for the remainder of the experiment. A total of 5 loops around the course are completed in the time allotted.

Two experiments are run on the data collected from the first driving course. In the first experiment, the PosteriorPose algorithm is run on the logged sensor data with Skynet's full sensor suite and full GPS availability. The PosteriorPose algorithm is run twice in this configuration: once estimating only Skynet's East position, North position, and heading, and once additionally estimating East and North GPS biases. Results from these two runs are compared to ground truth provided by the OmniSTAR HP differential corrections service fused with Skynet's GPS and inertial navigation sensors. No GPS blackouts occur in this first experiment. Figure 3.2 plots a time history of position error magnitudes for each variant of the algorithm, along with a time history of position error magnitudes of the fused GPS / INS solution without OmniSTAR HP differential corrections.

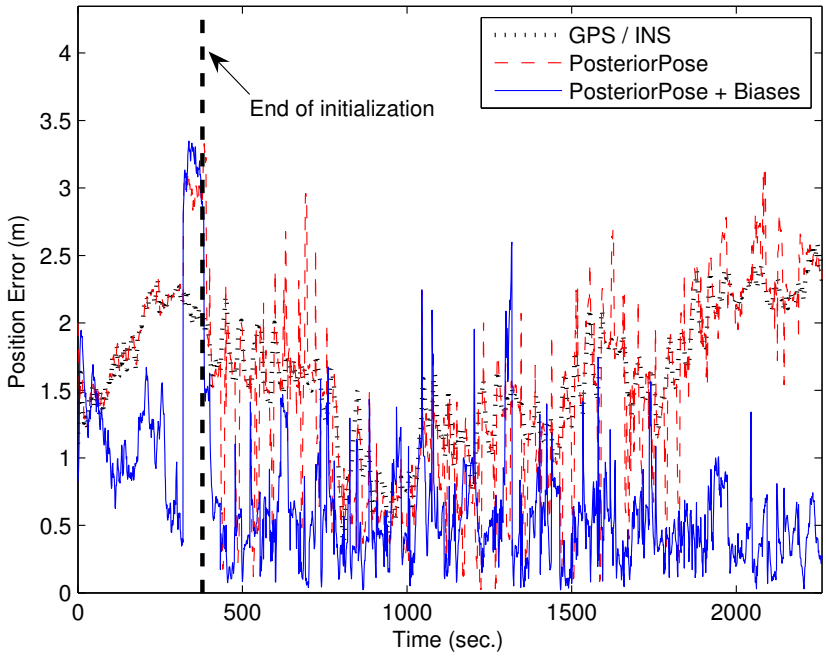

Figure 3.2: Variants of the PosteriorPose algorithm run with no GPS blackouts on a course with dense road information. The PosteriorPose algorithm produces lower errors on average than the GPS / INS solution, even with GPS fully available. When additionally estimating GPS biases, the PosteriorPose algorithm uses road information to significantly reduce the effects of time-correlated GPS errors.

In many error sampling intervals, the PosteriorPose algorithm yields lower position errors than the GPS / INS solution even when not estimating GPS biases. At best, this variant of the PosteriorPose algorithm beats the GPS / INS solution by 1.82 m, though on average its errors are only 0.11 m lower in the full GPS environment. Despite the seemingly small benefit, the average performance improvement is statistically significant: a paired T-test run on the results concludes that this variant of the PosteriorPose algorithm produces at least 0.09 m less error than Skynet's fused GPS / INS solution at the 5% significance level. The results improve substantially when PosteriorPose is allowed to esti-

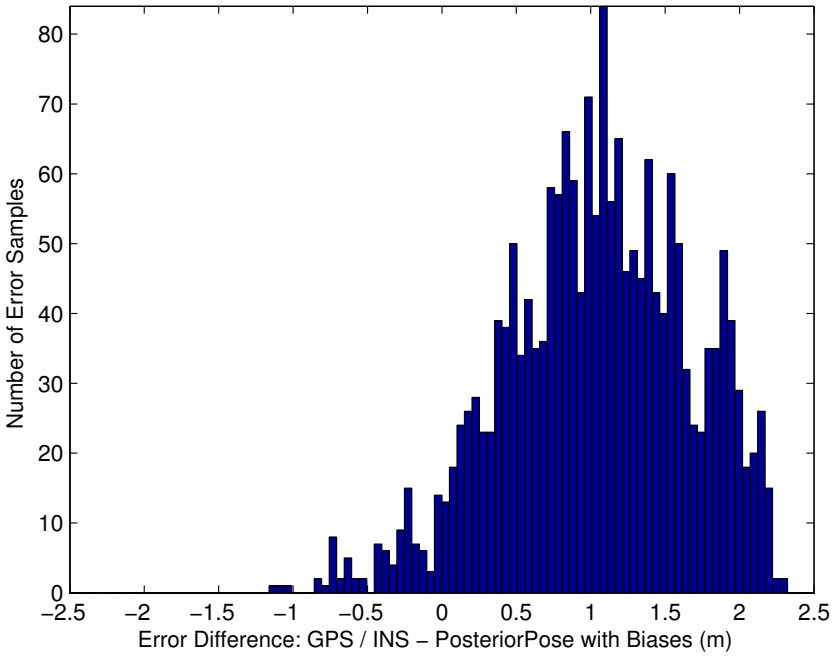

Figure 3.3: Differences in error between the fused GPS / INS solution and the PosteriorPose algorithm estimating GPS biases. The PosteriorPose algorithm uses road information to significantly reduce the effects of time-correlated GPS errors.

mate GPS biases; on average, this addition achieves errors 1.04 m lower than the fused GPS / INS solution. Statistically, estimating GPS biases in the PosteriorPose algorithm reduces positioning errors by at least 1.02 m over the fused GPS / INS solution at the 5% significance level. This represents a substantial reduction in error, as the average error in the fused GPS / INS system is 1.58 m. A histogram of differences in error between the fused GPS / INS solution and the PosteriorPose algorithm estimating biases is shown in Figure 3.3.

A notable exception to the statistical improvements made by the PosteriorPose algorithm exists near time $t \approx 320 - 380$ sec., where both variants of the PosteriorPose algorithm have approximately 1 m more error than the fused GPS / INS solution. The error occurs during initialization, just before Skynet starts to

move, and is the result of one of Skynet's lane detection algorithms measuring the wrong lane with insufficient information to reject the error. The temporary error is eliminated when Skynet begins to move, as the lane ambiguity is easily resolved by aggregating information across different viewpoints.

Although the error is resolved quickly when Skynet begins to move, it emphasizes the difficulty of using map-based relative cues to estimate time-correlated biases in absolute positioning signals. Relative sensor cues possess an inherent ambiguity if multiple map objects, such as lanes, can produce the same sensor measurements. This ambiguity, related to the *data association* problem in target tracking, may temporarily admit multiple position hypotheses that are all consistent with recent relative measurements. Estimates of time-correlated biases in absolute positioning signals are extremely sensitive to these moments of ambiguity, as they are only observable through inconsistencies between absolute and relative sensor cues. Since the ambiguity is resolved by eliminating potential position hypotheses through vehicle motion and changing sensor viewpoints, it is critical to maintain as many consistent position hypotheses as possible until they can be eliminated. In the PosteriorPose algorithm, these are maintained in the set of particles through the use of measurement likelihoods and hypothesis tests that model data association errors.

The PosteriorPose algorithm's model-based approach to resolving position ambiguity allows correct estimation of time-correlated GPS biases. Figure 3.4 plots error magnitudes in the PosteriorPose algorithm's MMSE estimates of these biases, where the true time-correlated biases are determined by differencing the true position solution from the fused GPS / INS solution. It is evident that these biases are sensitive to sensor mistakes, as errors can fluctuate by 1 m

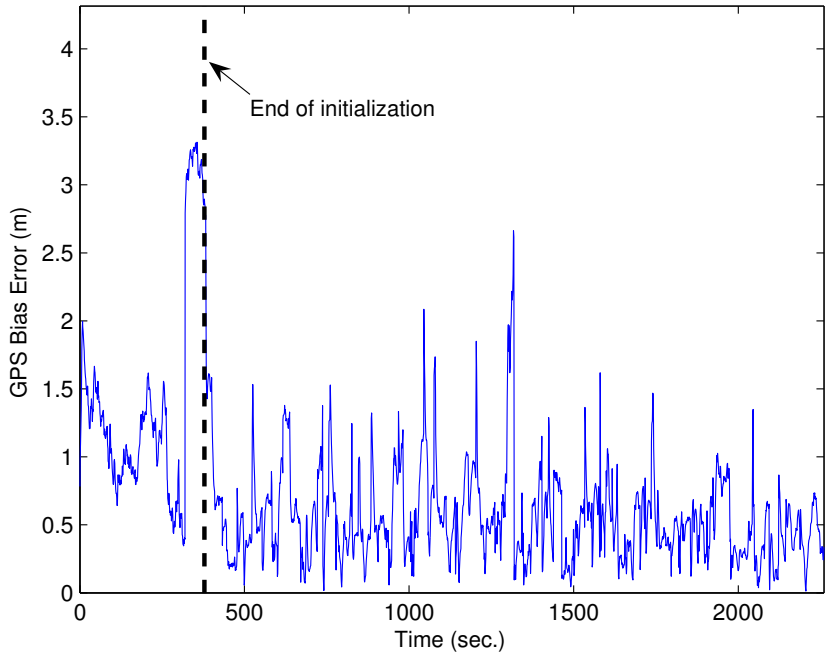

Figure 3.4: The PosteriorPose algorithm's model-based approach to resolving position ambiguity correctly estimates time-correlated GPS biases.

or more over 20 seconds. The primary example of this sensitivity is the minute of error near time $t \approx 320 - 380$ sec., where most of the positioning error shown in Figure 3.2 is due to sensor mistakes corrupting MMSE estimates of GPS biases during initialization. Despite the sensitivity, inclusion of these biases in the model for absolute position sensors yields significant reduction in positioning errors.

In the second experiment run on the first driving course, the PosteriorPose algorithm is run on the logged sensor data in an artificial GPS blackout. Skynet's fused GPS / INS system is given full GPS data for the 5 minute initialization period, then it is denied all GPS measurements for the 30 minutes that Skynet is driving. During this extended GPS blackout, the fused GPS / INS system is

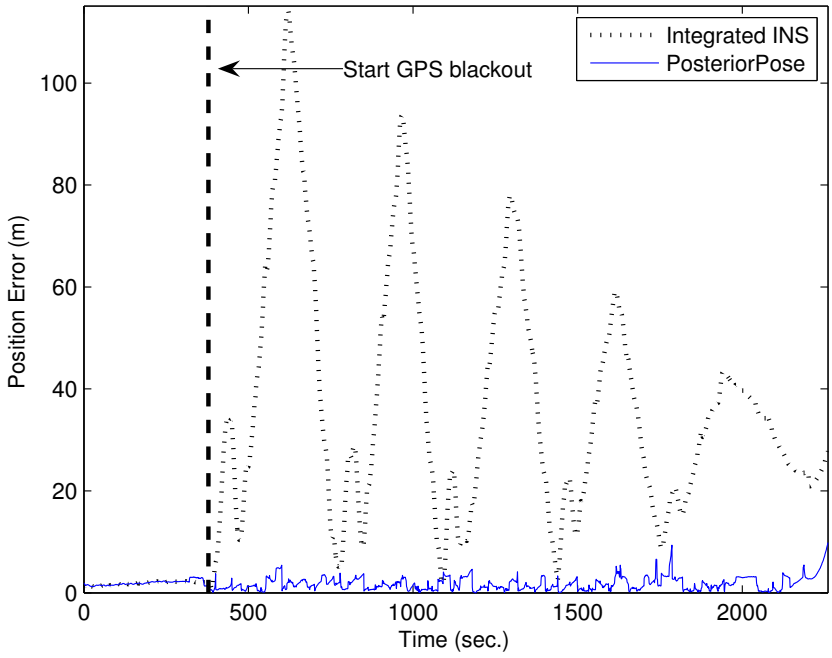

Figure 3.5: The PosteriorPose algorithm remains converged in a 30 minute extended GPS blackout using relative sensor cues. In contrast, the integrated INS solution drifts due to numerical integration errors.

effectively reduced to a dead reckoning INS system. The PosteriorPose algorithm is also run under the same blackout conditions: it uses the fused GPS / INS solution to initialize while Skynet is stationary, then it uses only Skynet's integrated INS and relative sensor cues for the 30 minutes that Skynet drives. The PosteriorPose algorithm is only run once in this configuration, estimating only Skynet's East position, North position, and heading. GPS biases are not estimated in this second experiment, as they are unobservable in a GPS blackout.

Figure 3.5 plots the errors of the PosteriorPose algorithm in the 30 minute extended GPS blackout. For comparison, the integrated INS solution under the same GPS blackout is also plotted. The integrated INS solution experiences large position errors during the blackout, since there are no absolute position

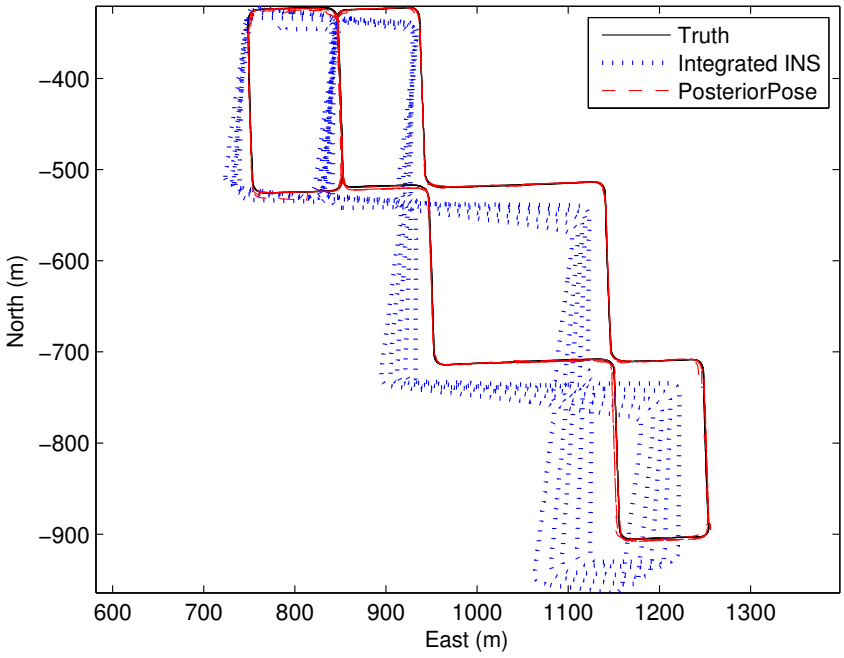

Figure 3.6: Skynet's ground track during a 30 minute extended GPS blackout according to the truth data, the PosteriorPose algorithm, and the integrated INS solution. The PosteriorPose algorithm is able to use relative sensor cues to eliminate drift in the INS solution caused by numerical integration errors.

measurements to correct errors in its numerical integrations. In this experiment, the integrated INS experiences up to 114 m errors. These errors are worst when Skynet is farthest from its initial position and orientation, as the integrated INS solution receives no information to correct drift stemming from small errors formed during initialization. The errors reach local minima each time Skynet completes a loop, where it nearly returns to its initial configuration. As the integrated INS loses information over time, these errors continue to grow unbounded in the blackout. Unlike the integrated INS solution, the PosteriorPose algorithm maintains an accurate position solution throughout the entire 30 minute extended GPS blackout. The algorithm's performance is in fact largely

unaffected by the lack of absolute positioning information, as it achieves an average error of only 1.78 m over the entire experiment. Figure 3.6 shows Skynet's ground track according to the truth data, the PosteriorPose algorithm, and the integrated INS solution. The integrated INS solution experiences substantial drift due to errors in numerical integration, whereas the PosteriorPose algorithm is able to correct the drift by incorporating relative sensor information.

3.5.2 Course 2: Sparse Road Information

The second driving course is a loop approximately 2.0 km long consisting of fewer road features than the first course: 6 stop lines, 8 turns, 1.7 km of painted solid lines, and 0.3 km of painted dashed lines. As before, a data log is collected consisting of approximately 5 minutes of initialization data followed by 30 minutes of driving data. In this experiment, however, the initial 5 minutes of data is taken while Skynet drives a single loop around the course. Skynet's lane and stop line algorithms are activated at the end of this initialization period. A total of 9 loops around the course are completed in the time allotted: one during the 5 minute initialization period, and 8 more during the 30 minute driving period. The same two experiments are run on the second driving course as were run on the first: one experiment with full GPS availability, and one in a 30 minute GPS blackout.

In the first experiment, the PosteriorPose algorithm is compared to Skynet's fused GPS / INS solution under full GPS signal availability. Figure 3.7 plots time histories of magnitudes of position errors of PosteriorPose algorithm variants and of Skynet's fused GPS / INS solution without OmniSTAR HP differential

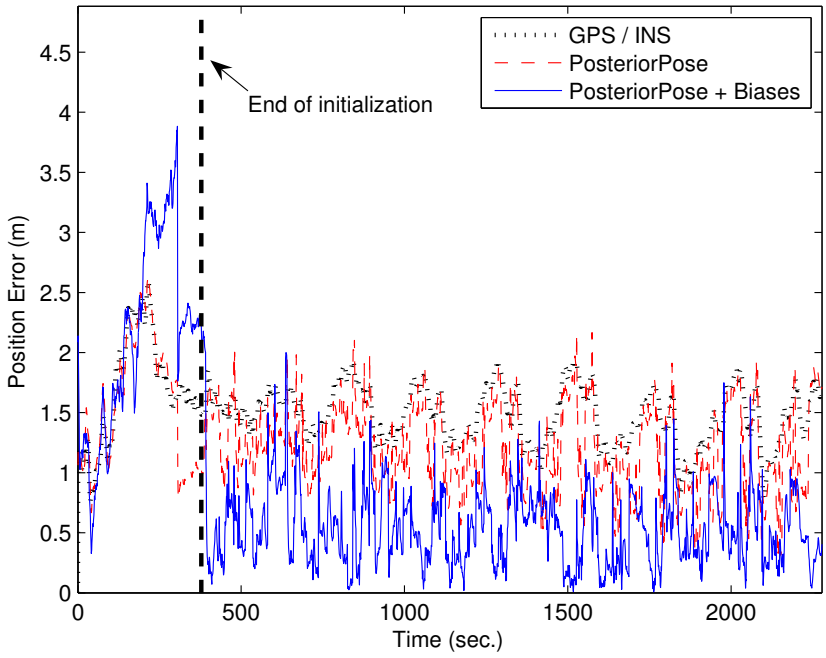

Figure 3.7: Variants of the PosteriorPose algorithm run with no GPS blackouts on a course with sparse road information. The PosteriorPose algorithm produces lower errors on average than the GPS / INS solution, and performance improvements are largely unaffected by the changes in the availability of road cues.

corrections.

As on the first driving course, the PosteriorPose algorithm yields lower position errors than the fused GPS / INS solution in many sampling intervals. These improvements amount to an average of 0.28 m less error when the PosteriorPose algorithm is not estimating GPS biases. At the 5% significance level, this corresponds to at least a 0.27 m reduction in positioning errors over the fused GPS / INS solution. Similarly, the PosteriorPose variant estimating GPS biases offers at least a 0.92 m reduction in positioning errors. Note these statistical conclusions are similar to those made on the first course, where road information is more abundant. This result suggests that the PosteriorPose algorithm's over-

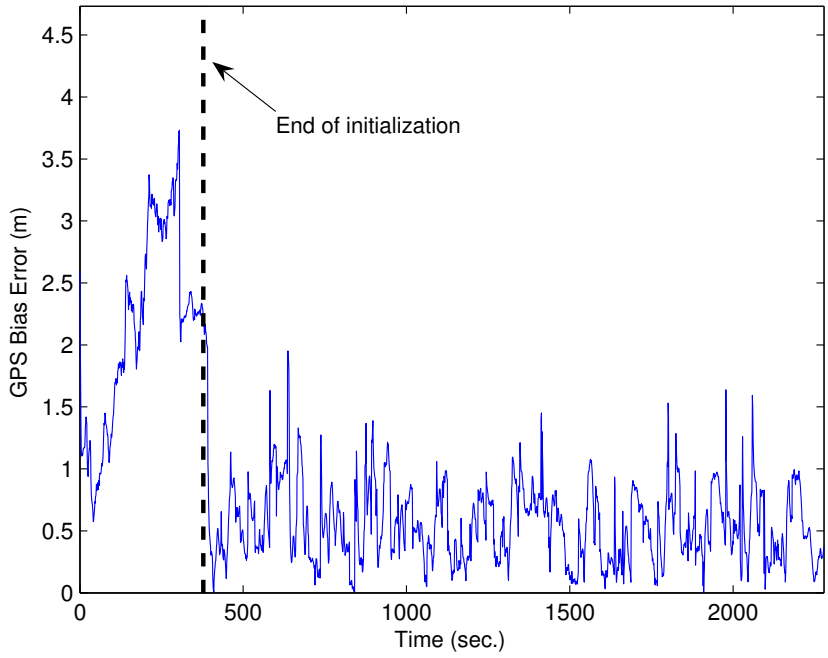

Figure 3.8: The PosteriorPose algorithm estimates GPS biases on a course with sparse road information. Errors in these biases are similar to those accrued on a course with dense road information.

all positioning improvements depend largely on the ability to extract weak and possibly infrequent information from relative positioning sensors without succumbing to their potentially large, non-Gaussian errors. The PosteriorPose algorithm achieves robustness not only by applying appropriate sensor likelihoods, but also by pairing those likelihoods with scrutinizing hypothesis tests to reject information that violates modeling assumptions. In short, the frequency of road information appears less important than its quality.

This conclusion is supported by Figure 3.8, which plots error magnitudes for the PosteriorPose algorithm's estimates of GPS bias. Despite stronger road information in the first course, GPS bias errors average 0.56 m on both courses. These errors are nearly indistinguishable; standard deviations of these errors are

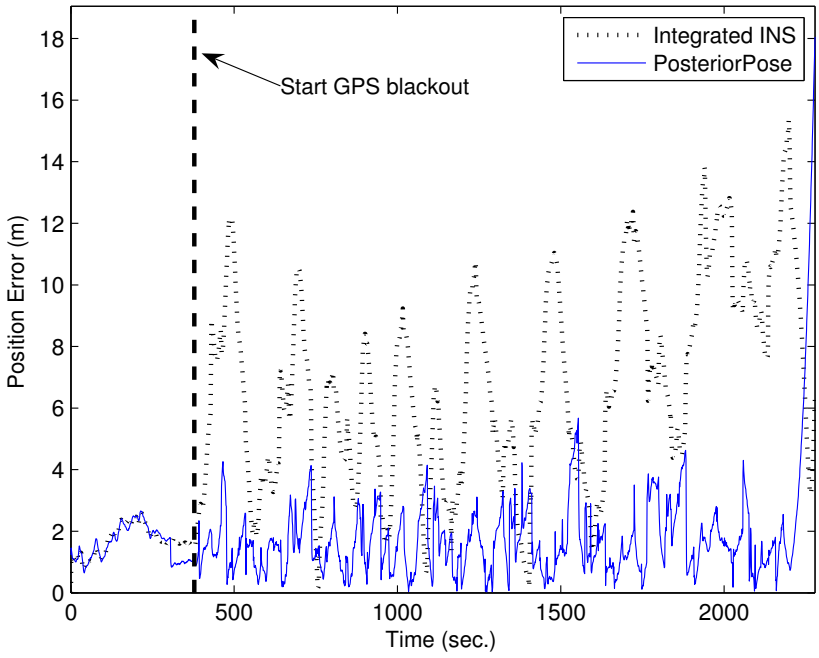

Figure 3.9: The PosteriorPose algorithm remains converged in a 30 minute extended GPS blackout despite sparse relative sensor cues. An integrated INS algorithm does not remain converged, even when initialized from moving vehicle data.

0.36 m on the first course and 0.33 m on the second. From this result it is evident that small changes in the availability of road information have little effect on the quality of the PosteriorPose positioning solution.

In the second experiment on the second driving course, the PosteriorPose algorithm is run on the logged sensor data in an artificial GPS blackout. As on the first course, Skynet's fused GPS / INS system is given full GPS data for the 5 minute initialization period, then it is denied all GPS measurements for the 30 minute evaluation period. The PosteriorPose algorithm is run under the same conditions. Note the differences between this and the previous 30 minute blackout: here, Skynet's fused GPS / INS system is initialized while moving,

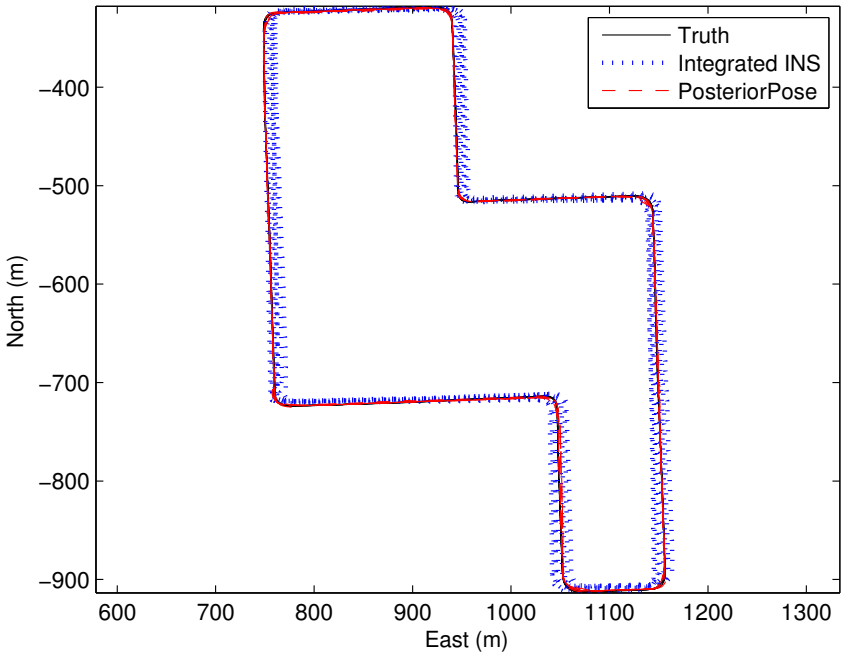

Figure 3.10: Skynet's ground track during a 30 minute extended GPS blackout on a course with sparse road information. The PosteriorPose algorithm is able to remain converged using road cues, even on a course with less stop lines and half the turns.

and the course driven contains fewer road features.

Figure 3.9 plots the errors of the PosteriorPose algorithm in the 30 minute extended GPS blackout on the second course. As on the first course, the integrated INS solution experiences large numerical integration errors that accumulate over time. Like the errors on the first course, these errors achieve local minima each time Skynet completes a full loop, where it passes close to its initial configuration. Note the errors are much smaller than on the first course, however, and achieve a maximum of only 15.35 m during the 30 minute blackout. This reduction in error is due to the fact that the integrated INS is initialized while Skynet is moving; the variety of data obtained from a moving vehicle

provides a more reliable calibration than data taken from a stationary vehicle in the previous blackout. Like the first blackout, the PosteriorPose algorithm achieves an average error of 1.84 m and is largely unaffected by the GPS blackout. Figure 3.10 plots Skynet's ground track according to the integrated INS and the PosteriorPose solutions.

3.6 Conclusion

The PosteriorPose algorithm has been presented as a particle filtering approach for augmenting absolute position measurements with relative landmark measurements compared against a known map. The algorithm has been developed specifically in the context of driving in an urban area, where buildings and trees commonly obstruct satellite-based navigation signals for extended periods of time. Techniques have been developed to correctly fuse absolute navigation signals with measurements of nearby lanes, lane lines, and stop lines in the Bayesian probabilistic framework. These techniques have been augmented with rigorous measurement hypothesis testing to cope with real-world sensor errors, especially non-Gaussian errors in sensors based on computer vision techniques.

The PosteriorPose algorithm has been implemented on Cornell University's autonomous Chevrolet Tahoe, 'Skynet,' where it fuses a GPS / INS solution with two lane detection algorithms and a stop line detection algorithm to provide an accurate vehicle navigation solution. The algorithm runs at 100 Hz with 2000 particles on a standard Windows computer using the real-time implementation techniques presented. The algorithm is shown to be statistically superior to the GPS / INS solution on its own, and insensitive to small variations in the

availability of road cues. Impressively, the PosteriorPose algorithm retains a converged and accurate position estimate during two 30 minute extended GPS blackouts, whereas the integrated INS solution suffers from large errors in dead reckoning. This result suggests that it is possible to use the PosteriorPose algorithm to navigate successfully in mapped urban canyons with only an initial position solution, accurate onboard inertial navigation, and onboard environment sensors.

CHAPTER 4

A MIXTURE-MODEL BASED ALGORITHM FOR REAL-TIME TERRAIN ESTIMATION [1]

4.1 Introduction

A problem of critical importance to modern robotics and remote sensing alike is mapping: the task of constructing a computational representation of an environment from available sensor data. These data-driven environment maps have a number of classical applications, including robot localization, exploration, target tracking, and planning [90]. Continued improvements in sensor accuracy and computational power have also spurred the use of automated mapping techniques for non-traditional applications, including aerial surveying, reconnaissance, and model generation [2], [66], [5], [22].

The mapping approaches adopted for these applications are as diverse as the applications themselves. In the robotics field, common approaches include tracking features or beacons, building belief maps or evidence grids of obstacles, building $2\frac{1}{2}$D elevation grids, and building Monte-Carlo sampled environments [90], [52], [70], [69], [41], [42]. These approaches are generally constructed in real-time for robotic navigation, and they make tradeoffs between the richness of their models and computational time. Beacon-based representations provide statistically rigorous map estimates, for example, but the maps are not dense. Occupancy grid approaches are dense, but they are fundamentally limited to binary obstacle identification. Monte-Carlo and elevation grid approaches gen-

[1] Reprinted with permission of John Wiley & Sons, Inc., from I. Miller and M. Campbell, "A Mixture-Model Based Algorithm for Real-Time Terrain Estimation," in *Journal of Field Robotics*, S. Singh (Ed.). Copyright © 2006 John Wiley & Sons, Inc.

erate dense maps, but they must either resample or perform heuristic interpolations to avoid computational bottlenecks. Initial research in representing robotic sensor data using true three dimensional terrain maps has recently been investigated [91], [34], [97]. These approaches either maintain large histories of sensor measurements or attempt to identify planar structures within their sensor data. They are currently limited to structured environments or constrained sensor geometries.

Mapping techniques associated with the remote sensing field, in contrast, are used to build digital elevation models for accurate surveying and geographical studies. These techniques often consist of a data collection phase with significant post processing to generate maps offline [8], [2], [45], [77]. Uncertainty in these digital elevation models is typically characterized in terms of errors sampled from a known set of reference survey points; therefore, this type of model fundamentally cannot be generated or maintained in real-time. Instead, these techniques are most commonly used for generating extremely precise digital elevation models for land surveying purposes.

While a unified mapping approach across these diverse applications is almost never adopted, an increasing interest in autonomous vehicle navigation and real-time reconnaissance suggests the merits of such an approach to a real-time terrain model. This paper presents one such possibility, using Gaussian sum representations of terrain uncertainty to generate and maintain an accurate and statistically rigorous real-time elevation model. The approach is unique in several regards. First, error transformation techniques are used to treat sensor measurements with a statistical model rather than as point clouds or inputs to an interpolation scheme. This allows the terrain estimation algorithm to handle

multiple sources of uncertainty during data collection rigorously. It also allows estimates of the errors in the terrain map to be generated on the fly rather than in post processing steps. In addition, the Gaussian sum representation allows the full terrain model to be built, stored, and maintained cheaply in real-time using a standard desktop computer. The specific case of generating an elevation model from a ground vehicle and laser rangefinders is presented, though the approach is general to all navigation problems such as airborne platforms and other sensors.

The work that follows derives the Gaussian sum algorithm for real-time terrain estimation along with experimental results obtained in a practical setup. Section 4.2 describes the terrain estimation problem and derives the Gaussian sum algorithm for terrain estimation, including steps for rigorous statistical analysis of terrain sensor measurements and assigning measurement locations within the terrain model. Section 4.3 gives a one-dimensional simulated example of the terrain estimation algorithm to describe its behavior. Section 4.4 presents experimental results of applying the algorithm on a full-sized ground vehicle operating at realistic speeds.

4.2 Terrain Estimation Algorithm

In representing terrain or digital elevation models, one common approach is to store terrain data as a 'Cartesian height map' or 'raster grid', where a region of the Earth's surface is parameterized by a location in the XY plane and an associated elevation relative to an origin of interest [69], [85]. This same parameterization is used to store elevation maps in this study. That is, it is assumed

there exists an origin of interest, described according to its latitude, longitude, and altitude (LLA) with respect to the WGS-84 ellipsoid model of the Earth [39]. The reference plane for this study is then calculated as an East-North-Up (ENU) plane tangent to this ellipsoid model at the origin of interest, where the X-axis points East along a line of latitude, the Y-axis points North along a line of longitude, and the Z-axis completes the coordinate frame. This reference plane is divided into N_c grid cells, with the j^{th} cell extending in the East direction from E_{j-} to E_{j+} and in the North direction from N_{j-} to N_{j+}. The goal is to develop an algorithm to estimate the elevations of each of these grid cells in real-time in the presence of multiple sources of noise. More specifically, the goal is to develop elevation estimates in real-time that are optimal in the sense of the minimum mean square error (MMSE).

The proposed terrain estimation algorithm has three separate steps to accomplish this goal. First, a statistical representation of each sensor measurement is formed, in order to account for multiple sources of sensing error in a probabilistically rigorous manner. Second, each sensor measurement is assigned or 'associated' to one or more grid cells to which it is likely to correspond. Finally, the measurements assigned to each grid cell are fused in real-time into an optimal elevation estimate for that grid cell. These three steps are discussed in turn below.

4.2.1 Statistical Treatment of Sensor Measurements

The first step of the terrain estimation algorithm is to form a statistical representation of each sensor measurement in order to account for all sources of sen-

sor error. These sensor errors are commonly due to four general sources: 1) errors due to the sensor itself, 2) errors due to uncertainty in the sensor's orientation and location on the sensing platform, 3) errors due to uncertainty in the orientation of the sensing platform itself, and 4) errors due to uncertainty in the location of the sensing platform [36]. The first type of error, due to the sensor itself, describes the sensor's accuracy. This type of error is a function of the method by which the sensor makes measurements, and it is generally independent of the sensor's orientation. The second type of error, sensor orientation and location error, arises because the terrain map is often not built in a sensor-centric coordinate frame. As a result, the sensor measurements must be transformed to other coordinate frames, and these transformations may introduce errors. For rigidly-mounted sensors, these errors can be approximately eliminated or reduced to inconsequential values with offline calibration against objects of known location. For actuated sensors or sensors subject to platform vibration, however, these errors must be considered as statistical quantities. The third and fourth types of errors are due to imperfect knowledge of the orientation and position of the sensing platform. For moving platforms, these errors may be reported by an inertial navigation system or another position estimation scheme. The statistical contributions of platform orientation and location errors affect all sensors on the platform in an identical manner.

In order to understand in a statistical sense how these four sources of error affect each sensor measurement, it is first necessary to transform each sensor measurement and its uncertainties into a common coordinate frame for terrain estimation. To begin, each raw sensor measurement $\underline{r}$ is expressed in the fixed ENU coordinate frame of the terrain map:

$$\underline{r}^{ENU} = \begin{pmatrix} E \\ N \\ U \\ 1 \end{pmatrix} = f\left(\underline{p}, \underline{r}\right) \tag{4.1}$$

where $\underline{r}$ is the raw sensor measurement, $\underline{p}$ is the set of parameters that describe the sensor's orientation with respect to the ENU map frame, and $f(\cdot)$ is the function that transforms the raw measurement into the ENU frame. Note that $\underline{r}^{ENU}$ is expressed as a four-element vector. The fourth element of each measurement, always 1, permits the use of 4×4 rotation and translation matrices to express the transformation function $f(\cdot)$ as a series of matrix multiplications [62], [64]. Although other representations of the transformation may be used, the sequential matrix representation will be shown to be particularly useful for real-time implementations.

Each transformed measurement $\underline{r}^{ENU}$ is a terrain detection generated from a single raw measurement. Each terrain detection gives a measurement of the elevation and location of a small patch of terrain near the sensing platform. These elevation measurements are built up from both the original raw sensor measurement $\underline{r}$ and the sensor's orientation parameters $\underline{p}$ at the time the measurement was produced. As discussed previously, the orientation parameters $\underline{p}$ and raw measurement $\underline{r}$ are uncertain; they are more accurately modeled using estimates of their true values with associated errors:

$$\begin{aligned} \underline{p} &= \underline{\hat{p}} + \delta\underline{p} \\ \underline{r} &= \underline{\hat{r}} + \delta\underline{r} \end{aligned} \tag{4.2}$$

where $\underline{p}$ and $\underline{r}$ are the true orientation parameters and noise free measurement, $\underline{\hat{p}}$ and $\underline{\hat{r}}$ are the values of the parameters reported by the sensors or state estima-

tors, and $\delta\underline{p}$ and $\delta\underline{r}$ are the errors in those reported values. Under this formulation, the values $\delta\underline{p}$ and $\delta\underline{r}$ can be due to any of the four error sources discussed in the beginning of section 4.2. It is important to note that the values $\underline{p}$ and $\underline{r}$ are not known perfectly unless there are no sources of error in any aspect of the sensing system. In general, only $\hat{\underline{p}}$ and $\hat{\underline{r}}$ are known.

To form a statistical representation taking into account all errors $\delta\underline{p}$ and $\delta\underline{r}$, equation 4.2 is substituted into equation 4.1. This expresses the true measurement in terms of the available sensor and estimator outputs:

$$\underline{r}^{ENU} = f\left(\hat{\underline{p}} + \delta\underline{p}, \hat{\underline{r}} + \delta\underline{r}\right) \tag{4.3}$$

Notice that equation 4.3 accounts for any potential sources of error arising either from the sensor or from the transformation to the ENU frame. If the elements of $\underline{p}$ and $\underline{r}$ fully describe the transformation from sensor measurements to terrain detections, equation 4.3 takes into account all sources of error.

To continue, a Taylor expansion of equation 4.3 is made about the observed measurement and orientation parameters. Then, assuming the estimation errors are small, the expansion is truncated at first order to make the expression more tractable:

$$\begin{aligned} \underline{r}^{ENU} &\approx f\left(\hat{\underline{p}}, \hat{\underline{r}}\right) + \left.\frac{\partial f}{\partial \underline{p}}\right|_{\underline{p}=\hat{\underline{p}}, \underline{r}=\hat{\underline{r}}} \delta\underline{p} + \left.\frac{\partial f}{\partial \underline{r}}\right|_{\underline{p}=\hat{\underline{p}}, \underline{r}=\hat{\underline{r}}} \delta\underline{r} \\ &= f\left(\hat{\underline{p}}, \hat{\underline{r}}\right) + J_{\underline{p}}\left(\hat{\underline{p}}, \hat{\underline{r}}\right)\delta\underline{p} + J_{\underline{r}}\left(\hat{\underline{p}}, \hat{\underline{r}}\right)\delta\underline{r} \end{aligned} \tag{4.4}$$

where $J_{\underline{p}}(\cdot)$ and $J_{\underline{r}}(\cdot)$ are the Jacobians of the transformation function $f(\cdot)$ with respect to the sensor orientation parameters and raw measurement, respectively. Additionally, assume the parameter estimates $\hat{\underline{p}}$ and raw measurement $\hat{\underline{r}}$ are both unbiased and conditioned upon all available orientation and sensor

information $\mathcal{I}$. Under these assumptions, a posterior estimate of the terrain detection may be formed by taking the expectation of equation 4.4 conditioned on all available information [11]:

$$\underline{\hat{r}}^{ENU} \equiv \begin{pmatrix} \hat{e} \\ \hat{n} \\ \hat{u} \\ 1 \end{pmatrix} = E\left[\underline{r}^{ENU} \middle| \mathcal{I}\right] \approx f\left(\underline{\hat{p}}, \underline{\hat{r}}\right) \tag{4.5}$$

Note that if some elements in the parameter estimates $\underline{\hat{p}}$ and $\underline{\hat{r}}$ are biased, the biases enter into the value of $\underline{\hat{r}}^{ENU}$ when taking the expectation of equation 4.4. In general, however, such biases can be removed through more refined calibration procedures. After such procedures the residual measurement errors will have a bias that is statistically indistinguishable from zero.

Continuing with the error analysis, the mean square error (MSE) matrix of the posterior measurement estimate yields the desired statistical measure of the measurement's uncertainty. Note this uncertainty takes into account all sources of error contained in the estimates $\underline{\hat{p}}$ and $\underline{\hat{r}}$:

$$\begin{aligned} P_{\underline{\hat{r}}} &= E\left[(\underline{r}^{ENU} - \underline{\hat{r}}^{ENU})(\underline{r}^{ENU} - \underline{\hat{r}}^{ENU})^T \middle| \mathcal{I}\right] \\ &\approx J_{\underline{p}}\left(\underline{\hat{p}}, \underline{\hat{r}}\right) E\left[\delta\underline{p} \cdot \delta\underline{p}^T \middle| \mathcal{I}\right] J_{\underline{p}}^T\left(\underline{\hat{p}}, \underline{\hat{r}}\right) + J_{\underline{r}}\left(\underline{\hat{p}}, \underline{\hat{r}}\right) E\left[\delta\underline{r} \cdot \delta\underline{r}^T \middle| \mathcal{I}\right] J_{\underline{r}}^T\left(\underline{\hat{p}}, \underline{\hat{r}}\right) \\ &= J_{\underline{p}}\left(\underline{\hat{p}}, \underline{\hat{r}}\right) Q_{\underline{p}} J_{\underline{p}}^T\left(\underline{\hat{p}}, \underline{\hat{r}}\right) + J_{\underline{r}}\left(\underline{\hat{p}}, \underline{\hat{r}}\right) Q_{\underline{r}} J_{\underline{r}}^T\left(\underline{\hat{p}}, \underline{\hat{r}}\right) \end{aligned} \tag{4.6}$$

where $Q_{\underline{p}}$ and $Q_{\underline{r}}$ are the mean square error matrices for the estimators used in determining $\underline{\hat{p}}$ and $\underline{\hat{r}}$. This representation includes the statistical effects of the orientation and sensor errors up through their second moments, mapped through a linear approximation of the actual transformation. These approximate linearized statistical techniques are common and work well in nonlinear

estimation problems [11]. Techniques that preserve higher order statistical effects through nonlinear transformations, such as the Unscented transform or Monte Carlo methods, could also be used [37], [7]. These are ignored in the present study, however, due to higher computational costs.

Notice the structure of equation 4.6 assumes the sensor orientation errors and raw measurement errors are uncorrelated. If that is not the case, it is straightforward to account for cross correlation by creating a stacked vector $\underline{s}$ of all uncertain parameters and their errors:

$$\underline{s} = \begin{pmatrix} \underline{p} \\ \underline{r} \end{pmatrix} = \begin{pmatrix} \underline{\hat{p}} + \delta\underline{p} \\ \underline{\hat{r}} + \delta\underline{r} \end{pmatrix} = \underline{\hat{s}} + \delta\underline{s} \tag{4.7}$$

The MSE matrix $P_{\underline{\hat{r}}}$ is now defined as:

$$P_{\underline{\hat{r}}} = J_{\underline{s}}\left(\underline{\hat{s}}\right) Q_{\underline{s}} J_{\underline{s}}^{T}\left(\underline{\hat{s}}\right) \tag{4.8}$$

where $Q_{\underline{s}}$ is a covariance matrix with $Q_{\underline{p}}$ and $Q_{\underline{r}}$ as its diagonal blocks and any cross correlations between $\delta\underline{p}$ and $\delta\underline{r}$ as its off-diagonal blocks.

Two additional comments about this statistical representation are in order. First, the analysis is only affected by elements of $\underline{p}$ and $\underline{r}$ that are uncertain. Any known elements in these vectors have no statistical variance and no correlation with any other elements of $\underline{p}$ or $\underline{r}$, so the rows of the Jacobian matrices corresponding to any known parameters have no effect on $P_{\underline{\hat{r}}}$. Second, the analysis is also independent of the lengths of $\underline{p}$ and $\underline{r}$; the MSE matrix $P_{\underline{\hat{r}}}$ is always 4×4. It is noted, however, that the MSE matrix $P_{\underline{\hat{r}}}$ will not be full rank unless the Jacobian matrix and covariance matrix that comprise it both have a rank of at least

4.

With the posterior measurement estimate $\underline{\hat{r}}^{ENU}$ defined in equation 4.5, equation 4.6 gives its 4×4 mean square error matrix $P_{\underline{\hat{r}}}$. This matrix describes the relative size and correlation of the terrain measurement errors in the elements of $\underline{\hat{r}}^{ENU}$ and behaves as a covariance matrix. The useful partition of the matrix $P_{\underline{\hat{r}}}$ is the upper-left 3×3 block P^{ENU}, which contains the ENU mean square errors and cross correlations for the measurement $\underline{\hat{r}}^{ENU}$. The remaining elements of the matrix, which involve correlations with the always-unity fourth term of $\underline{\hat{r}}^{ENU}$, are not used. That is,

$$P^{ENU} = \begin{pmatrix} I_{3\times3} & 0_{3\times1} \end{pmatrix} \cdot P_{\underline{\hat{r}}} \cdot \begin{pmatrix} I_{3\times3} \\ 0_{1\times3} \end{pmatrix} \tag{4.9}$$

Equations 4.5 and 4.9 model each measurement as a nominal terrain measurement $\underline{\hat{r}}^{ENU}$ with uncertainty characterized by ellipsoidal equiprobability surfaces. This strategy is itself an atypical representation of sensor measurements in remote sensing applications, where they are typically modeled only as discrete points with no associated error [85]. Horizontal and vertical errors are instead computed in post-processing validation against truth models [36]. In addition, errors in sensor orientation or sensor platform position and orientation are commonly treated as constants using offline calibration techniques [63].

More recently, Ref. [85] suggested representing each sensor measurement as a voxel, a three-dimensional pixel with volume characterized by its error along three directional axes. However, this representation cannot support cross-correlation between coordinate axes, so it must necessarily make conservative error estimates to capture uncertainty. In contrast, the benefit of the statistical

representation of equations 4.5 and 4.9 over typical representations is that it is dynamical: the error in each measurement is estimated in real-time based on the current accuracy of all the systems used to generate $\underline{\hat{p}}$ and $\underline{\hat{r}}$.

4.2.2 Measurement Association

The second step of the terrain estimation algorithm is to use the statistical analysis of section 4.2.1 to assign each terrain measurement to one or more grid cells from which it is likely to have originated. This problem of determining which measurements belong in each cell arises only because the full uncertainty in the measurements is considered during processing. That is, measurement correspondence on a cell-by-cell basis is uncertain due to the uncertainty in each measurement's in-plane location as well as its elevation. This problem is similar to the problem of association in target tracking literature such as [10] and [40]. The technique used here to assign measurements to cells is similar to those discussed in [10] for assigning measurements to a single target. The present application is significantly different from traditional target tracking, however, because all measurements correspond to some portion of the fixed terrain map.

For this step of the terrain estimation algorithm, it is assumed that sensor measurement data has been processed according to equations 4.5, 4.6, and 4.9 to generate a set of N measurement estimates $\underline{\hat{r}}_i^{ENU} = (\,\hat{e}_i \quad \hat{n}_i \quad \hat{u}_i\,)^T$ and their associated MSE matrices P_i^{ENU} for $i \in [1, \ldots, N]$. To begin, this information is used to calculate the probability that the i^{th} measurement $\underline{r}_i^{ENU}$ belongs in a cell of interest. Assuming the ENU probability distribution $\mathcal{P}_i\,(E, N, U)$ of the measurement is known, the probability can be evaluated explicitly by integrating

over the area covered by the cell. First, the vertical coordinate is integrated out to yield the in-plane marginal distribution:

$$\mathcal{P}_i\left(E,N\right)=\int_{-\infty}^{\infty}\mathcal{P}_i\left(E,N,U\right)dU \tag{4.10}$$

This in-plane distribution of the measurement can then be integrated over the area of the j^{th} cell to yield the probability that the measurement corresponds to that cell:

$$P\left(\underline{\hat{r}}_i^{ENU}\in j\right)=\int_{E_{j-}}^{E_{j+}}\int_{N_{j-}}^{N_{j+}}\mathcal{P}_i\left(E,N\right)dNdE \tag{4.11}$$

where the j^{th} cell is defined by the rectangle $E_{j-}\leq E\leq E_{j+}$ and $N_{j-}\leq N\leq N_{j+}$. To complete this integral, the posterior measurement $\underline{\hat{r}}_i^{ENU}$ and its associated MSE matrix P_i^{ENU} are used to approximate the joint ENU distribution as a multivariate Gaussian:

$$\mathcal{P}_i\left(E,N,U\right)\approx\mathcal{N}\left(\underline{\hat{r}}_i^{ENU},P_i^{ENU}\right) \tag{4.12}$$

This distribution is exact for the case of linear transformations, Gaussian noise sources, and unbiased estimates of the sensor orientation parameters and raw measurements. The Gaussian approximation is also commonly made for non-linear, non-Gaussian cases using either the linearization presented above or the first and second moments implied by the Unscented Transform [37]. Applying the Gaussian transform, the EN joint probability distribution can be written in closed form:

$$\mathcal{P}_i\left(E,N\right)=\mathcal{N}\left(\hat{\underline{r}}_i^{EN},P_i^{EN}\right) \tag{4.13}$$

where $\hat{\underline{r}}_i^{EN}$ is the first two components of the measurement estimate $\hat{\underline{r}}_i^{ENU}$ and P_i^{EN} is the upper left 2×2 EN block of the MSE matrix:

$$P_i^{ENU}=\begin{pmatrix} P_i^{EN} & P_i^{EN,U} \\ P_i^{U,EN} & P_i^{U} \end{pmatrix} \tag{4.14}$$

Using the given measurement estimate and its MSE matrix, the integral of equation 4.11 can now be computed. For real-time computational purposes, this integral is approximated as a single Riemann square [51]:

$$p_i\equiv P\left(\hat{\underline{r}}_i^{EN}\in j\right)\approx\left(E_{j+}-E_{j-}\right)\left(N_{j+}-N_{j-}\right)\cdot\mathcal{P}_i\left(\frac{1}{2}(E_{j+}+E_{j-}),\frac{1}{2}(N_{j+}+N_{j-})\right) \tag{4.15}$$

which gives the approximate probability p_i that the measurement belongs to the j^{th} cell. Equation 4.15 can then be used with equation 4.13 to approximate the probability p_i that a measurement estimate $\hat{\underline{r}}_i^{ENU}$ corresponds to a particular cell.

4.2.3 In-cell Terrain Measurement Fusion

The final step of the terrain estimation algorithm is to fuse all the terrain measurements into an optimal terrain estimate. Because the measurements are assigned to grid cells according to the probability that they belong to those cells, this task is equivalent to determining the distribution of elevations in each cell

given all measurements assigned to it. There are, however, two competing objectives in this task. First, a more accurate representation of the elevation distribution creates a better estimate within each cell. With this objective, ideally all individual measurements are retained separately to preserve the full set of data reported by the sensors. The tradeoff with this type of accurate representation is a second competing objective: computational feasibility. Memory and computational requirements scale worst case as $O(N_c \cdot N)$ with N_c the number of cells and N the number of individual measurements retained, so it very quickly becomes infeasible to retain all measurements separately for any reasonably-sized terrain grid. As a result, a more compact representation is desired, one that yields useful terrain information without sacrificing computational feasibility.

To develop this computationally tractable, real-time representation, recall that the original measurement estimates $\hat{\underline{r}}_i^{ENU}$ are uncertain in three axes: East, North, and Up. What is desired in the adopted grid representation, however, is the univariate distribution of elevations in a particular cell at a particular East and North location. This knowledge can be used to form a posterior univariate elevation estimate $\hat{U}_i = E\left[\hat{u}_i | E, N\right]$ for the i^{th} measurement by conditioning on the East and North location of the j^{th} cell [11]:

$$\hat{U}_i = \hat{u}_i + P_i^{U,EN} \left(P_i^{EN}\right)^{-1} \cdot \left[\frac{1}{2}\begin{pmatrix} E_{j+} + E_{j-} \\ N_{j+} + N_{j-} \end{pmatrix} - \begin{pmatrix} \hat{e}_i \\ \hat{n}_i \end{pmatrix}\right] \tag{4.16}$$

where $\begin{pmatrix} E & N \end{pmatrix} = \frac{1}{2}\begin{pmatrix} E_{j+} + E_{j-} & N_{j+} + N_{j-} \end{pmatrix}$ is the center of the j^{th} cell. This estimate $\hat{U}_i$ has conditional variance:

$$\sigma^2_{\hat{U}_i} = P_i^U - P_i^{U,EN} \left(P_i^{EN}\right)^{-1} P_i^{ENU} \tag{4.17}$$

In forming this elevation estimate $\hat{U}_i$ and its variance $\sigma^2_{\hat{U}_i}$ for association to the j^{th} cell, each measurement is assumed to originate from the center of that cell, $\frac{1}{2}\begin{pmatrix} E_{j+} + E_{j-} \\ N_{j+} + N_{j-} \end{pmatrix}$, as per equation 4.16. This approximation approaches the exact continuous-terrain solution as cell size decreases, presenting a tradeoff between terrain map resolution and computational feasibility. Experimentally, it is found that cell size is commensurate to the smallest terrain features that can be detected.

Each of the measurements $\hat{U}_i$ is effectively a terrain detection: a measurement of a piece of the terrain within a particular cell. It is convenient to assume that each of these measurements corresponds to a different patch of terrain within the cell, so that no two measurements occur at precisely the same location. This assumption is justified by the fact that the terrain is continuous, the location of the terrain sensed by each measurement is uncertain, and often the sensing platform is moving. Two other assumptions are also made. First, each terrain measurement is assumed equally likely; that is, there is no *a priori* terrain information. This uniform, uninformative prior is adopted for convenience, as the uniform prior acts as a constant removed by normalization in the final distribution. Second, it is assumed that each cell has one correct or 'dominant' elevation to be estimated, and that elevation is represented within the set of terrain measurements obtained.

Several conclusions follow from these assumptions. First, the likelihood of each elevation measurement $\hat{U}_i$ is dependent only on the set of sensor orientation parameters $\underline{\hat{p}}_i$ and raw sensor measurement $\underline{\hat{r}}_i$ used to generate it. In particular, the likelihood of each elevation estimate conditioned on all measurements made so far is just equal to the likelihood of that elevation estimate

conditioned on only the parameters used to generate it. Second, the likelihood of each measurement being a measure of the correct elevation for a particular cell is equal to the probability that the measurement came from that cell, i.e. its association probability p_i. Third, because the *a posteriori* distributions of the elevation measurements are all Gaussian, the elevation distribution within the j^{th} cell $\mathcal{P}\left(U_j|\underline{\hat{p}}_{1\ldots M}, \underline{\hat{r}}_{1\ldots M}\right)$ conditioned on the full set of measurements $\underline{\hat{p}}_{1\ldots M} = \left\{\underline{\hat{p}}_1, \underline{\hat{p}}_2, \ldots, \underline{\hat{p}}_M\right\}$ and $\underline{\hat{r}}_{1\ldots M} = \{\underline{\hat{r}}_1, \underline{\hat{r}}_2, \ldots, \underline{\hat{r}}_M\}$ is a Gaussian sum or 'Gaussian mixture' constructed from the elevation estimates [11]:

$$\mathcal{P}(U_j|\underline{\hat{p}}_{1\ldots M}, \underline{\hat{r}}_{1\ldots M}) = \frac{\sum_{i=1}^{M} p_i \mathcal{N}\left(\hat{U}_i, \sigma^2_{\hat{U}_i}\right)}{\sum_{i=1}^{M} p_i} \tag{4.18}$$

where $\hat{U}_i$ from equation 4.16 is the i^{th} elevation measurement estimate, $\sigma^2_{\hat{U}_i}$ from equation 4.17 is its associated conditional variance, p_i from equation 4.15 is the probability that the elevation measurement belongs in the cell, and M is the number of measurements assigned to the cell.

Equation 4.18 represents a desired data driven elevation distribution in the cell which takes into account all sources of uncertainty present in the system. However, the model is not computationally feasible, because the Gaussian mixture stored in each cell grows with the number of sensor measurements assigned to that cell. For real-time terrain estimation, a smaller set of information about each cell is desired. This set of data must be descriptive enough to exceed raw measurements in usefulness but small enough to be computationally feasible. The mean and variance of the elevation distribution satisfy both these requirements. Taking the expected value of the elevation distribution yields an approximate MMSE estimate of the characteristic or 'dominant' elevation of the j^{th} cell:

$$\hat{U}_{GM,j} = \frac{\sum_{i=1}^{M} p_i \hat{U}_i}{\sum_{i=1}^{M} p_i} \approx E\left[U_j | \underline{\hat{p}}_{1\ldots M}, \underline{\hat{r}}_{1\ldots M}\right] \tag{4.19}$$

by the linearity of the expectation operator. Similarly, the second central moment of the elevation distribution gives the mean square error of the elevation estimate within the j^{th} cell [11]:

$$\begin{aligned} \sigma^2_{GM,j} &= E\left[(U_j - \hat{U}_{GM,j})^2 | \underline{\hat{p}}_{1\ldots M}, \underline{\hat{r}}_{1\ldots M}\right] \\ &= E\left[U_j^2 | \underline{\hat{p}}_{1\ldots M}, \underline{\hat{r}}_{1\ldots M}\right] - \hat{U}^2_{GM,j} \\ &= \frac{\sum_{i=1}^{M} p_i (\hat{U}_i^2 + \sigma^2_{\hat{U}_i})}{\sum_{i=1}^{M} p_i} - \hat{U}^2_{GM,j} \end{aligned} \tag{4.20}$$

Equations 4.19 and 4.20 give the first two moments of the elevation distribution of the j^{th} cell. Physically, equation 4.19 is an estimate of the characteristic elevation of the cell. Mathematically, it is an approximate MMSE estimate of the elevation of the j^{th} given the assumptions discussed above. Equation 4.20 may be interpreted as a measure of the roughness or spread of elevations within the cell, though it also stores information about the confidence of the mean elevation estimate. The estimates of cell mean and variance can be used to make statistical statements about the elevations in the cell, taking into account the noise present in the entire system.

4.2.4 Algorithm Benefits and Real-Time Implementation

The proposed statistical representation of the terrain has several unique advantages over more traditional mapping strategies. First, the variance estimate within each cell gives information about the spread of elevations likely to be encountered within that cell, enabling real-time statements of elevation confidence

intervals. These confidence intervals arise naturally as part of the estimation process, because the terrain estimation algorithm includes error estimates. The error estimates thus represent a richer set of information and physical meaning than standard binary obstacle detection algorithms without requiring the additional post processing of typical surveying representations.

A second advantage of this terrain model is that it can be generated and maintained in real-time. Recall from sections 4.2.2 and 4.2.3 that each measurement estimate is assigned to each cell according to the probability that the measurement lies in that cell. For practical implementations with finite numerical precision, however, only cells near the nominal in-plane measurement location derived from $\hat{\underline{r}}_i^{ENU}$ are affected by that measurement estimate. As a result, each measurement estimate need only be applied to cells in a small neighborhood of the nominal measurement. To counteract this issue, measurements can also be thresholded based on the probability that they belong to a particular cell to provide further reduction in computational complexity. These steps place limits on the number of cells to which each measurement can be applied, so the computational complexity is reduced from $O(C \cdot N)$ spent applying N raw measurements to all C cells in the entire terrain map, to $O(k \cdot N)$ spent applying each measurement to a maximum of k terrain cells. Furthermore, if only the first two moments of the elevation distribution are desired, then each measurement can easily be fused with previous measurements. In fact, only the following four quantities are required for each cell:

$$\sum_{i=1}^{M} p_i, \qquad \sum_{i=1}^{M} p_i \hat{U}_i, \qquad \sum_{i=1}^{M} p_i \hat{U}_i^2, \qquad \sum_{i=1}^{M} p_i \sigma_{\hat{U}_i}^2$$

where each of these quantities is itself a scalar. Also, because fusing a new measurement with the measurement history only requires knowledge of these four

variables, the computational complexity and memory requirements of maintaining each cell are $O(1)$. This makes it possible to fuse measurements from many sensors at once in a distributed multithreaded architecture built from standard desktop computers, all while maintaining the terrain map in real-time. Finally, it is important to note that once sensor measurements have been used to update the appropriate cells in the terrain model, the original measurements can be discarded. In other words, the entire terrain map can be maintained without storing any measurement history. This makes it possible to maintain a full terrain map over many square miles of terrain in memory on a standard desktop computer.

Further computational savings can also be made in the measurement and MSE transformations of equations 4.5 and 4.6. In general, both transformations require a large number of multiplications and trigonometric evaluations. However, the transformation function $f(\cdot)$ from equation 4.1 can be written as a series of matrix multiplications, as suggested above:

$$f\left(\underline{\hat{p}}, \underline{\hat{r}}\right) = T_n\left(\underline{\hat{p}}[n]\right) \cdot \ldots \cdot T_2\left(\underline{\hat{p}}[2]\right) \cdot T_1\left(\underline{\hat{p}}[1]\right) \cdot R\left(\underline{\hat{r}}\right) \tag{4.21}$$

where $\underline{\hat{p}}[l]$ is the l^{th} element of $\underline{\hat{p}}$, $T(\cdot)$ is a 4×4 transformation matrix, and $R\left(\underline{\hat{r}}\right)$ is a matrix representation of the raw sensor measurement $\underline{\hat{r}}$. This matrix-based representation is useful because each transformation matrix $T_i(\cdot)$ is a function of only one orientation parameter. Many of these orientation parameters are common to more than one measurement or constant in time. As a result, many of the matrix multiplications can be precomputed once and cached for future measurements.

The Jacobian calculations of equation 4.6 can also be computed efficiently by utilizing the structure of equation 4.21. In general calculating the Jacobians $J_{\underline{p}}(\cdot)$ and $J_{\underline{r}}(\cdot)$ of the transformation function require repeated application of the chain rule. However, because each independent orientation parameter appears in only one matrix in equation 4.21, the chain rule is not necessary. With this representation, each element of the Jacobian matrices is reduced to matrix multiplications and the differentiation of a single matrix. The matrix multiplications can be cached in the manner described above to reduce the expense of calculating the Jacobians. In essence, the algorithm benefits from the fact that many measurements share all but one or two orientation parameters.

4.3 A Simple Example

The behavior and mechanics of the terrain estimation algorithm presented in section 4.2 are best illustrated with a simple example. Consider a cart moving forward at constant speed of 5 m/sec along one dimension. The cart is equipped with a typical rangefinder that returns noisy ranges between it and the terrain below. The rangefinder is mounted on a platform elevated 1.5 m above the bottom of the vehicle wheels. Furthermore, the rangefinder is pitched forward by an approximate angle of 5° with respect to the horizontal. That is, the rangefinder is angled down to scan along the ground as the vehicle moves forward. Figure 4.1 shows the hypothetical setup for this one-dimensional simulated vehicle.

The rangefinder is modeled after a SICK LMS 291 laser rangefinder or 'LIDAR' [84]. In this simple one-dimensional example, the sensor returns one

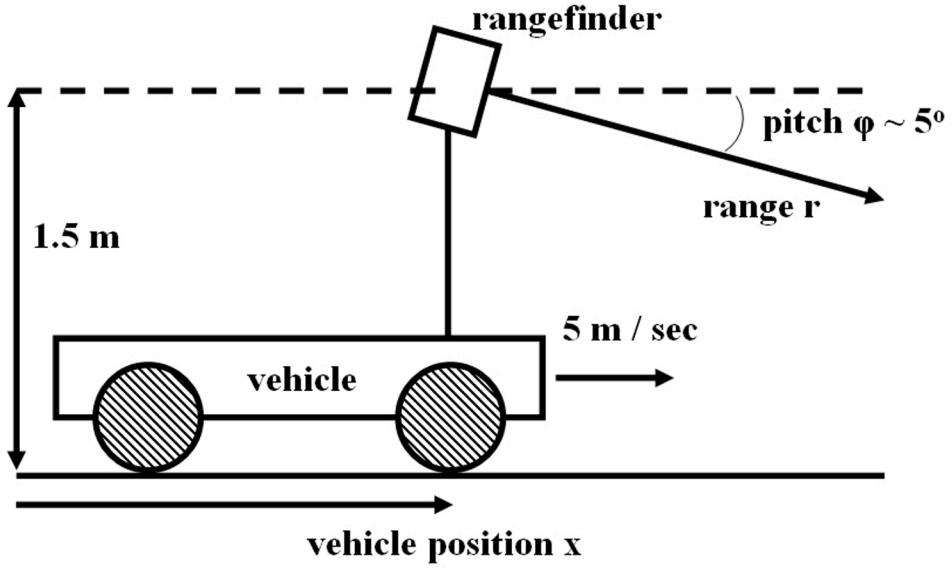

Figure 4.1: Problem geometry of the simulated one-dimensional cart and rangefinder.

range per scan and scans at 75 Hz. The LIDAR is modeled with three sources of noise: raw ranging error due to sensor noise, uncertainty in sensor pitch due to encoder noise, and uncertainty in vehicle location due to GPS noise. These three sources of noise are specific instances of uncertainty of types 1, 2, and 4 from section 4.2.1. The first noise source, raw ranging error due to sensor noise, is assumed corrupted by additive Gaussian noise so that the measured ranges are:

$$\hat{r} = r + \delta_r \tag{4.22}$$

where r is the true range and $\delta_r \sim \mathcal{N}(0, \sigma_r^2)$ is the additive Gaussian noise. The other uncertainties, pitch and vehicle position, are incorporated into the sensor orientation parameters $\underline{p}$,

$$\underline{p} = \begin{pmatrix} \phi \\ x \end{pmatrix} \tag{4.23}$$

where ϕ is the sensor's pitch angle downward from horizontal, and x is the vehicle's position relative to the arbitrarily-set origin. These parameters are assumed corrupted by additive Gaussian noise:

$$\hat{\underline{p}} = \underline{p} + \underline{\delta}_p \tag{4.24}$$

with $\underline{\delta}_p \sim \mathcal{N}\left(\underline{0}, diag\left[\sigma_\phi^2, \sigma_x^2\right]\right)$, where σ_ϕ^2 is the variance in the sensor's pitch measurements and σ_x^2 is the variance in the vehicle's position measurements. The noise sources are set according to typical accuracy of a LIDAR, angle encoder, and differential GPS receiver:

$$\begin{aligned} 3\sigma_r &= 0.1m \\ 3\sigma_\phi &= 0.5^\circ \\ 3\sigma_x &= 0.6m \end{aligned} \tag{4.25}$$

The simulation is constructed from a true terrain model entered at the program's start. Ranges are calculated according to truth values, and then the 'measured' values of the range, sensor pitch, and vehicle location are corrupted with appropriate noise for use in the terrain estimation algorithm. To proceed, note the transformation $f(\cdot)$ to produce the terrain measurement is:

$$\hat{\underline{r}}^{ENU} = \hat{r} \cdot \begin{pmatrix} \cos(\hat{\phi}) \\ -\sin(\hat{\phi}) \end{pmatrix} + \begin{pmatrix} 0 \\ 1.5 \end{pmatrix} + \begin{pmatrix} \hat{x} \\ 0 \end{pmatrix} \tag{4.26}$$

where $\hat{\underline{r}}^{ENU}$ is a two-element vector estimate of the in-plane location of the measurement as well as the elevation of the measurement itself. The Jacobian ma-

trices also have a particularly simple representation:

$$\begin{aligned} J_{\underline{p}}\left(\hat{\underline{p}}, \hat{r}\right) &= \begin{pmatrix} -\hat{r} \cdot \sin(\hat{\phi}) & 1 \\ \hat{r} \cdot \cos(\hat{\phi}) & 0 \end{pmatrix} \\ J_{\underline{r}}\left(\hat{\underline{p}}, \hat{r}\right) &= \begin{pmatrix} \cos(\hat{\phi}) \\ -\sin(\hat{\phi}) \end{pmatrix} \end{aligned} \tag{4.27}$$

Using the transformation and Jacobian matrix from equations 4.26 and 4.27, the computations proceed as described in section 4.2. The only caveat is that the equations are modified from a full three-dimensional East-North-Up representation to a two-dimensional East-Up representation. In addition, nominal measurement locations $\hat{E}$ are rounded to the nearest cell center for simplicity. The terrain is divided into 1 m grid cells, and measurements are applied up to 3 cells away from their nominal location.

The simulation features a hypothetical ground with flat terrain $(0 \leq x < 100)$, a small bump $(100 \leq x < 150)$, a small ditch $(150 \leq x \leq 155)$, a large vertical face $(155 < x \leq 170)$, and a plateau $(170 < x \leq 200)$. Figure 4.2 shows the true ground surface along with the estimated elevation $\hat{U}_{GM}$ and its associated $\pm 2\sigma_{GM}$ bounds, in 1 m increments. The terrain estimation algorithm reveals several points of interest. First, grid cells 1 to 13 have no terrain estimate because the vehicle starts at $x = 0$ and the rangefinder is angled such that no data is ever gathered near these cells. Figure 4.3 (left) gives the total association probability and number of measurements assigned to each cell to confirm sensor measurement density in more detail. Figure 4.3 (left) also shows that more measurements are assigned to cells near $j = 100$, 155, and 170, which contain near-vertical faces. Cells near $j = 150$ receive less measurements, in contrast, because a portion of the ditch is occluded. A second feature of the terrain estimation algorithm is that it estimates the sensed portions of the terrain accurately. Figure 4.4 (left) shows the terrain estimate for grid cell $j = 50$ (on the flat ground) as it evolves

during the simulation. The estimate is observed to have no information until $t \approx 5.85$ seconds, when the first measurements are applied to this particular cell. From $t \approx 5.85 \rightarrow 7.33$ seconds, the rangefinder continues to sweep over the cell, delivering new measurements to the terrain estimation algorithm. During this time, the terrain estimate is refined, and the variance within the cell fluctuates based on the quality of the measurements received. Because the terrain is flat and the measurements are not particularly noisy, the variance for this cell is dominated by the variances of the individual measurements assigned to it. Figure 4.3 (right) shows the level of probability mass $\sum p_i$ of the estimate of cell $j = 50$ as the simulation progresses. The approximately linear increase in probability mass / time is characteristic of a rigid sensor mounting and constant vehicle speed. A third point to note about the simulation is that the estimation errors σ_{GM} are smaller for the top of the plateau than they are for the bottom. This is a consequence of sensor geometry: the elements of the Jacobian matrix in equation 4.27 that map sensor pitch error into vertical terrain uncertainty are smaller when the true ranges r are smaller, which occurs for taller terrain relative to the sensor. This particular point emphasizes the effects of the statistical representation over *ad hoc* approaches, as it utilizes geometrical relationships to improve estimation accuracy where appropriate.

The vertical face present in the terrain in cell $j = 170$ is also estimated appropriately. Figure 4.2 shows that the estimated elevation of the cell is approximately half the elevation of the plateau, though the variance of the estimate near these cells is quite large. In these cells, the variance is dominated by the spread of the terrain measurements, which are taken all along the vertical face of the plateau. In addition, the probability mass along the vertical face of the plateau is significantly higher than at any other point in the simulation, as shown in

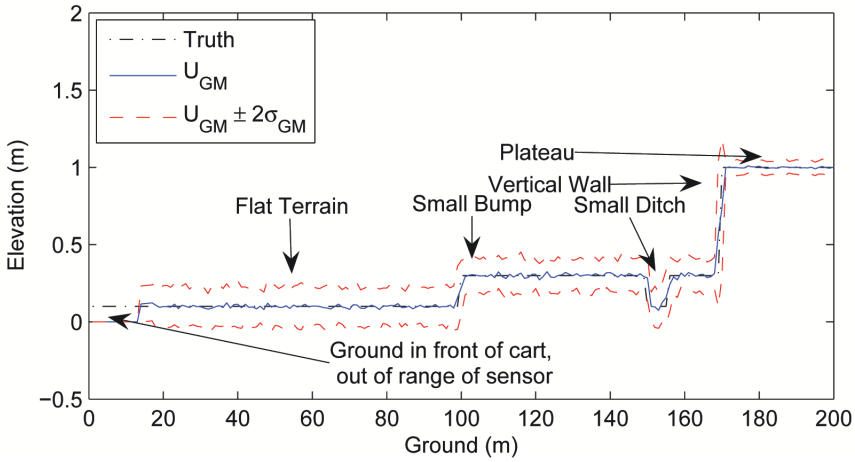

Figure 4.2: True terrain, terrain estimate $\hat{U}_{GM}$, and $\pm 2\sigma_{GM}$ bounds for a one-dimensional terrain example with cart moving at 5 m/s.

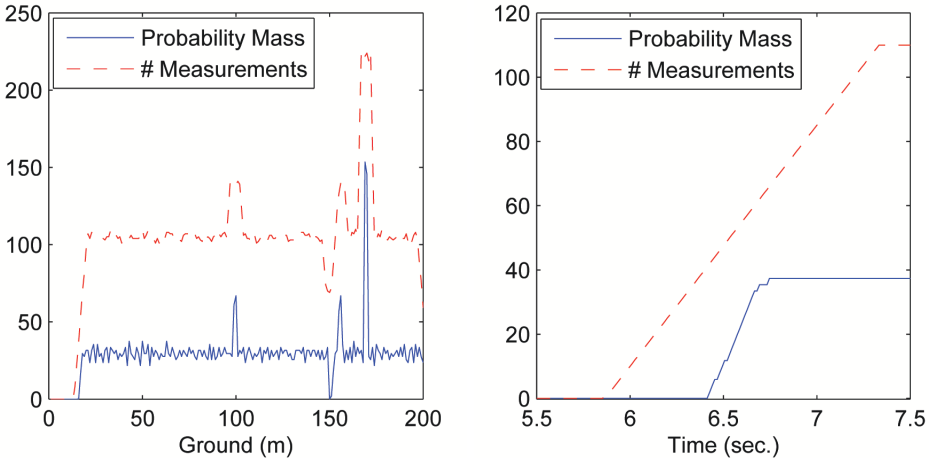

Figure 4.3: (Left) Terrain estimate measurement probability mass levels and number of measurements assigned for one-dimensional terrain example. High probability mass in this example indicates the presence of a vertical or near-vertical face. (Right) Terrain estimate measurement probability mass level and total number of measurements accumulated over time for terrain cell $j = 50$.

Figure 4.3 (left). This property is particularly useful for gathering information about objects protruding from the natural ground plane.

As a point of reference, the terrain estimation algorithm is compared against a naïve algorithm that simply reports the maximum and minimum elevation measurement assigned to each cell. Note this naïve algorithm is effectively a convolution using both a maximum and minimum operator over the same ranges of cells considered by the statistical algorithm. The maximum and minimum measurement lines for the naïve algorithm are plotted in dotted green in Figure 4.4 (left) for terrain cell $j = 50$. For 5 m/s cart speeds and standard flat terrain like cell $j = 50$, the naïve approach behaves similarly to the $\pm 2\sigma_{GM,j}$ elevation bounds. This supports the use of the terrain estimation algorithm for statements of statistical confidence about the terrain. However, Figure 4.4 (left) shows that the statistical algorithm requires fewer measurements to converge to its final uncertainty $\sigma_{GM,j}$, because it considers each measurement as a statistical quantity. The naïve approach, in general, requires larger sample sizes to establish reasonable minimum and maximum elevation bounds.

A second simulation, performed on the same terrain but with a cart speed of 0.5 m/s, is shown in Figure 4.4 (middle). In this simulation the cart moves much slower, so there are a larger number of measurements assigned to each cell. Figure 4.4 (middle) shows that the terrain estimation algorithm makes use of the extra data to generate smoother estimates and uncertainties. The naïve algorithm produces worse terrain estimates with the extra data, however, as its minimum and maximum estimates reflect the outliers in the sensed terrain data. In this sense the minimum and maximum measurements in the naïve algorithm diverge monotonically from each other, regardless of sensor measurement statistics. The statistical algorithm, in contrast, provides a well-defined framework for statements of statistical confidence about the terrain model.

A third simulation, performed on the same terrain with a vehicle speed of 25 m/s, is shown in Figure 4.4 (right). In this simulation the cart moves very fast, so very few measurements are gathered in each cell. In this case the estimates from the statistical terrain algorithm are not as smooth as they were in the cases with more data. They do, however, still report uncertainties similar to those reported in the previous examples. The output of the naïve algorithm is not as stable as the amount of data changes. In Figure 4.4 (right), for example, the naïve algorithm consistently reports bounds that are smaller than the bounds reported in the previous two simulations. The statistical algorithm generates accurate terrain uncertainties after only one measurement, while the naïve algorithm consistently underestimates uncertainty. This simulation shows the robustness of the statistical terrain estimation algorithm even with much less data than in the other simulations. The naïve algorithm is much worse: it produces optimistic estimates from a small number of measurements, and pessimistic estimates as more data is gathered.

Figures 4.5 and 4.6 show magnified views of the final terrain estimates near the ditch and wall, respectively. These Figures support the robustness of the terrain estimation algorithm in comparison to the naïve algorithm. In Figure 4.5, for example, the naïve algorithm blurs over the small ditch at each simulated cart speed. These estimates could be improved in the naïve algorithm at slow speeds by only applying a measurement in its nominal cell. At faster speeds this will reduce the naïve algorithm's performance, however, due to the decreased number of measurements in each cell. The statistical algorithm, which uses association weights to form optimal estimates, suffers from neither of these problems. The statistical terrain estimates remain accurate at all speeds, and improve as the amount of data increases.

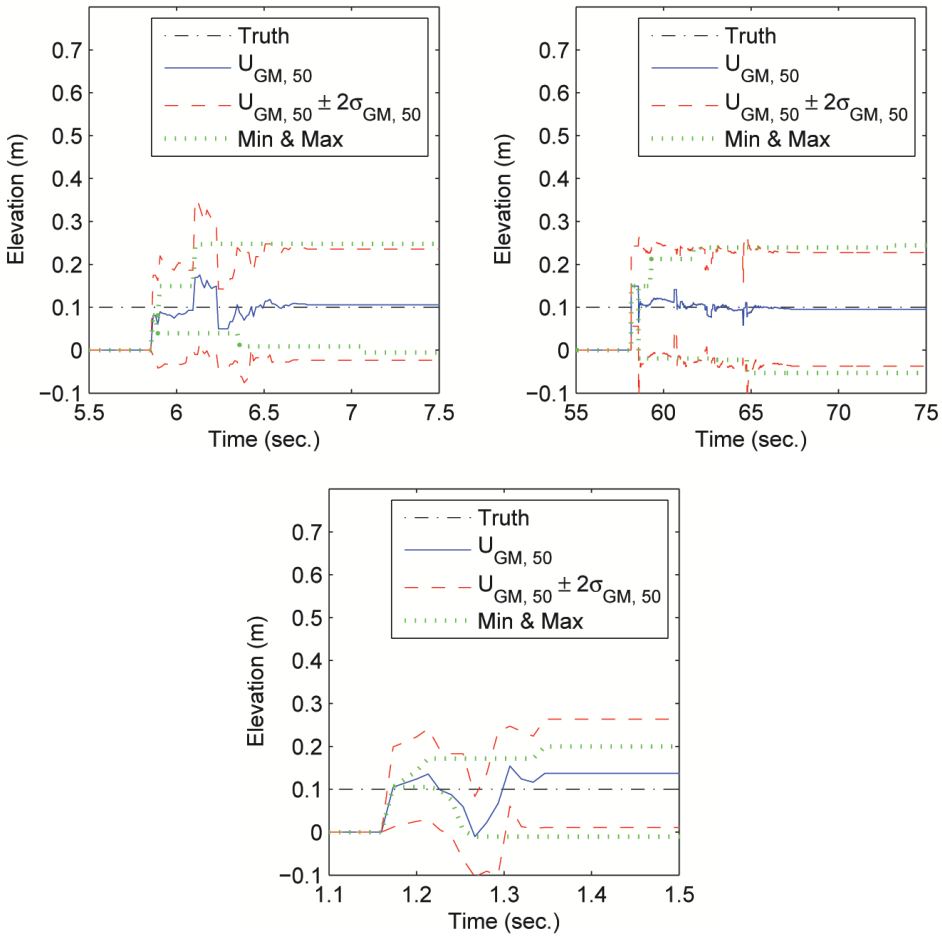

Figure 4.4: (Left): True terrain, terrain estimate $\hat{U}_{GM,j}$, $\pm 2\sigma_{GM,j}$ bounds, and minimum and maximum measurements accumulated over time for terrain cell $j = 50$ with vehicle moving at 5 m/s. (Middle): The same quantities with vehicle moving at 0.5 m/s. (Right): The same quantities with vehicle moving at 25 m/s.

A similar comparison can be seen in Figure 4.6, which shows a magnified view of the final terrain estimates near the wall. As before, the naïve algorithm grows the boundaries of the wall due to its *ad hoc* method of applying measurements to cells. Figure 4.6 (middle) shows the naïve algorithm counterintuitively produces worse terrain estimates as the number of measurements increases, as it is sensitive to outliers. Figure 4.6 (right) also confirms that the algorithm produces overly optimistic estimates as the number of measurements decreases.

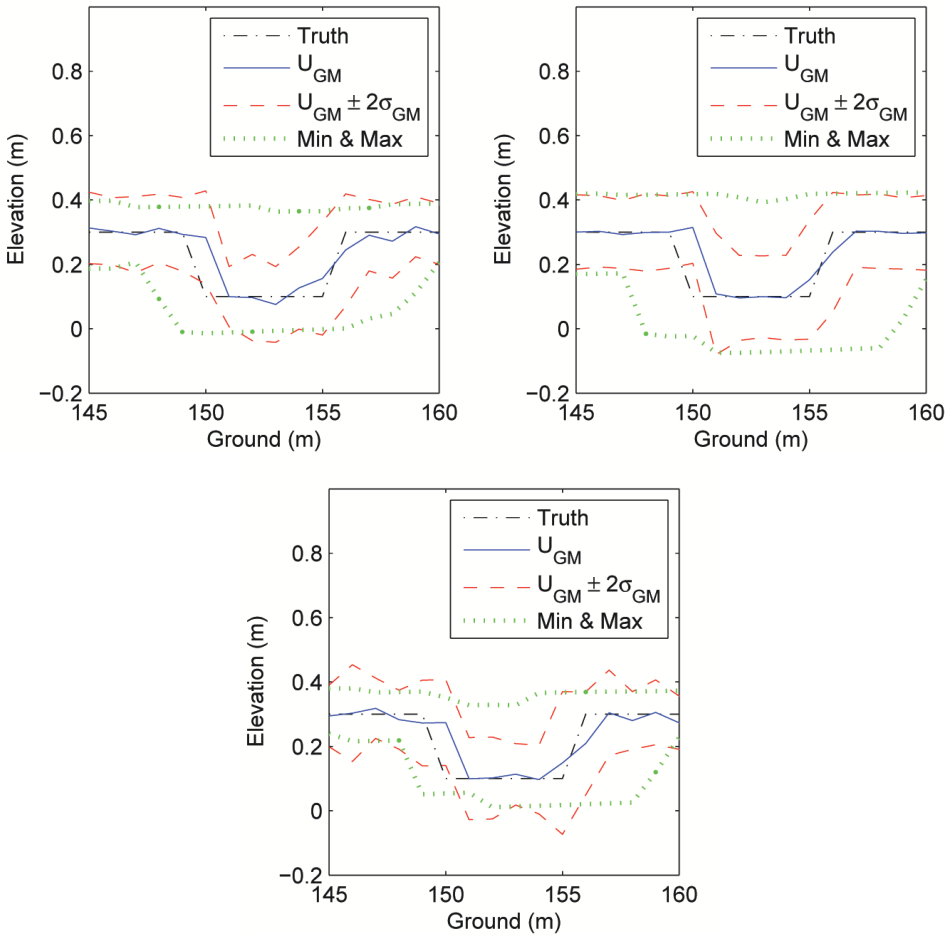

Figure 4.5: (Left): True terrain, terrain estimate $\hat{U}_{GM,j}$, $\pm 2\sigma_{GM,j}$ bounds, and minimum and maximum measurements near a simulated ditch with vehicle moving at 5 m/s. (Middle): The same quantities with vehicle moving at 0.5 m/s. (Right): The same quantities with vehicle moving at 25 m/s.

The statistical algorithm, in contrast, produces consistent estimates across all the simulated cart speeds.

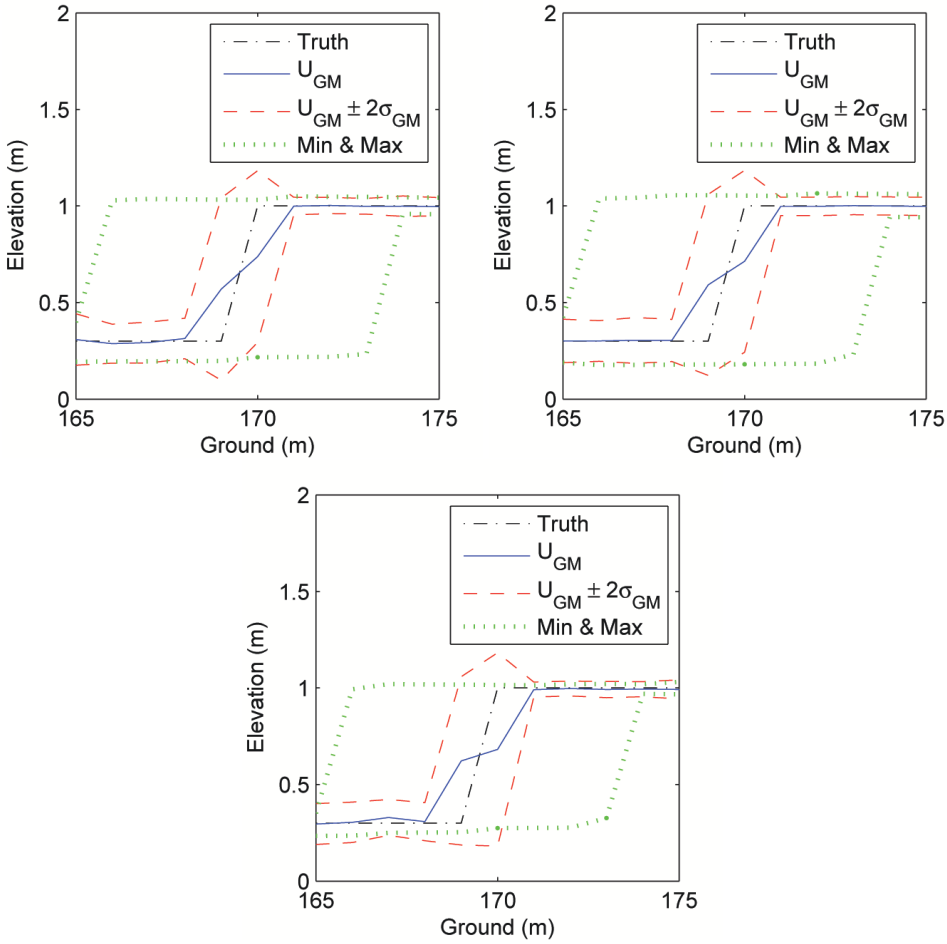

Figure 4.6: (Left): True terrain, terrain estimate $\hat{U}_{GM,j}$, $\pm 2\sigma_{GM,j}$ bounds, and minimum and maximum measurements near a simulated wall with vehicle moving at 5 m/s. (Middle): The same quantities with vehicle moving at 0.5 m/s. (Right): The same quantities with vehicle moving at 25 m/s.

4.4 Real World Application: A Moving Ground Vehicle Equipped with Multiple Sensors

The terrain estimation algorithm described in section 4.2 is capable of generating and maintaining a terrain model with confidences in real-time. In this section, the capabilities of the terrain estimation algorithm are applied to a more

challenging and realistic problem: constructing and maintaining a real-time terrain map using multiple sensors in a dynamical environment. In particular, this section tests the algorithm on Cornell University's 2005 DARPA Grand Challenge autonomous ground vehicle [89].

In the DARPA Grand Challenge and the National Qualifying Event, the terrain estimation algorithm provided estimates to a path planner, which used the terrain estimates to determine traversable areas of the course. In both the Grand Challenge and the National Qualifying Event, the algorithm helped the vehicle successfully distinguish traversable flat ground, speed bumps, and shallow inclines from intraversable features such as rocks, large vegetation, and tank traps. Miller *et al.* discusses the method by which the terrain estimates were used for autonomous path planning, which will not be addressed further in this paper [58]. In the present study, the Cornell vehicle is used as a means to validate the terrain estimation algorithm against a known, surveyed landscape. To begin, the Cornell vehicle is built upon a 'Spider Light Strike Vehicle' from Singapore Technologies Kinetics, shown in Figure 4.7.

The Spider is equipped with a Trimble Ag252 GPS receiver and an inertial navigation system for attitude and position determination, as well as three SICK LMS 291 LIDARs for terrain estimation. All three LIDAR units are fixed to the front hood of the Spider, as shown in Figure 4.7. The left and right LIDARs are mounted rigidly and pitched to intersect the ground plane approximately 15 m and 20 m in front of the vehicle. The center LIDAR is mounted on a two-axis gimbal, with yaw and pitch actuated by a pair of MAXON EPOS 70-4 rotary motors and Harmonic Drive gear boxes. Figure 4.8 shows the Spider's actuated gimbal and LIDAR units.

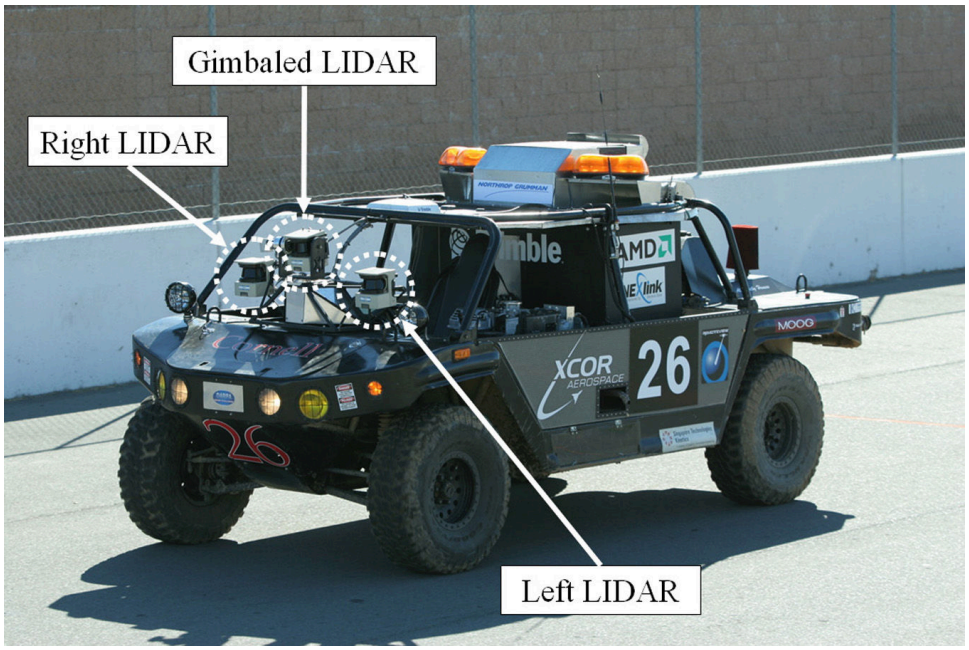

Figure 4.7: The Spider Light Strike Vehicle used to test the terrain estimation algorithm.

All types of error discussed in section 4.2.1 are modeled in this experiment. Individual LIDAR errors are modeled as ranging error with a typical accuracy of ± 5 cm, and an angular error with coning angle of $\pm 0.24^{\circ}$ due to the expansion of the detecting light pulse [84]. These two error sources affect terrain measurements differently: the range error yields uncertainty in the sensing direction regardless of detection range, while the angular error term yields higher uncertainty at longer ranges for this problem geometry. The second type of error, sensor orientation error, is modeled as error up to $\pm 0.7^{\circ}$ on the yaw and pitch angles reported by the gimbal encoders. This term only affects the gimbaled LIDAR; the two fixed LIDARs' orientation parameters are determined via offline calibration against the gimbaled LIDAR. The third and fourth sources of sensing error are due to the attitude and position estimator. Orientation errors of less than 0.2° and position errors of less than $0.2m$ are common in this sys-

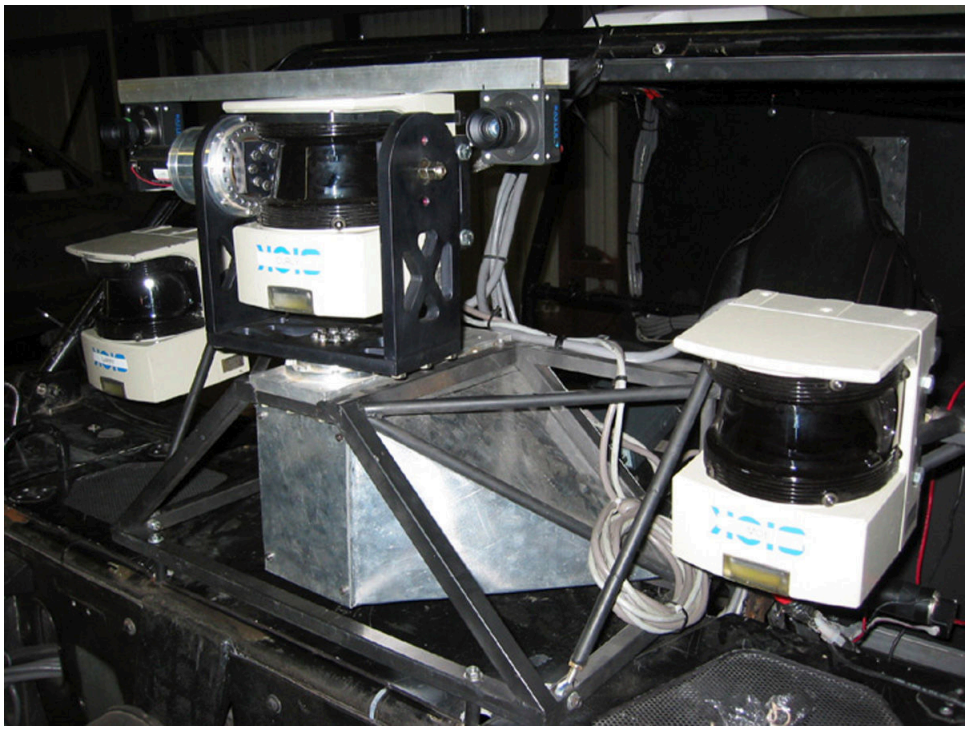

Figure 4.8: The Spider's three laser rangefinders (LIDARs) and two-axis gimbal platform.

tem, though the exact MSE matrices used are those reported by the estimators in real-time.

The statistical analysis and Jacobian transformations discussed in section 4.2.1 are easily applied to the sensors on the Spider to model these four sources of error. To begin, each raw LIDAR range measurement $\underline{r}$ is defined in its own measurement axes:

$$\underline{r} = \begin{pmatrix} \rho \cdot \cos\left(\theta_{\mathrm{D}}\right) \\ \rho \cdot \sin\left(\theta_{\mathrm{D}}\right) \\ 0 \\ 1 \end{pmatrix} \tag{4.28}$$

where ρ is a scalar range reported by the LIDAR and θ_{D} is the reported detection

angle. In this measurement frame, the x-axis is chosen to point out the front of the LIDAR, the z-axis is chosen perpendicular to the LIDAR detection plane, and the y-axis completes the right-handed coordinate system. The origin of the coordinate axis is the origin of the measurement: the point at which $\rho = 0$. From here, the measurement is transformed into a terrain detection in the ENU reference frame. These two coordinate axes are represented graphically in Figure 4.9. Mathematically, the transformations are performed using 12 separate orientation parameters for each LIDAR:

1. LIDAR Euler angles S_ψ, S_ϕ and S_θ with respect to the vehicle body.
2. LIDAR position elements S_x, S_y, and S_z with respect to the inertial navigation system.
3. Vehicle Euler angles ψ, ϕ, and θ with respect to the ENU reference frame.
4. Vehicle position O_x, O_y, and O_z with respect to the ENU reference origin.

The vector $\underline{p}$ of sensor orientation parameters for this particular application is therefore:

$$\underline{p} = (\, S_\psi \quad S_\phi \quad S_\theta \quad S_x \quad S_y \quad S_z \quad \psi \quad \phi \quad \theta \quad O_x \quad O_y \quad O_z \,)^T \tag{4.29}$$

where all elements of this vector except for S_x, S_y, and S_z are modeled as uncertain.

These parameters are used along with the raw range measurement from equation 4.28 to generate measurements and an associated mean square error (MSE) matrix as described in section 4.2.1. All uncertain orientation parameters

Figure 4.9: Example sensor axes and ENU reference axes that determine the transformation from sensor to terrain measurements.

are treated as corrupted with additive zero mean, Gaussian white noise. The maximum magnitude of each error source, discussed in the beginning of section 4.4, is taken as its 3σ value. The exceptions are the attitude and position covariance matrices, which are taken from the attitude and position estimator as the algorithm runs. The covariance matrices $Q_{\underline{p}}$ and $Q_{\underline{r}}$ for the experimental setup are block diagonal:

$$\begin{aligned} Q_{\underline{p}} &= diag\left[\sigma^2_{S_\psi}, \sigma^2_{S_\phi}, \sigma^2_{S_\theta}, \sigma^2_{S_x}, \sigma^2_{S_y}, \sigma^2_{S_z}, P_{\psi\phi\theta}, P_{xyz}\right] \\ Q_{\underline{r}} &= diag\left[\sigma^2_{\rho}, \sigma^2_{\theta_D}\right] \end{aligned} \tag{4.30}$$

where $P_{\psi\phi\theta}$ and P_{xyz} are the attitude and position MSE matrices reported by the attitude and position estimators at the time each measurement is taken.

The remainder of the sensor fusion algorithm was implemented according to sections 4.2.2 and 4.2.3. A list of algorithm steps is given below:

1. Obtain current encoder, attitude, and position measurements to construct $\hat{\underline{p}}$.
2. Obtain a scan line of raw measurements $\hat{\underline{r}}_i$ from a LIDAR.
3. Construct and cache the transformation and Jacobian matrices for the measurements in the obtained scan line.

4. For each raw measurement $\hat{\underline{r}}_i$:

   - Calculate a terrain measurement estimate $\hat{\underline{r}}_i^{ENU}$ and MSE matrix P_i^{ENU} using equations 4.5 and 4.6 and the cached transformation and Jacobian matrices.
   - For each cell (up to k) near the nominal measurement $\hat{\underline{r}}_i^{ENU}$:
     - Calculate the association probability p_i for this measurement in this applied to the cell using equation 4.15.
     - Calculate a posterior elevation measurement $\hat{U}_i$ and variance $\sigma^2_{\hat{U}_i}$ from $\hat{\underline{r}}_i^{ENU}$ for the cell, using equations 4.16 and 4.17.
     - Fuse $\hat{U}_i$, $\sigma^2_{\hat{U}_i}$, and p_i with existing measurements in the cell by updating the 4 numbers discussed in section 4.2.4 stored for the cell.

5. When elevation and uncertainty estimates are desired, calculate them using equations 4.19 and 4.20.

For this particular implementation, grid cells were set to be 40 cm by 40 cm squares. This cell size was motivated by the size of the Spider's wheels and the desired resolution of the landscape's features [58]. An arbitrary latitude and longitude near the experiment were chosen as the reference origin. For simplicity, the nominal East - North location of each measurement was rounded to the nearest cell center.

To ensure computational feasibility, measurements were applied to cells at most 2 m from the nominal measurement location. This threshold was used because the measurement MSE matrix P^{ENU}, when accounting for all sources of error, yielded no significant effect in cells farther than 2 m from the nominal

measurement. In addition, a probability threshold of $p_i \geq 10^{-4}$ was also enforced, so measurements were applied only to cells in which they were likely to belong. While this did not reduce the computational search space of this particular implementation, it did tend to eliminate the effects of rare faulty measurements.

The entire terrain estimation algorithm was implemented on a commercial four-processor server running Microsoft Windows Server 2003. Each processor in the server was a 2.0 GHz AMD Opteron 846. Each of the three rangefinders was run on its own event-driven reader thread at the LIDARs' scanning frequency of 75 Hz. Data from each LIDARs was read across a 500 kbps serial connection. LIDAR scans were queued for sequential fusion into the terrain map, so that a full scan from each LIDAR was processed before moving on to the next. Each LIDAR measurement was transformed to a terrain detection using all 12 LIDAR orientation parameters. Transformation matrices common to a particular LIDAR scan were computed once per scan and applied to all measurements in that scan. Common components of the Jacobian matrices were also computed once per scan. These optimizations enabled the Spider to process all 40725 measurements per second delivered by the 3 LIDARs using less than 10% of a single processor's computing resources. These results suggest the algorithm could be run on a much smaller processor, or scaled across a parallel architecture for a large number of distributed sensors. The algorithm required storage space linear in the number of cells: only 4 floating point numbers per cell. It could thus maintain many miles of mapping regions in active memory. In contrast, a system retaining the same measurements without a fusion scheme would require storage space of nearly 0.95 Mb per second.

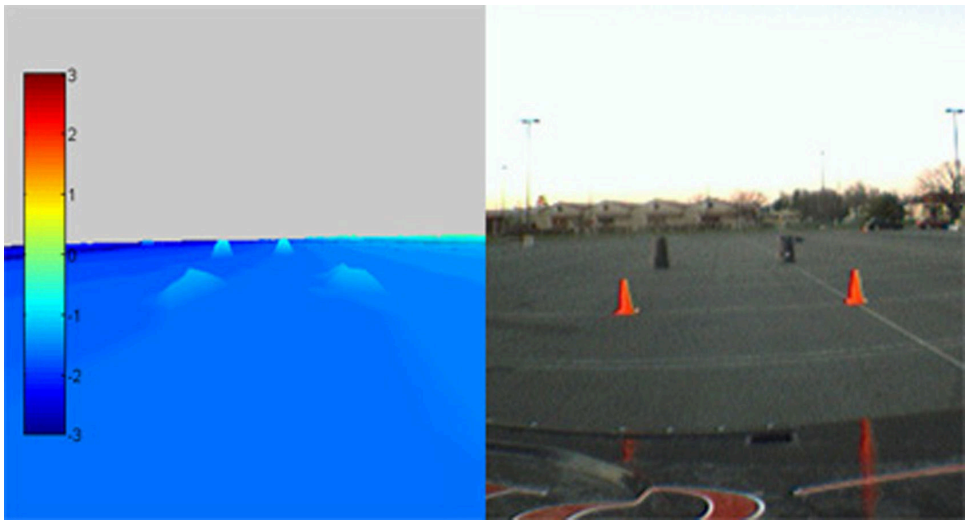

Figure 4.10: Sample $\hat{U}_{GM} + 2\sigma_{GM}$ elevation map resulting from the Spider's sensor fusion scheme.

Figure 4.10 shows a sample terrain model generated from the terrain estimation algorithm as the Spider approaches several objects in an experimental run. In this example, the $\hat{U}_{GM} + 2\sigma_{GM}$ elevations from equations 4.19 and 4.20 (in meters) are plotted relative to an arbitrary ENU origin selected near the test site. These elevations are plotted across a color spectrum, and the axes in Figure 4.10 are to scale. Like the example in section 4.3, the algorithm generates valid estimates and accurate confidence bounds. For example, the $\hat{U}_{GM}$ elevations for the cells near the 0.8 m tall trash cans are approximately 0.45 m higher than surrounding cells, and the $\hat{U}_{GM}$ elevations for the cells near the 0.46 m traffic cones are approximately 0.25 meters. These lower elevation estimates reflect the fact that the cells containing these objects also contain some exposed portions of the ground plane. The uncertainties, however, are appropriately large for these cells: $\sigma_{GM} \approx 0.35$ m for the cells near the trash cans and $\sigma_{GM} \approx 0.17$ m for the cells near the traffic cones. Figure 4.11 shows the estimated elevation $\hat{U}_{GM}$ and estimation uncertainty σ_{GM} near the trash cans in greater detail.

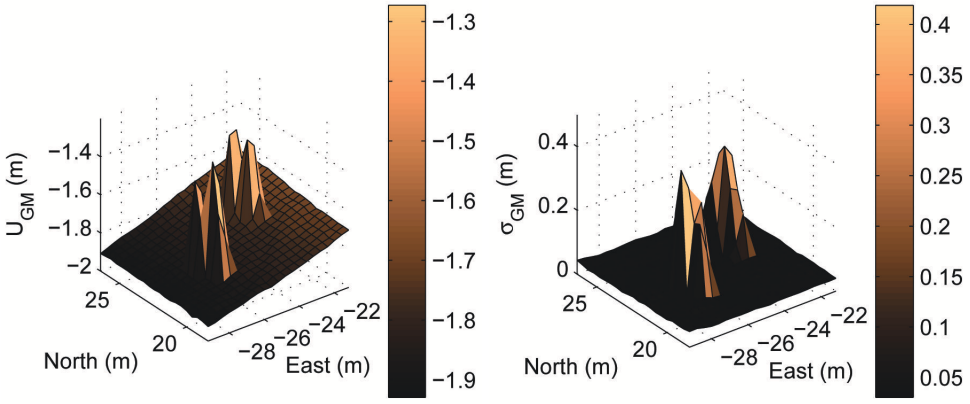

Figure 4.11: (Left) Final $\hat{U}_{GM}$ map near the two 0.8 m tall trash cans. (Right) Final σ_{GM} map near the two 0.8 m tall trash cans.

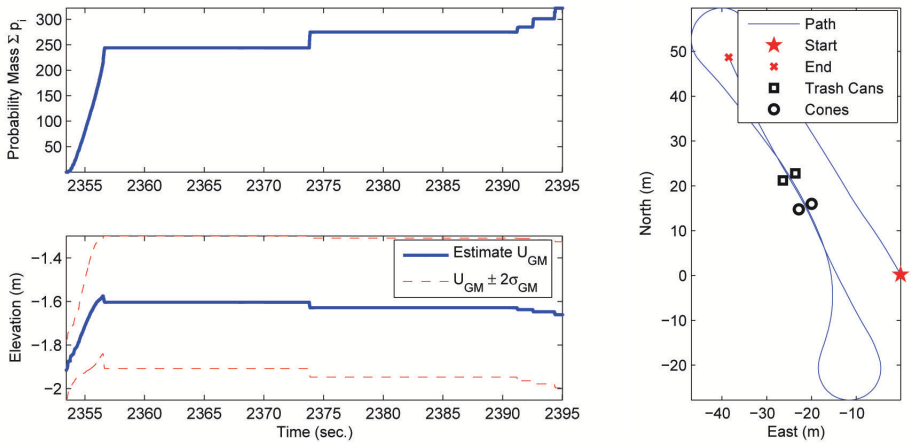

Figure 4.12: (Left) Total association probability and $\hat{U}_{GM} \pm 2\sigma_{GM}$ bounds over time for a particular terrain cell containing a portion of a trash can. (Right) Vehicle ground track during real-time terrain experiment. The vehicle intersects the line connecting the trash cans at $t \approx 2358$ sec., $t \approx 2374$ sec. and $t \approx 2395$ sec.

Figure 4.12 (left), in contrast, shows the evolution of the terrain estimate for one cell near one of the trash cans. Figure 4.12 (right) shows that during this test, the Spider passes near the trash cans three times: once with the trash cans in the periphery of the sensors' footprints at $t \approx 2358$ sec., and twice directly

between the two trash cans at $t \approx 2374$ sec. and $t \approx 2395$ sec. Figure 4.13 shows vehicle speed and heading during the test. Notice that the average speed of 5.4 m/s is comparable to speeds at which a human drives, confirming the real-time capabilities of the algorithm. Figure 4.12 (left, top) shows the total association probability sum $\sum p_i$ accumulated for the cell over time. Only points at which the probability increases are included. Notice that most of the association probability is assigned on the Spider's first pass by the trash can, suggesting that repeated passes by the trash can are not necessary for the algorithm to generate accurate results in real-time. The bottom left subplot of Figure 4.12 shows the terrain elevation estimate $\hat{U}_{GM}$ and the $\pm 2\sigma_{GM}$ bounds for the same cell. Note that the elevation estimate for the terrain is relative to an arbitrarily-set ENU origin, not the ground plane, so the absolute elevation scale on the vertical axis is non-intuitive. The variance in this particular cell is also much higher than in surrounding cells, indicating the presence of variegated terrain or obstacles in comparison with other nearby cells. The elevation estimate in this cell also fluctuates over time as more measurements are applied to it: some measurements, from the near-vertical face of the trash can, tend to increase the elevation estimate upward. Others, from the surrounding flat ground, tend to decrease the elevation estimate. The effect of these different measurements is also seen in the cell's variance, which increases throughout the experiment. Finally, notice that the elevation estimate changes most on the Spider's first pass by the trash can, and that subsequent passes do not cause a substantial increase in total association probability of measurements assigned to that cell. The algorithm is therefore capable of producing accurate estimates in real-time, at reasonable speeds, and without revisiting old terrain. Notice that the algorithm produces relatively smooth and dense terrain estimates; this occurs because individual

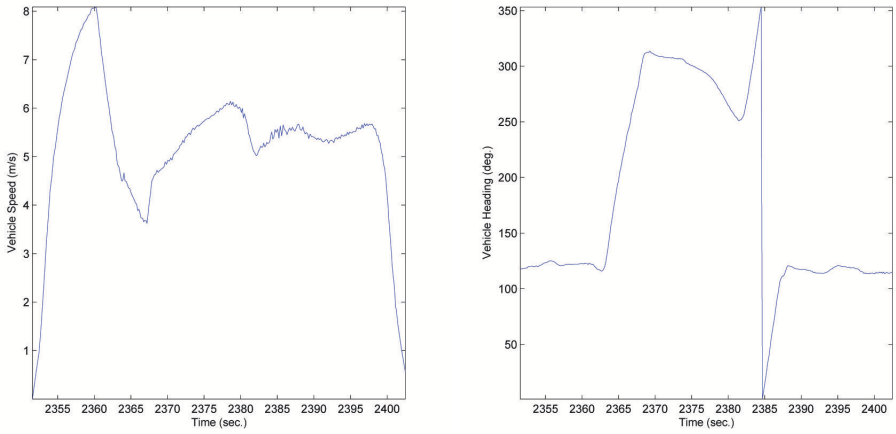

Figure 4.13: (Left) Vehicle speed during real-time terrain experiment. (Right) Vehicle heading during real-time terrain experiment.

sensor measurements are applied in more than one cell as per equation 4.18. This smoothed and correlated terrain model arises naturally from the estimation algorithm and the continuous posterior distribution of the errors in the sensors. It contrasts sharply with other terrain estimation algorithms that make use of multiple passes or recursion on the terrain estimate to smooth it out [69], [5], [34], [97].

While this algorithm produces relatively consistent estimates, it does have several drawbacks. First, it is not integrated with the attitude and position estimator, so it is directly subject to incorrect attitude and position estimates. For example, large discontinuous jumps resulting from update steps in an Extended Kalman Filter produce 'phantom' walls and obstacles due to incorrectly-placed LIDAR scans. These types of phantom walls can also arise from unmodeled timing delays and inaccurate noise models, depending on the severity of the modeling error. Figure 4.14 shows a sample phantom obstacle resulting from a mismodeled noise source. This result was generated in a second driving exper-

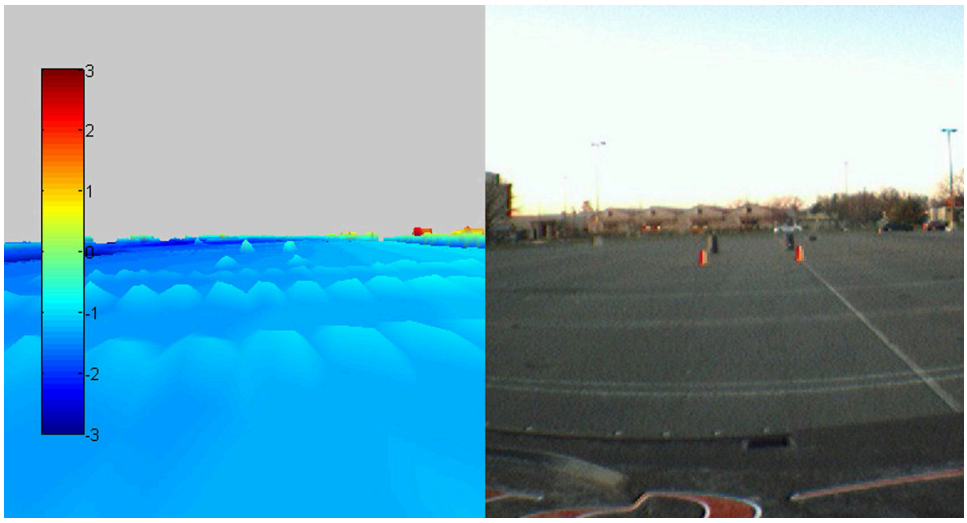

Figure 4.14: 'Phantom' walls in the $\hat{U}_{GM} + 2\sigma_{GM}$ map arising from high variance due to incorrectly-modeled gimbal encoder noise.

iment in which the gimbaled LIDAR was set to a sinusoidal pitching pattern, suggesting that mismodeled noise in the gimbal encoders may be the cause. In general these anomalies can be fixed or minimized with the addition of more accurate models of the offending sources of error. Error checks can also be performed to determine whether entire LIDAR scans are likely to be faulty.

A second drawback of this and other terrain estimation schemes is that all the sensors must be calibrated correctly. Such a task is particularly difficult with LIDAR units mounted on a vehicle, as precise positions and orientation angles are very difficult to measure. For this implementation, Cornell's Grand Challenge team used the Spider's attitude and position estimator along with a section of flat airplane runway to calibrate the gimbaled LIDAR with respect to the ground. Then, the fixed LIDAR orientations were found in software through a greedy search over their orientation angles by comparing their terrain estimates to those produced by the gimbaled LIDAR [74]. This automatic calibration technique produced terrain estimates consistent to within one or two centimeters

between the LIDAR units.

A final drawback of this terrain estimation scheme is the fact that only the first two moments of each terrain cell are retained. As a result, the algorithm can give misleading terrain estimates inside tunnels, or in cells in which there are distinct clusters of elevations such as tree canopies. These arise because the algorithm makes no attempt to characterize or classify macroscopic features of the terrain, such as vegetation or terrain solidity, across multiple cells. One alternative explored in the literature is the idea of expanding the terrain estimate to a 3D set of binary obstacle / no-obstacle evidence grid cells [52]. However, computational resources tend to limit the practicality of such 3D grids. Furthermore, path searches through a 3D environment take large amounts of time, especially considering the vehicle is effectively constrained to the ground. The given approach instead attempts to concentrate on representations that preserve the computational efficiency and searchability of the 2D framework without sacrificing the richness of the representation.

4.5 Conclusions

This paper presents a real-time terrain mapping and estimation algorithm using Gaussian sum height densities to model terrain variations in a planar gridded elevation model problem. Techniques for statistical analysis and estimation of each individual sensor measurement are used to account for multiple sources of error in the transformation from sensor measurements to terrain detections in a rigorous manner. Linearization techniques are used to approximate the uncertainty of each sensor measurement with a nominal measurement and a joint

Gaussian distribution in the physical East, North, and Up directions. Measurements are associated to multiple locations in the elevation model using a Gaussian sum conditional density to account for uncertainty in measured elevation as well as uncertainty in the in-plane location of the measurement.

The accuracy and interpretation of the algorithm is evaluated using a simple one-dimensional example with a hypothetical range sensor. Results show accurate statistics as the vehicle traveled over a bump, ditch, large vertical face, and plateau. Consistent statistical estimates were also verified across a range of vehicle velocities. Straightforward use of minimum / maximum elevations measured directly from the sensors produced comparatively inconsistent results, with degrading performance as the number of sensor measurements increased.

The algorithm was demonstrated in a practical application of real-time mapping with multiple sensors on a platform moving at realistic speeds. The vehicle, Cornell's DARPA Grand Challenge entry, included multiple LIDAR sensors, GPS, and attitude sensors. The algorithm was shown capable of producing statistically accurate and computationally feasible elevation estimates on dense terrain models, as well as estimates of the errors in the terrain model. This type of dense terrain map holds more information than traditional real-time mapping approaches, and it has the potential to be useful in a number of autonomous or remote mapping applications.

CHAPTER 5

STABLE AND EFFICIENT TRACKING OF MULTIPLE DYNAMIC OBSTACLES UNDER LARGE VIEWPOINT CHANGES

5.1 Introduction

Modern autonomous mobile robots operating at human scales of size face the significant challenge of understanding the dynamic environment in which they (and we) interact. The challenge is more than a pedagogical curiosity, as failures in a human-populated operating environment could turn science fiction into nightmarish science fact. These dangers are particularly acute for full-size robotic vehicles, where tracking accidents and traffic accidents are separated only by a thin layer of hardware.

The primary difficulty lies not in sensor deficiency, but in sensor interpretation. Since no single sensor directly measures positions and velocities of all moving objects in the environment, some form of data fusion is required to build an understanding of the environment incrementally over time. Techniques for doing so have evolved in several active areas of research, from target tracking for missile and aerospace defense to collision warning systems for automotive driver assistance [9, 87]. Among the various approaches adopted in these fields, two central problems are commonly addressed. The first is measurement assignment, where sensor data is divided and distributed among objects to account for the fact that no sensor perfectly segments each object in the environment. The second is feature extraction and tracking, where raw sensor data is preprocessed into detections of specific object features to be used as inputs in tracking algorithms. This latter problem is typically addressed to reduce large

amounts of raw sensor data into a small set of organized metadata.

Several successful methods have arisen in the target tracking literature for addressing the data assignment problem [9]. The simplest is *maximum likelihood* or *nearest neighbor* data assignment, where measurements are assigned to the objects according to a best-fit criterion. Many popular robotic simultaneous localization and mapping (SLAM) and detection and tracking of moving obstacles (DATMO) algorithms rely on this method of data assignment [90, 96, 92]. Although simple to implement, these approaches tend to make assignment mistakes in situations where objects are temporarily indistinguishable, such as when two moving objects cross paths. More accurate approaches acknowledge such situations, either by assigning measurements to more than one object in the case of the *joint probabilistic data association filter*, or by maintaining multiple assignment hypotheses in the case of *multiple hypothesis tracking* [9, 40, 10]. A recent class of approaches to gain momentum are *Monte Carlo data association* techniques, whereby Monte Carlo methods are used to randomly generate likely data assignments or complete tracking hypotheses [20, 76, 16].

On the other side of the problem, research in automotive driver assistance has yielded feature extraction and tracking algorithms suitable for full-size automotive environments. Many of the feature extraction algorithms have been designed specifically to operate on clouds of points generated by automotive laser rangefinders, including circle detection for pedestrian leg recognition, center of mass measurements for vehicle tracking, rectangle and bounding box fitting, corner detection, and line and segment detection for vehicle identification [82, 27, 28, 87, 29, 86, 24, 47, 53]. Other automotive algorithms combine features from complementary sensors, such as laser rangefinders and optical cameras, to

detect and track vehicles [50, 48, 49, 98].

From a robotics point of view, correct understanding of the dynamic environment requires simultaneous solutions for both the data assignment and the feature extraction and tracking problems. The challenge lies in the fact that these two problems are inextricably linked in a single joint estimation task, because no sensor can perfectly identify and separate all the objects in a populated dynamic environment. Many of the aforementioned automotive feature extraction and tracking algorithms ignore this fact for computational reasons, as it is combinatorially expensive to track multiple objects from detections of their respective parts. While such an assumption makes the automotive tracking problem more tractable, the instability of feature extraction algorithms yields erroneous tracking artifacts in most realistic automotive environments [47]. In contrast, the works of Maehlisch *et al.* specifically address the joint estimation problem with a *probabilistic hypothesis density filter*, which fuses an optical camera with a small laser rangefinder to simultaneously estimate the number of moving objects and their trajectories [50, 49]. Computationally, the expense of this and other integrated multitarget tracking algorithms limits tracking to an environment with a small number of targets, where most sensor data is rejected as clutter.

This paper presents the LocalMap algorithm, a practical multitarget tracking algorithm meant to approach data assignment and dynamic obstacle tracking as a single joint estimation problem under bearable computational load. The LocalMap algorithm extends the previous work of Miller and Campbell, in which simulations of a similar obstacle tracking algorithm were shown capable of simultaneously determining the number of obstacles in an environment and tracking those obstacles as they moved [54]. Preliminary experimental re-

sults for stable obstacle tracking with a single stationary laser rangefinder were also presented. Here, the same joint estimation techniques are extended to stable, accurate, real-time multisensor tracking of a large number of potentially dynamic obstacles from a potentially dynamic robot. Specifically, the LocalMap algorithm abandons commonly-used parameterized models of target geometry in favor of a richer point cloud representation and more stable measurement updates. The new representation yields accurate estimation in two practical but very difficult scenarios: when one or more dynamic obstacles occupy a large portion of sensor fields of view, and when the dynamic obstacles' observed geometry changes rapidly due to maneuvers at short sensor ranges. The LocalMap algorithm is derived to fuse three common types of full-size robotic sensors: laser rangefinders, radars, and optical cameras, though it is general to any mode of sensing. The capabilities of the LocalMap algorithm are verified with controlled field experiments conducted from full-size vehicles traveling at city speeds. Experimental results are also given for Cornell University's entry in the DARPA Urban Challenge, a 6 hour, 60 mile urban robotics challenge for which the LocalMap algorithm was implemented. Section 5.2 introduces the LocalMap algorithm, providing a theoretical foundation for the data assignment and obstacle tracking joint estimation problem. Section 5.3 presents experimental performance for the LocalMap algorithm, first in controlled full-size vehicle tracking experiments, and then in an uncontrolled environment full of traffic. Section 5.4 summarizes the performance of the LocalMap algorithm in the DARPA Urban Challenge, and Section 5.5 concludes.

5.2 The LocalMap Tracking Algorithm

Derivation of the LocalMap algorithm begins by casting the dynamic urban perception problem in a Bayesian framework. To that end, a set of variables O describing obstacle positions and motions are estimated to permit basic robotic navigation and obstacle avoidance. In this context, the cardinality of the obstacle variables O implicitly represents the number of obstacles in the world, and may change over time. In addition to the obstacle variables O, a set of data assignment variables A are also estimated. These assignment variables record the history of sensor measurements generated by each obstacle, thereby dividing the user robot's sensor measurements into historical sequences associated with each individual obstacle. Neither the obstacle variables O nor the assignment variables A are known with certainty in the unstructured environment, so both must be estimated. The LocalMap algorithm therefore estimates the joint probability density over these two sets of variables, conditioned on all available sensor measurements:

$$p\left(A_k, O_k | Z_k\right) \tag{5.1}$$

where Z_k are the set of observed sensor measurements, and the subscript k represents an integer time index. The use of capital letters A_k, O_k, and Z_k indicates a full time history of these quantities, from the filter's inception at time index 0 to the present time index k.

Though general filtering methods exist to estimate joint probability densities over multiple variables, the case of perception of a dynamic urban environment on a mobile robot presents several difficulties. First, the probability density in equation 5.1 is hybrid, because the data assignment variables A are discrete while the obstacle variables O are continuous. Second, the number of poten-

tial data assignment histories grows exponentially in time, number of measurements, and number of obstacles, preventing exact probabilistic reasoning even in modest environments. A final challenge is the fact that obstacles moving in a dynamic urban environment may be intelligent, capable of executing complex nonlinear maneuvers over time. These three challenges in concert make most traditional estimation approaches unsuitable for estimating the joint probability density in equation 5.1.

To help make the estimation problem tractable amidst these challenges, the joint probability density is first factorized. The definition of conditional probability is employed to rewrite the joint probability density in equation 5.1:

$$p\left(A_k, O_k | Z_k\right) = p\left(A_k | Z_k\right) \cdot p\left(O_k | Z_k, A_k\right) \tag{5.2}$$

This factorization is exact, and is similar to one made by Montemerlo *et al.* to estimate the joint density between an ego robot's pose and the positions of static landmarks in a variant of the SLAM problem [61]. Intuitively, the factorization made in equation 5.2 is beneficial because it decouples the discrete data assignment estimation problem $p\left(A_k | Z_k\right)$ from the continuous tracking problem $p\left(O_k | Z_k, A_k\right)$. Although the discrete data assignment problem is still too large to be solved with exact inference techniques, the continuous tracking problem conditioned on known data assignments may be solved with less expensive parametric filters such as the Kalman Filter (KF) or Sigma Point Filter (SPF) [11, 9, 38].

5.2.1 The Discrete Data Assignment Problem

Despite the factorization made in equation 5.2, the number of discrete data assignment permutations still grows exponentially in time, number of obstacles, and number of measurements. It is therefore computationally infeasible to evaluate the corresponding data assignment density $p(A_k|Z_k)$ exactly, so approximation techniques are used instead. For the LocalMap algorithm, the discrete data assignment density $p(A_k|Z_k)$ is approximated by a small number of randomly-drawn samples, utilizing well-worn Monte Carlo likelihood-weighted sampling techniques [74, 7]. The goal of the factorization made in equation 5.2 is to make these non-deterministic data assignment choices 'obvious,' reducing sample size requirements through the use of intelligent but inexpensive parametric solutions to the continuous tracking problem.

This type of hybrid particle filter, which estimates a factorized probability density by combining Monte Carlo sampling techniques with closed-form parametric filters, is known as a Rao-Blackwellized Particle Filter (RBPF); see, for example, Doucet *et al.* [21]. Särkkä *et al.* have studied such an estimator in the context of target tracking from a single fixed sensor, where it is shown in simulation to be robust against target confusion and other classically challenging data assignment problems [76]. The LocalMap extends that RBPF framework to track the motion of dynamic obstacles in an urban environment relative to a moving sensing platform, i.e. a ground robot, where nearby obstacles occupy much of the field of view and viewpoints of these obstacles change rapidly. The approach offers a more rigorous Bayesian solution to the joint estimation problem than maximum likelihood data assignments, and the solution is made practical through the RBPF's computationally efficient factorization of data assignment

and obstacle tracking.

In the LocalMap, as with other particle filters derived for Monte Carlo data assignment, each randomly-drawn particle stores one possible data assignment history [7]. The particles are drawn according to a *proposal density* $q\left(A_k|Z_k\right)$, selected for its efficient sampling algorithms and its similarity to $p\left(A_k|Z_k\right)$. The particles drawn from this proposal density represent an approximation of the true density $p\left(A_k|Z_k\right)$ for the purposes of approximate inference:

$$p\left(A_k|Z_k\right) \approx \sum_i w_k^i \cdot \delta\Big(A - A_k^i\Big) \tag{5.3}$$

where w_k^i is the likelihood weight of the i^{th} particle A_k^i at time index k:

$$w_k^i = \frac{p\left(A_k^i|Z_k\right)}{q\left(A_k^i|Z_k\right)} \tag{5.4}$$

and, because the data assignment problem is a discrete estimation problem, $\delta\left(\cdot\right)$ is the Kronecker delta function. Note the weights w_k^i must sum to unity to preserve the density's normalization:

$$\sum_i w_k^i = 1 \tag{5.5}$$

With equation 5.3 providing an approximate representation of $p\left(A_k|Z_k\right)$ and a closed-form parametric filter used to represent $p\left(O_k|Z_k, A_k\right)$, the full joint probability density from equation 5.2 is written approximately as:

$$p\left(A_k, O_k|Z_k\right) \approx \sum_i w_k^i \cdot \delta\Big(A - A_k^i\Big) \cdot p\left(O_k|Z_k, A_k^i\right) \tag{5.6}$$

where the obstacle densities $p\left(O_k|Z_k, A_k^i\right)$ are conditioned on the specific data assignment history A_k^i of a particular particle. In essence, each particle in equation 5.6 represents a complete hypothesis about the ego robot's urban environment. Each particle contains both a history of data assignment decisions as well as a parametric filter estimating states of obstacles whose existence are implied

by the particle's measurement assignments. Since each particle may have a different data assignment history, particles may have different estimates of obstacle states and may even have different numbers of obstacles. Regardless, the factorization has simplified the urban perception problem dramatically. By exploiting the success of parametric filters such as the KF and SPF for target tracking under known measurement assignments, the LocalMap only needs to draw particles over the discrete data assignment portion of the joint density $p\left(A_k, O_k | Z_k\right)$.

Further simplifications to the dynamic urban perception problem can be made by realizing that the ego robot operating in real-time only requires an estimate of its surroundings at the present time. The problem is therefore simplified considerably by deleting measurement assignment and obstacle state histories from each particle after the information has been incorporated into the current estimate. This may be done by modifying the proposal density $q\left(A_k^i | Z_k\right)$, nominally a design choice 'similar to' the true distribution $p\left(A_k | Z_k\right)$, to delete old information. In particular, if the proposal density $q\left(A_k | Z_k\right)$ is chosen to factorize as follows:

$$q\left(A_k | Z_k\right) = q\left(a_k | Z_k, A_{k-1}\right) \cdot q\left(A_{k-1} | Z_{k-1}\right) \tag{5.7}$$

then the likelihood weight from equation 5.4 can be expressed recursively [7]:

$$w_k^i \propto \frac{p\left(z_k | Z_{k-1}, A_k^i\right) \cdot p\left(a_k^i | Z_{k-1}, A_{k-1}^i\right)}{q\left(a_k^i | Z_k, A_{k-1}^i\right)} \cdot w_{k-1}^i \tag{5.8}$$

where lowercase variables z_k and a_k indicate measurements and data assignments at a particular time index k. Equation 5.8 has been obtained, up to a normalization constant, by applying Bayes's rule to the data assignment density $p\left(A_k | Z_k\right)$. Notice that both terms in the numerator of equation 5.8 are familiar quantities: $p\left(z_k | Z_{k-1}, A_k^i\right)$ is the joint filter likelihood of the tracked obstacles after the k^{th} data assignment decision in the i^{th} particle, and $p\left(a_k^i | Z_{k-1}, A_{k-1}^i\right)$

is the predicted joint likelihood of the k^{th} data assignment decision in the i^{th} particle, before the k^{th} measurement has actually been made.

One final step remains in deriving the particle filter for data assignment, and that is choosing the proposal density $q\left(a_k^i|Z_k, A_{k-1}^i\right)$ from which the random data assignment hypotheses for each particle are drawn. Because the density defines how the particles are sampled, it should be a density that can be sampled inexpensively. In particle filtering applications that estimate the state of a dynamic system, the proposal density is often chosen based only on the system's transition model and its process noise, thereby ignoring the measurement history [7]. This choice is suboptimal in the sense that it does not minimize the sample variance on the particles' weights w_k^i, which more rapidly drives all but a few particle weights to zero under repeated renormalizations. In contrast, the optimal proposal density $q_{\text{opt}}(\cdot)$ minimizing the sample variance on the particles' weights has been shown to be the true density [7]. For the LocalMap's data assignment problem, this density is:

$$q_{\text{opt}}\left(a_k^i|Z_k, A_{k-1}^i\right) = \alpha_k^i \cdot p\left(a_k^i|Z_k, A_{k-1}^i\right) \tag{5.9}$$

where α_k^i is a normalization constant explicitly included to make clear that the proposal density must sum to unity across the set of available data assignments in the i^{th} particle. In the special case of data assignment, the optimal proposal density may be sampled directly and inexpensively [76]. To derive the corresponding sampling algorithm, the optimal density is first rewritten using Bayes's rule:

$$\begin{aligned} p\left(a_k^i|Z_k, A_{k-1}^i\right) &= \alpha_k^i \cdot p\left(z_k|a_k^i, Z_{k-1}, A_{k-1}^i\right) \\ &\cdot \quad p\left(a_k^i|Z_{k-1}, A_{k-1}^i\right) \end{aligned} \tag{5.10}$$

where α_k^i is the normalizing constant, potentially different for each particle, that

ensures all data assignment probabilities in the i^{th} particle sum to unity. The first of the remaining two terms, $p\left(z_k|a_k^i, Z_{k-1}, A_{k-1}^i\right)$, is the likelihood of the measurement z_k, assuming it originated from a particular tracked obstacle in the i^{th} particle. This term is nothing more than the filter likelihood for the tracked obstacle, which is often accurately assumed to be Gaussian in parametric filters [11]. The second term, $p\left(a_k^i|Z_{k-1}, A_{k-1}^i\right)$, is a transition model for the data assignments. This term represents any *a priori* assignment information that may be present before the measurement at time index k is actually made. In the present implementation of the LocalMap, this term is assigned a uniform probability across all data assignments. This choice is made to represent the fact that sensors are fused asynchronously in the LocalMap, and the measurements in general have no predictable ordering.

Equation 5.10 thus allows each particle to choose a random data assignment for the measurement z_k with assignment probabilities proportional to the measurement's likelihood of corresponding to each tracked obstacle in the particle. More precisely, the optimal proposal density from equation 5.10 is substituted into equation 5.8 to yield the final form of the weight update:

$$w_k^i = w_{k-1}^i \cdot \frac{1}{\alpha_k^i} \tag{5.11}$$

The remaining constant α_k^i is the normalizing constant for the optimal proposal density $q_{\text{opt}}(\cdot)$ from equations 5.9 and 5.10, left over after computing the likelihood ratio in equation 5.8 with the optimal proposal density. In the specific case where there is no prior information about the next measurement assignment, i.e. $p\left(n_k^i|Z_{k-1}, N_{k-1}^i\right)$ is uniform, the normalization constant is:

$$\alpha_k^i = \left[\frac{1}{M} \cdot \sum_{m=1}^{M} p\left(z_k|a_{m,k}^i\right)\right]^{-1} \tag{5.12}$$

where the sum is performed across all M obstacles tracked within the i^{th} particle, and $a^i_{m,k}$ is the event that the measurement z_k taken at time k is assigned to the m^{th} of M obstacles in the i^{th} particle. Notice this final form of the LocalMap's weight update has the satisfying interpretation that particles whose obstacles better match the received sensor measurements have higher weights.

The sampling and reweighting calculations complete the data assignment particle filter defined by equation 5.3 and the RBPF defined by equation 5.6. A high-level description of the algorithmic steps taken to run the LocalMap RBPF for data assignment and obstacle tracking are given below:

1. Draw an initial set of particles $A^i_0, \forall i \in [1, N]$.
2. Predict all obstacles in each particle forward in time to the next measurement to yield a parametric representation of $p\left(O_k | Z_{k-1}, A^i_{k-1}\right)$.
3. For each particle, pick a data assignment for the measurement using the optimal proposal density in equation 5.10.
4. Update the parametric tracking filter to yield $p\left(O_k | Z_k, A^i_k\right)$ for the obstacle in each particle chosen to receive the measurement.
5. Update particle weights according to equation 5.11.
6. Resample particles to keep the filter well-conditioned, if necessary. Effective resampling strategies are discussed in Arulampalam *et al.* and Grisetti *et al.* [7, 33].
7. Go to step 2.

Occasionally, the LocalMap performs step 6, a particle resampling step, to alleviate degeneracy problems. These common particle filtering problems occur over time due to unlucky sampling choices that cause particle weights

to degrade asymptotically to zero over repeated renormalizations [7]. In the LocalMap, such a problem might arise over time as the particles continue to sample random data assignment decisions without a means of correcting poor choices. If left unchecked, all particles in the LocalMap would eventually make poor data assignment decisions, leaving the filter with no particularly good hypothesis among its particles. To avoid this problem, the LocalMap's particles are occasionally resampled into a new set of particles with probability proportional to their weight. Particles with higher weights, which correspond to more likely interpretations of the urban environment, are more likely to be sampled into the new set of particles one or more times. Particles with lower weights, which correspond to unlikely hypotheses, may not even be sampled into the new set of particles at all. To decide when to resample, the LocalMap adopts the method proposed by Grisetti *et al.* for estimating the effective number of particles $\hat{N}_k$ that substantially influence the shape of the filtered density at time index k [33]:

$$\hat{N}_k = \frac{1}{\sum_i \left(w_k^i\right)^2} \tag{5.13}$$

For normalized particle weights drawn over N particles, equation 5.13 implies that $1 \leq \hat{N}_k \leq N$. Intuitively, $\hat{N}_k$ achieves its maximum when all particle weights are equal, which occurs when the particle filter assigns equal likelihood to each of its particles. Similarly, $\hat{N}_k$ achieves its minimum when all but one weight are zero, the undesirable situation in which the particle filter effectively places all likelihood on only one of its hypotheses. Grisetti *et al.* suggest thresholding $\hat{N}_k$ to determine whether resampling is necessary [33]. The LocalMap adopts this approach, resampling the particles when $\hat{N}_k \leq N/2$.

Thus far, it has been assumed that each particle is given a list of preordained obstacles O_0^i at $k = 0$ that are all present and visible from the moment the LocalMap is first started. In many practical situations this *a priori* information is

not available, and instead the LocalMap must begin with a set of N empty particles: $O_0^i = \emptyset, \forall i \in [1, N]$. The creation of new obstacles is then handled within the particle filtered data assignment framework. In particular, a birth likelihood $p\left(z_k | a_{m=0,k}^i\right)$ can be defined for each measurement z_k to represent the likelihood that the measurement originates from a newly-visible obstacle. The LocalMap's RBPF then considers this likelihood with those of any existing obstacles when choosing data assignments: the measurement is either assigned to an existing obstacle, or used to initialize a new one [54]. In contrast, the LocalMap does not remove obstacles from particles with a corresponding death likelihood, as each obstacle's existence depends on its state history as well as its assigned measurements. Obstacle death is therefore modeled in the LocalMap's parametric estimates of the obstacle density $p\left(O_k | Z_k, A_k\right)$, where, for practical reasons, it is implemented as a deterministic decision based on the obstacle's state and measurement history. The convenience of this formulation is that the LocalMap automatically determines the number of obstacles in the environment, and obstacles may pass in and out of sensor range without a growing computational burden.

5.2.2 The Continuous Tracking Problem

Section 5.2.1 describes the method used in the LocalMap to run an RBPF that samples a set of high likelihood data assignment hypotheses. This solution to the data assignment problem relies on the effectiveness of an inexpensive yet accurate parametric filter for tracking obstacles in the urban environment under known measurement assignments. The goal of this parametric filter is to describe obstacle geometry and motion as accurately as possible, making each

obstacle distinct and distinguishable for measurement assignment. This reduces computational effort in the RBPF by decreasing the number of high likelihood data assignment hypotheses, and accordingly, the expected number of particles required to fully sample those hypotheses.

To further reduce the computational burden of the problem, all obstacles are assumed to be conditionally independent given sensor measurements and their assignments. This allows the continuous tracking problem to be factorized into a set of single obstacle tracking problems:

$$p\left(O_k|Z_k, A_k^i\right) = \prod_m p\left(O_{m,k}|Z_k, A_k^i\right) \tag{5.14}$$

where $p\left(O_{m,k}|Z_k, A_k^i\right)$ is the probability density of the m^{th} obstacle in the i^{th} particle. By assuming conditional independence among the obstacles, each particle in the LocalMap uses a bank of small parametric filters, one for each obstacle tracked. This factorization offers substantial computational savings, since parametric filters often rely on expensive matrix operations cubic in the size of the filter's state vector. The alternative, a single large parametric filter simultaneously estimating the joint state of all obstacles in a particle, would be prohibitively expensive. The parametric filter described in this section therefore sets out to track a single obstacle, potentially moving, under known measurement assignments.

Since the LocalMap is designed to solve the dynamic urban perception problem, all obstacles are assumed to have size, shape, and motion similar to full-size motor vehicles. This assumption is made to constrain the LocalMap's attention to vehicular traffic, though it does not preclude an expanded list of obstacle classes including pedestrians, bicyclists, tractor-trailers, and other dynamic objects commonly present in urban environments. The benefit of adopting the former constraint is that it avoids an ancillary target classification problem, as

each obstacle is tracked with an instance of the same parametric filter.

The most common parametric filter used for tracking a single full-size moving vehicle models the vehicle as a rectangle [87, 86, 54, 24, 98]. Such a filter typically estimates the vehicle's length and width in addition to motion parameters such as position and velocity. Problems with this approach stem from two sources: the fact that vehicles are not rectangles, and the fact that no sensor measures the entire vehicle as if it were. Unfortunately, these two problems have competing solutions: the rectangle model is not rich enough to describe a moving vehicle, but it is too complex to permit simple Bayesian updates. Many existing approaches strike a compromise by accepting the former problem and applying an *ad hoc* feature extraction or matching algorithm to raw sensor data to address the latter [87, 27, 86, 24]. However, as MacLachlan and Mertz point out, these feature extraction techniques yield erroneous motion estimates due to the substantial change in the sensed shape of a vehicle as it moves around the sensor's field of view [47].

The LocalMap tracking algorithm avoids these problems with a filter that operates directly on raw sensor data. The filter contains five state variables describing the position, motion, and shape of the moving vehicle in a coordinate frame fixed to the ego robot. Figure 5.1 shows how these state variables are used to define the moving vehicle within the parametric filter. Internally, the filter's tracked vehicle is stored as two pieces of data: a parameterized probability density $p\left(O_{m,k}|Z_k, A_k^i\right)$ over the vehicle's state $O_{m,k}$, and a cloud of sensed points describing the vehicle's shape. Because it is sensed rather than parameterized, this point cloud representation stores a far more accurate description of the tracked vehicle's geometry than more common rectangular models. A

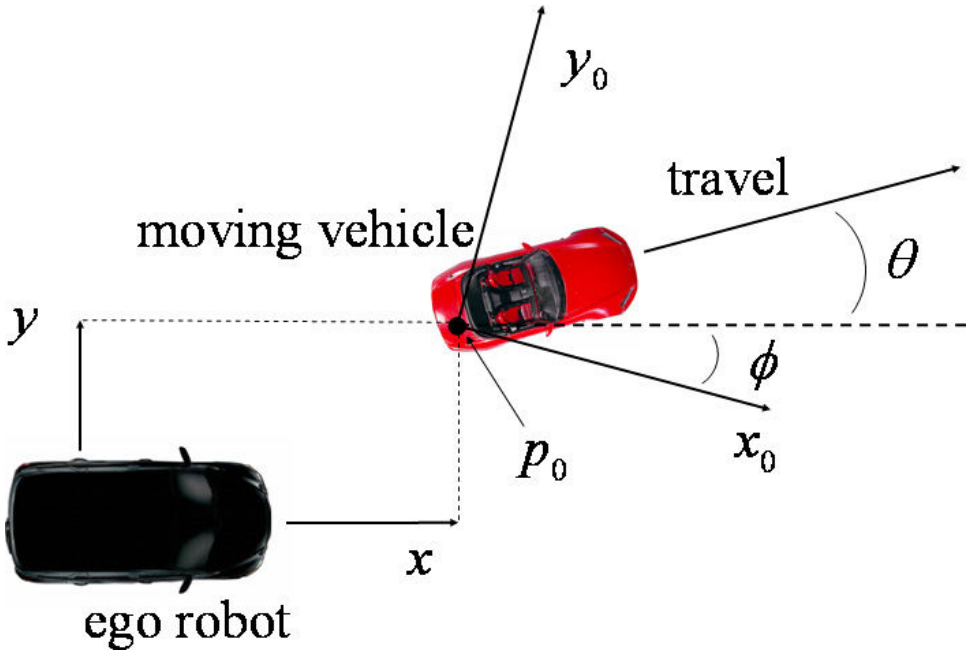

Figure 5.1: Coordinate frames used for tracking a single obstacle (a moving vehicle) under known measurement assignments. In the LocalMap, an obstacle's position is defined by the x and y location of a reference point p_0 fixed to the obstacle.

more subtle observation is the fact that integrating the point cloud representation is no more expensive than integrating parameterized and highly simplified vehicle geometries, because the tracked vehicle is a rigid body. As a result, the position and motion of all points in the cloud are related through an unchanging set of affine transformations. If the relative positions of these points are stored in an obstacle-fixed coordinate frame, the motion of one such reference point in the ego robot frame is sufficient to reconstruct the motion of all points. A further key observation is that direct measurements of this one target-fixed reference point, labeled p_0 in Figure 5.1, are not necessary to accurately estimate the motion of the target vehicle. Instead, the position of the fixed reference point is included in the set of states to be estimated. The resulting tracking filter produces consistent motion estimates by utilizing sensor information within the

Bayesian framework to simultaneously estimate the location and motion of an arbitrary fixed reference point on the vehicle being tracked. Because the filter does not rely on measuring a *specific* fixed reference point, such as the tracked vehicle's center of mass, it avoids estimation artifacts that arise from *ad hoc* attempts to locate these specific points in raw sensor data.

The proposed filter parameterizes a tracked vehicle's position, motion, and shape with five states:

$$o_{m,k} = \begin{bmatrix} x_k & y_k & s_k & \theta_k & \phi_k \end{bmatrix}^T \tag{5.15}$$

where, from now on, $o_{m,k}$ will be used instead of $O_{m,k}$ to reflect the desire to estimate the state of the tracked vehicle only at the current time step k. From Figure 5.1, the position variables x and y describe the location of the tracked vehicle's arbitrary fixed point relative to the ego robot. The motion of the tracked vehicle's fixed point is parameterized by the velocity variables s and θ, which store the point's absolute ground speed and heading relative to the ego robot. The shape of the tracked vehicle is parameterized by ϕ, which describes the rigid body rotation angle between the ego robot's coordinate frame and the obstacle-fixed frame in which the tracked vehicle's point cloud is stored. Intuitively, ϕ accumulates the total change in the angle at which the vehicle is observed while it is being tracked. Note that if the tracked vehicle's point cloud is stored relative to the fixed point, the combined rotation ϕ and translation $\begin{bmatrix} x & y \end{bmatrix}^T$ suffice to locate any point in the point cloud with respect to the ego robot. Similarly, the transformation may be combined with the motion parameters s and θ to compute the motion of any point in the tracked vehicle's point cloud.

The time evolution of $o_{m,k}$ in the (potentially moving) coordinate frame fixed to the ego robot is modeled by the following system of continuous-time nonlin-

ear differential equations:

$$\dot{x} = s \cdot \cos(\theta) - v_x + y \cdot \omega_z + e_x \tag{5.16}$$

$$\dot{y} = s \cdot \sin(\theta) - v_y - x \cdot \omega_z + e_y \tag{5.17}$$

$$\dot{s} = e_s \tag{5.18}$$

$$\dot{\theta} = -\omega_z + e_\theta \tag{5.19}$$

$$\dot{\phi} = -\omega_z + e_\phi \tag{5.20}$$

where v_x and v_y are components of the ego robot's velocity, ω_z is the ego robot's rate of rotation, and e_x, e_y, e_s, e_θ, and e_ϕ are zero mean, mutually uncorrelated, Gaussian, white random variables acting as *process noise*. Intuitively, these random variables account for unmodeled maneuvers executed by the tracked vehicle. Note that although equation 5.18 assumes the tracked vehicle moves on average in a straight line at constant speed, any parameterized dynamics model may be used. This flexibility is a major benefit of the LocalMap's point cloud representation: the dynamics model merely describes the time evolution of the rigid body transformation between the coordinate frame fixed to the ego robot and the one fixed to the tracked vehicle.

In addition to the aforementioned process noise accounting for vehicle maneuvers, it is noted that v_x, v_y, and ω_z are measured by noisy sensors. As a result, the following substitutions are made:

$$v_x = \hat{v}_x + e_{v_x} \tag{5.21}$$

$$v_y = \hat{v}_y + e_{v_y} \tag{5.22}$$

$$\omega_z = \hat{\omega}_z + e_{\omega_z} \tag{5.23}$$

where $\hat{v}_x$, $\hat{v}_y$, and $\hat{\omega}_z$ are measured from odometry sensors on the ego robot, and e_{v_x}, e_{v_y}, and e_{ω_z} are zero mean, mutually uncorrelated, Gaussian, white random

variables acting as additional process noise. Regrettably, these noisy estimates of ego robot motion result in correlations between all obstacles tracked in the LocalMap, so that equation 5.14 is only an approximate factorization. In SLAM literature, where obstacles are static and often modeled with no process noise, such correlations are large and central to the localization problem [92]. In a dynamic environment, the opposite is true: obstacles are modeled with significant uncorrelated process noise to capture uncertainty in maneuvers such as acceleration and turning. This process noise effectively swamps correlations between obstacles, as uncertainty in their maneuvers is far larger than uncertainty in commonly-available automotive odometry sensors. As a result, the factorization in equation 5.14 is taken as a valid approximation.

To facilitate the use of a computationally inexpensive parametric filter, $p\left(o_{m,k}|Z_k, A_k^i\right)$ is assumed to be Gaussian. In light of the weak nonlinearities in equations 5.16 - 5.20, the time evolution of this Gaussian is computed in the LocalMap using the *prediction step* of the Extended Kalman Filter (EKF). The prediction step is implemented using a fourth order Runge-Kutta numerical integration to convert equations 5.16 - 5.20 to a nonlinear discrete time difference equation of the form:

$$o_{m,k+1} = f\left(o_{m,k}, v_{m,k}\right) \tag{5.24}$$

where $v_{m,k}$ is a vector of zero mean, mutually uncorrelated, Gaussian, white random variables derived from e_x, e_y, e_s, e_θ, e_ϕ, e_{v_x}, e_{v_y}, and e_{ω_z}. Traditional EKF equations are then used to compute the time evolution $p\left(o_{m,k+1}|Z_k, A_k^i, \hat{v}_{x,k}, \hat{v}_{y,k}, \hat{\omega}_{z,k}\right)$ conditioned on past measurements, assignments, and the measured motion of the ego robot [11]. Although conditioning on measurements of ego robot motion is henceforth suppressed for brevity, it is understood to be present in the time evolution of all obstacles.

When a new sensor measurement z_{k+1} is assigned to the tracked vehicle, the vehicle's posterior probability density can be updated to $p\left(o_{m,k+1}|Z_{k+1}, A^i_{k+1}\right)$ to reflect the new measurement. The exact form of the update depends greatly on the information contained in the measurement, and therefore on the type of sensor generating the measurement. Three popular automotive sensing modalities are explored for the LocalMap: laser rangefinder, radar, and optical camera. Measurement updates for these sensing modalities are discussed in Sections 5.2.2 and 5.2.2.

Laser Rangefinder

The first type of sensor fused in the LocalMap is the laser rangefinder, which measures a point cloud of returns generated from patches in the environment reflecting the sensor's emitted energy. Though laser rangefinders offer centimeter-level ranging and sub-degree bearing accuracy, a practical problem arises due to the fact that they measure individual points rather than entire objects. Existing approaches solve the problem by extracting features from the point cloud, such as lines, corners, rectangles, or the center of mass, and then using these features as measurements to update the parameterized probability density of the tracked vehicle [87, 29, 86, 24, 47, 53]. Unfortunately, these feature extraction algorithms are unstable in even mildly dynamic environments, where the shape and motion of tracked vehicles and pitch and roll of the ego robot cause rapid changes in object shape. Unmodeled instability in these features frequently yields erroneous state estimates, particularly in the motion of the tracked vehicle. Figure 5.2 illustrates the problem by plotting the locus of point cloud centers of mass observed as a moving vehicle passes in front of a stationary laser rangefinder.

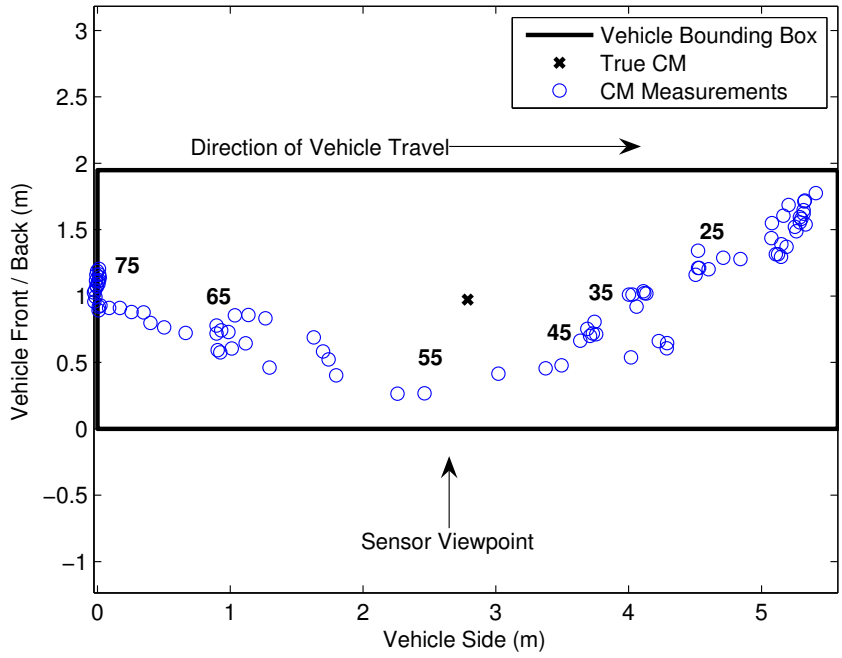

Figure 5.2: Locus of point cloud centers of mass observed as a moving vehicle passes in front of a stationary laser rangefinder. The apparent drift in the observed center of mass results from dramatic changes in the sensed shape of the moving vehicle as it passes in front of the laser rangefinder. This sensitivity, appearing in even mildly dynamic environments, causes instability in feature extraction algorithms such as center of mass, edge, and corner detectors applied to laser rangefinder data.

Here, the vehicle's motion causes the observed center of mass to drift with respect to the true center of mass. On average the observed center of mass lies more than 2 m from the true center of mass, and all measurements are at least 60 cm from the truth. Numeric indices of center of mass measurements in Figure 5.2 also show the motion of the observed measurements is largely opposite to the vehicle's direction of travel.

Two alternatives exist to alleviate estimation problems stemming from instability in sensed features. The first alternative is to model the instabil-

ity, by accounting for the extra uncertainty in the measurement likelihood $p\left(z_{k+1}|o_{m,k+1}, A^i_{k+1}\right)$. While this solution may yield consistent state estimates, it is unsatisfactory in the sense that the resulting estimation errors are larger to account for measurement instability. The second alternative, adopted for laser rangefinder data fused in the LocalMap, is to use an alternate set of measurements that are more reliably stable. First, the laser points are grouped into distinct objects via a clustering algorithm. The exact clustering algorithm used is not important, though the conservative algorithms presented in Miller and Campbell [54] or Miller *et al.* [56] yield stable performance in practice. After the laser points are clustered into distinct objects, three measurements are extracted: the smallest and largest bearings b_{min} and b_{max} from each cluster, and the range r_{min} to the closest point in each cluster. A measurement vector z_{k+1} consists of these three measurements extracted from a single cluster:

$$z_{k+1} = \begin{bmatrix} b_{\text{min}} & b_{\text{max}} & r_{\text{min}} \end{bmatrix}^T \tag{5.25}$$

and depending on the number of clusters present, multiple measurement vectors may be extracted from a single frame of laser returns.

The most important aspect of the bearing-bearing-range measurement described in equation 5.25 is its stability: the values of b_{min}, b_{max}, and r_{min} change slowly as the tracked vehicle's point cloud undergoes small translations and rotations. This property is not shared by other extracted features, such as the point cloud's center of mass, which may suffer large discontinuities in the face of small transformations. Such strong nonlinearities make those features unsuitable for linear estimation techniques.

In contrast, the weak nonlinearities in the bearing-bearing-range measurement make it ideal for use in linear estimation techniques, which linearize the

relationship between the measurement and the state vector. For the LocalMap, this relationship is made explicit with the following auxiliary variables:

$$\begin{aligned} \beta_{\text{min}} &= \min_{p \in P_{m,k+1}} \angle \left[T_s \left(x, y, \phi \right) \cdot \left(p - p_s \right) \right] \\ \beta_{\text{max}} &= \max_{p \in P_{m,k+1}} \angle \left[T_s \left(x, y, \phi \right) \cdot \left(p - p_s \right) \right] \\ \rho_{\text{min}} &= \min_{p \in P_{m,k+1}} \left\| T_s \left(x, y, \phi \right) \cdot \left(p - p_s \right) \right\| \end{aligned} \tag{5.26}$$

where β_{min}, β_{max}, and ρ_{min} constitute the state-generated measurement, $P_{m,k+1}$ is the set of points corresponding to the m^{th} tracked vehicle, $T_s \left(\cdot \right)$ is the transformation matrix that projects the point cloud $P_{m,k+1}$ into the coordinate frame of the laser rangefinder, p_s is the location of the laser rangefinder, and the operators $\angle \left(\cdot \right)$ and $\left\| \cdot \right\|$ return the bearing and magnitude of their vector arguments, respectively. With these auxiliary variables defined, the bearing-bearing-range measurement in equation 5.25 relates to the m^{th} tracked vehicle's state through the nonlinear *measurement function* $h_L \left(\cdot \right)$:

$$h_L \left(o_{m,k+1}, P_{m,k+1} \right) = \begin{bmatrix} \beta_{\text{min}} \\ \beta_{\text{max}} \\ \rho_{\text{min}} \end{bmatrix} \tag{5.27}$$

With the measurement function defined, the measurement likelihood $p \left(z_{k+1} | o_{m,k+1}, A^i_{k+1} \right)$ is modeled as Gaussian, corrupted by additive zero mean Gaussian white noise w_{k+1}:

$$z_{k+1} = h_L \left(o_{m,k+1}, P_{m,k+1} \right) + w_{k+1} \tag{5.28}$$

where the Gaussian measurement noise w_{k+1} is modeled as additive to reflect the fact that the laser rangefinder measures bearings and ranges directly. With $p \left(z_{k+1} | o_{m,k+1}, A^i_{k+1} \right)$ modeled as Gaussian, linear measurement updating techniques can be used to update $p \left(o_{m,k+1} | Z_k, A^i_{k+1} \right)$ to $p \left(o_{m,k+1} | Z_{k+1}, A^i_{k+1} \right)$ to reflect the new measurement. In particular, the LocalMap utilizes the Sigma

Point Transform in the update step of the SPF, because numerical differentiation of $h_L\left(o_{m,k+1}, P_{m,k+1}\right)$ is more convenient than explicit differentiation. This measurement is an ideal complement to the point cloud representation of the tracked vehicle, as neither rely on knowledge of a particular fixed point on the tracked vehicle. Instead, information about the tracked vehicle's motion and evolving rigid body transformation are gathered indirectly from robust measurements via fusion in a Bayesian tracking filter.

Radar And Optical Camera

The second and third types of sensors fused in the LocalMap are radar and optical cameras. Like the laser rangefinder, measurements from these sensors are fused in the LocalMap at the object level. In other words, the LocalMap relies on external processing to group raw sensor data into measurements of distinct objects. In the case of radar and optical camera data, this processing is commercially available; radar and camera systems built for collision detection often process raw data into objects. For the radar, the LocalMap utilizes measurements of the tracked vehicle's bearing, range, and range rate in the sensor's coordinate frame:

$$z_{k+1} = \left[\, b_s \quad r_s \quad \dot{r}_s \,\right]^T \tag{5.29}$$

and as with equation 5.25, multiple measurements may be available if more than one vehicle is present.

Much like the laser rangefinder, the radar measurement suffers from ambiguity. The radar's range and bearing components do not measure a particular fixed point on the tracked vehicle, and in fact the measured point may change over time as reflective portions of the vehicle become visible or occluded. The

difficulty is further compounded by the fact that the radar's algorithm for generating bearing and range measurements is often unknown, as many off-the-shelf radar units do not document their internal measurement processing algorithms. As a result, there is no way to relate the radar's measurements to the tracked vehicle's point cloud to eliminate the measurement ambiguity completely. Instead, the tracked vehicle's point cloud is used to generate an approximate measurement function $h_R\left(o_{m,k+1}, P_{m,k+1}\right)$ for the radar:

$$h_R\left(\cdot\right) = \begin{bmatrix} \frac{1}{2}\left(\beta_{\text{min}} + \beta_{\text{max}}\right) \\ \rho_{\text{min}} \\ \left\{v_v\left(s,\theta\right) - v_s\left(\hat{v}_x, \hat{v}_y, \hat{\omega}_z\right)\right\} \cdot e_r\left(x,y\right) \end{bmatrix} \tag{5.30}$$

where $v_v\left(\cdot\right)$ is the velocity of the tracked vehicle, $v_s\left(\cdot\right)$ is the velocity of the sensor on the ego vehicle, and $e_r\left(\cdot\right)$ is the unit vector from the radar to the tracked vehicle. As with the laser rangefinder, the measurement likelihood $p\left(z_{k+1}|o_{m,k+1}, A^i_{k+1}\right)$ is modeled as Gaussian, corrupted by additive zero mean Gaussian white noise w_{k+1}:

$$z_{k+1} \approx h_R\left(o_{m,k+1}, P_{m,k+1}\right) + w_{k+1} \tag{5.31}$$

where in approximation, the radar measures the center bearing, the closest range, and the range rate of the tracked vehicle's arbitrary fixed point. The ambiguity of the radar measurement is then addressed in the covariance matrix of the measurement noise w_{k+1}, where bearing and range measurement noise standard deviations are set large enough to account for the fact that the measurements can correspond to any point on the tracked vehicle. As with the laser rangefinder, the Sigma Point Transform of the SPF update is then used to generate $p\left(o_{m,k+1}|Z_{k+1}, A^i_{k+1}\right)$ from the radar measurement.

Optical camera measurements are incorporated into $p\left(o_{m,k+1}|Z_k, A^i_{k+1}\right)$ similar to radar measurements. For the optical camera, the LocalMap utilizes mea-

surements of the tracked vehicle's position, range rate, and width in the sensor's coordinate frame:

$$z_{k+1} = \begin{bmatrix} x_s & y_s & \dot{r}_s & w_s \end{bmatrix}^T \tag{5.32}$$

where (x_s, y_s) is the tracked vehicle's location, $\dot{r}_s$ is the vehicle's range rate, and w_s is the width of the tracked vehicle in the image plane.

Unlike the laser rangefinder and the radar, the optical camera measurement is not necessarily ambiguous. If edge or symmetry kernels are used to find the tracked vehicle's centerline as a position measurement, then the measurement is not ambiguous. In fact, the sensor-driven measurement of the centerline bearing b_{avg} is related to the minimum and maximum bearing measurements used in equation 5.25:

$$b_{\text{avg}} = \frac{1}{2}\left(b_{\text{min}} + b_{\text{max}}\right) \tag{5.33}$$

if the edge or symmetry kernels discover the same boundary as the laser rangefinder clustering algorithm. In contrast, the model-driven centerline bearing β_{avg} from the point cloud representation is:

$$\beta_{\text{avg}} = \frac{1}{2}\left(\beta_{\text{min}} + \beta_{\text{max}}\right) \tag{5.34}$$

Using β_{avg} and the other stable measurements, the optical camera's measurement function $h_C\left(o_{m,k+1}, P_{m,k+1}\right)$ is:

$$h_C\left(\cdot\right) = \begin{bmatrix} \rho_{\text{min}} \cdot \cos\left(\beta_{\text{avg}}\right) \\ \rho_{\text{min}} \cdot \sin\left(\beta_{\text{avg}}\right) \\ \left\{v_v\left(s,\theta\right) - v_s\left(\hat{v}_x, \hat{v}_y, \hat{\omega}_z\right)\right\} \cdot e_r\left(x,y\right) \\ 2 \cdot \rho_{\text{min}} \cdot \tan\left(\frac{1}{2}\left\{\beta_{\text{max}} - \beta_{\text{min}}\right\}\right) \end{bmatrix} \tag{5.35}$$

Using the measurement function, the camera measurement likelihood $p\left(z_{k+1} | o_{m,k+1}, A^i_{k+1}\right)$ is also modeled as Gaussian, corrupted by additive zero mean Gaussian white noise w_{k+1}:

$$z_{k+1} = h_C\left(o_{m,k+1}, P_{m,k+1}\right) + w_{k+1} \tag{5.36}$$

and the Sigma Point Transform of the SPF update is used to generate $p\left(o_{m,k+1}|Z_{k+1}, A^i_{k+1}\right)$ from the camera measurement.

5.3 Experimental Performance

The LocalMap tracking algorithm has been implemented in real-time on Cornell University's 'Skynet,' an autonomous 2007 Chevrolet Tahoe shown in Figure 5.3. Skynet is equipped with 7 laser rangefinders: 3 Ibeo ALASCA XTs in the front bumper, 1 SICK LMS 220 in the back bumper, 2 SICK LMS 291s in the rear driver and passenger doors, and a Velodyne HDL-64E on the roof. The placement and coverage of these laser rangefinders is shown in Figure 5.4. Skynet is also equipped with 8 Delphi FLR millimeter-wave radar units: 5 in the front bumper, and 3 in the back bumper. The placement and coverage of these radars is shown in Figure 5.5. Finally, Skynet is equipped with 2 optical cameras: a forward-facing Basler A622F, and a backward-facing Unibrain Fire-i 520b, both mounted on the roof and running MobilEye SeeQ vehicle tracking software. All sensors are accurately time-stamped via synchronized microcontroller interfaces, and each is available for obstacle detection and tracking in the LocalMap [56]. Skynet is also equipped with a position, velocity, and attitude estimator that fuses GPS with onboard vehicle odometry and inertial navigation to generate accurate and robust localization and differential vehicle motion estimates [59].

For evaluation purposes, the LocalMap has been implemented in C++ and connected to Skynet's time-stamped data logs. In this practical implementation, the LocalMap is initialized with no prior information. A birth likelihood

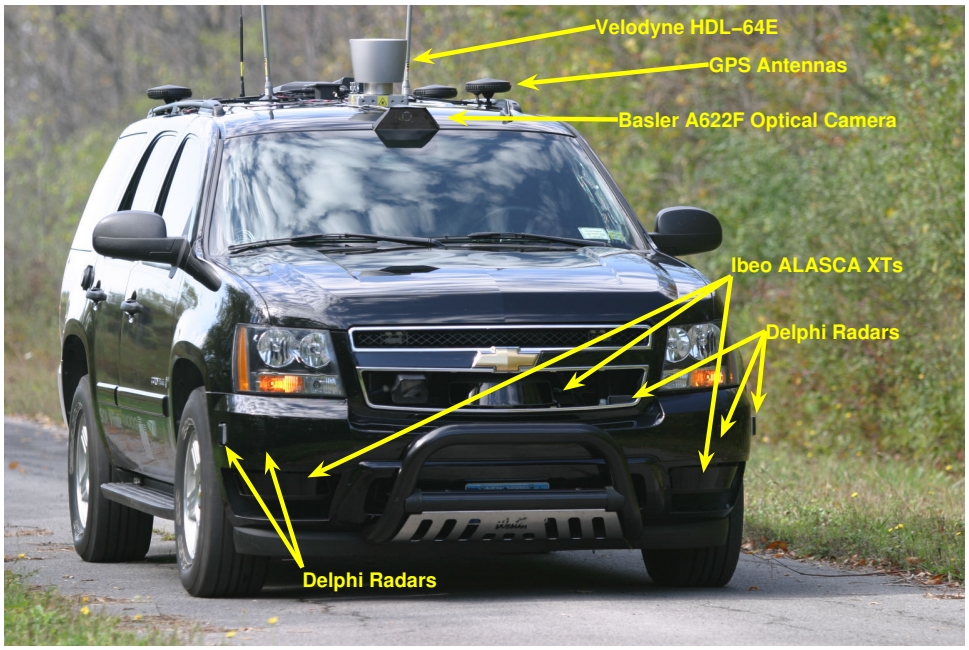

Figure 5.3: The LocalMap is implemented in real-time on Cornell University's 'Skynet,' an autonomous 2007 Chevrolet Tahoe that completed the 2007 DARPA Urban Challenge.

is then used to discover new obstacles in the RBPF framework as per Section 5.2.1, with sensor-dependent uniform likelihoods used to represent the likelihood of observing a new obstacle from each of Skynet's sensors. Similar to the birth model, each measurement is also evaluated against a clutter model created for each of Skynet's sensors. The corresponding clutter likelihood captures common sensor errors with uniform, Gaussian, and multi-modal Gaussian densities that account for multiple reflections, signal multipath, and false positives in each sensor. Deletion of old obstacles is similarly tied to the Bayesian framework, but implemented with deterministic thresholds to guard against the danger of randomly deleting a threatening obstacle. Each obstacle's existence is modeled as a probability that decays exponentially to zero with a 95% time constant of 3 seconds. Each measurement assigned to an obstacle provides evidence, in a Bayesian sense, for that obstacle's existence, and an occupancy grid

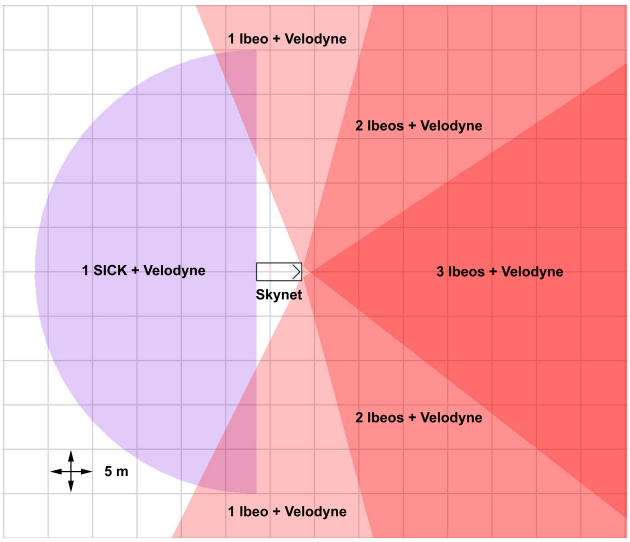

Figure 5.4: Sensor placement and coverage diagram for Skynet's laser rangefinders. Skynet sits in the center of the diagram, facing right. Redundant laser rangefinders cover the front of Skynet's field of view, where it encounters the most dangerous moving obstacles.

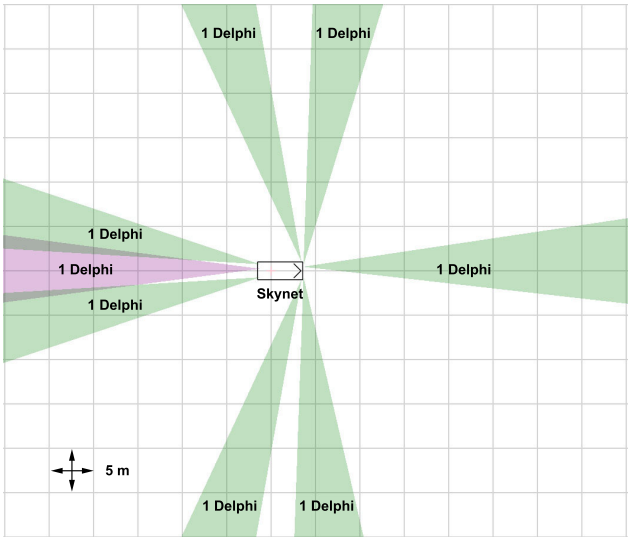

Figure 5.5: Sensor placement and coverage diagram for Skynet's radars. Skynet sits in the center of the diagram, facing right. Radars are placed to detect oncoming cars in opposing lanes, from the left and right in merging situations, and from the rear in passing situations.

created from Skynet's Velodyne provides evidence against false positives. Obstacles are removed from the LocalMap when their existence probability drops below 5%.

Specific practical accommodations are also made to adapt the point cloud representation across the LocalMap's three sensing modalities. Since neither the Delphi radars nor the MobilEye SeeQ software generate point clouds, obstacles that have not yet been assigned laser rangefinder measurements are tracked with parametric filters only, i.e. with no point clouds. In addition, new point clouds assigned to an obstacle always overwrite existing point clouds, but only after the measurement update is performed in the obstacle's parametric filter. In other words, no attempt is made to merge multiple point clouds over time; only the obstacle's rigid body transform and motion parameters undergo Bayesian estimation. Two reasons support this design choice: the accuracy and scanning rates of laser rangefinders make it unnecessary to combine multiple point clouds, and the computational burden required to store and process increasingly large point clouds would quickly overwhelm any real-time implementation.

The LocalMap is tested in a series of three experiments. The first two experiments evaluate the tracking capabilities of the LocalMap in common but difficult maneuvers: a perpendicular intersection encounter similar to the circumstances of Figure 5.2, and a parallel head-on encounter. In these experiments, the LocalMap is evaluated against truth data obtained from pose estimators on board the ego vehicle and the target vehicle being tracked. These estimators fuse traditional GPS and differential corrections with inertial navigation sensors on board both vehicles to produce position, velocity, and attitude esti-

mates for both the ego and target vehicle with sub-meter position accuracy, cm / sec. velocity accuracy, and sub-degree attitude accuracy [59]. The LocalMap simultaneously tracks other obstacles in the environment aside from the target vehicle, though ground truth for these obstacles is unavailable and therefore not evaluated. These experiments are performed at the Seneca Army Depot in Romulus, New York, which contains a variety of features including paved and unpaved roads, some painted road lines, potholes, railroad tracks, considerable short and tall vegetation, and storage buildings.

The third experiment evaluates the LocalMap's consistency in tracking multiple obstacles in a densely-populated, highly dynamic environment. In this experiment, variants of the LocalMap algorithm with different numbers of particles are run on the same segment of logged data, a 19 minute excerpt of a DARPA Urban Challenge qualifying round at George Air Force Base in Victorville, California. The data contains Skynet's sensor measurements of two concentric loops of heavy traffic traveling in opposite directions, recorded as Skynet merged into and out of this traffic multiple times. No truth data is available for the qualifying round, so consistency among the LocalMap variants is used to evaluate their performance.

5.3.1 Experiment 1: Perpendicular Intersection Encounter

The first experiment evaluates the LocalMap's ability to track a moving target vehicle as it crosses the ego vehicle's path at an intersection. In this experiment, the ego vehicle (Skynet) remains stationary at the intersection, while the target vehicle drives past at approximately 15 mph. The LocalMap is run on the

Figure 5.6: In the first experiment, the LocalMap tracks a moving target vehicle as it crosses the ego vehicle's path at an intersection.

collected sensor data with 50 particles, and each particle is allowed to track as many obstacles as it sees fit. Data is collected for a total of 22 perpendicular intersection encounters: 11 with the target vehicle approaching from the right, and 11 with the target vehicle approaching from the left. Figure 5.6 shows the experimental setup.

When evaluating the LocalMap's tracking performance on this data, comparisons are made to the LocalMap's maximum *a posteriori* (MAP) estimate of the environment, which is the particle with the largest weight. Within the most likely particle, evaluations are made with the tracked obstacle that most closely matches the truth data. This obstacle is chosen according to a minimum Mahalanobis distance criterion weighing minimum range, and minimum and maximum bearing. Weights are chosen such that a simultaneous range error of 5 m and bearing errors of 5° yield unit distance. Any LocalMap iterations in which

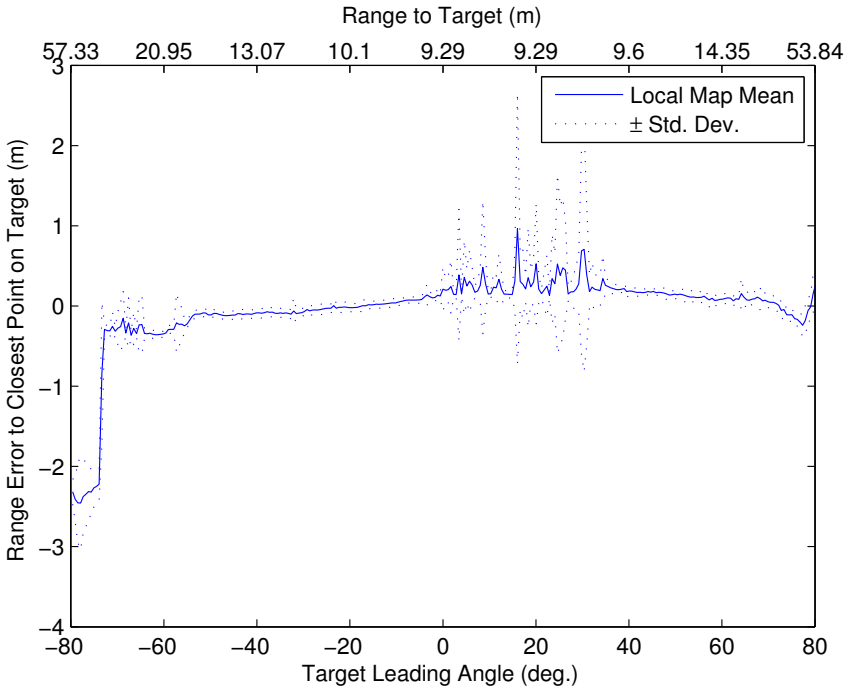

Figure 5.7: LocalMap range tracking errors to the closest point on a moving target vehicle over 11 perpendicular intersection encounters with the target approaching from the right at 15 mph. Error statistics are calculated across encounters according to the true bearing of the target's leading edge. Of the 284 average range errors considered, 251 are within 20 cm of zero at the 5% significance level.

no tracked obstacle has distance ≤ 7.8147, which corresponds to a 95% confidence bound on a χ^2 random variable with 3 degrees of freedom, are considered missed detections and discarded. A total of 11 such missed detections occurred over the 28100 LocalMap iterations considered in this experiment.

LocalMap tracking statistics averaged over the 11 from-the-right encounters are shown in Figures 5.7, 5.8, and 5.9. In each Figure, statistics are parameterized by the true bearing of the target vehicle's front bumper to align the 11 trials despite minor variations in maneuver duration. The Figures read left to right: the target vehicle approaches the intersection at negative bearings, crosses the

ego vehicle's path at bearing zero, and departs the intersection at positive bearings. The target vehicle is closest to the ego vehicle near bearing zero. Figure 5.7 plots average LocalMap errors in range to closest point on the moving target vehicle observed in the 11 intersection encounters. Predictably, ranging errors are smallest in the middle of each encounter, when the target vehicle is within 40 m (between -75° and 77°) of the ego vehicle and visible both by side-facing radars and laser rangefinders. Ranging errors increase slightly between approximately 0° and 40°, where the target vehicle is observed by Skynet's forward-facing radar. This radar occasionally provides erroneous information during perpendicular intersection encounters, as the target vehicle only travels perpendicular to its radial direction. These significantly incorrect measurements yield larger errors temporarily in one or two encounters, resulting in larger sample standard deviations. Errors are largest at the beginning of the maneuver, when the target vehicle is first observed between 60 and 80 m from the ego vehicle. Statistically, LocalMap ranging errors are within 20 cm of zero at the 5% significance level in 251 of the 284 bearings considered.

Figure 5.8 plots average LocalMap target vehicle ground speed estimation errors over the 11 from-the-right perpendicular intersection encounters. Speed estimation errors remain low from first acquisition through its approach, showing the LocalMap's ability to combine accurate radar speeds with accurate laser ranges to produce an obstacle estimate accurate in both speed and position. Accurate speeds are maintained even as the target vehicle crosses in front of the ego vehicle and out of view of the radars. Small speed errors are incurred temporarily as the target vehicle departs the intersection, where Skynet's left Ibeo laser rangefinder is the primary source of laser data. Errors incurred at these bearings appear to be the result of an angular miscalibration in the of-

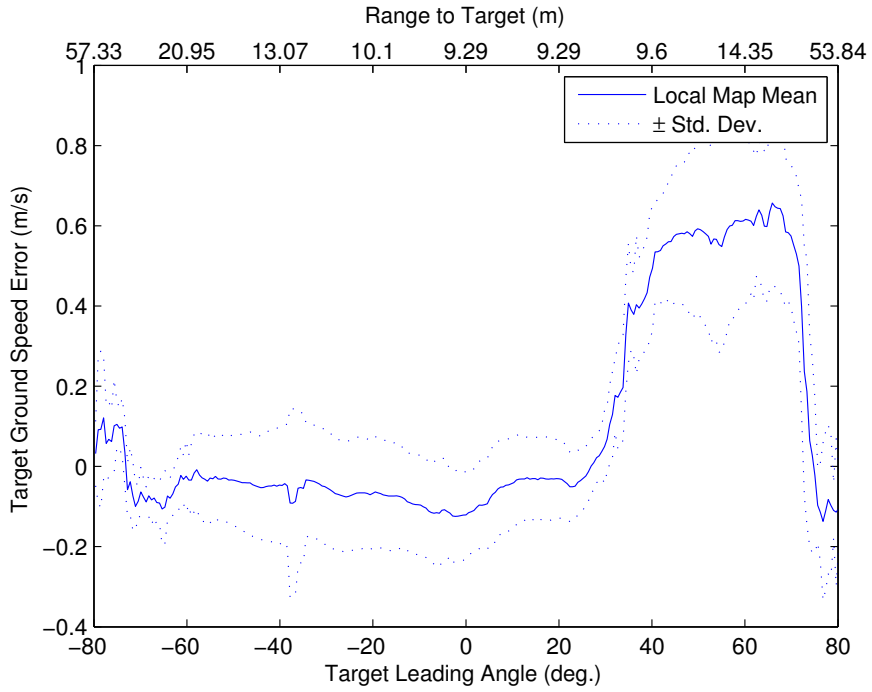

Figure 5.8: LocalMap ground speed tracking errors over 11 perpendicular intersection encounters with a moving target vehicle approaching from the right at 15 mph.

fending laser rangefinder, resulting in disagreement between Skynet's left and center laser rangefinders. Since the disagreement is in sensor yaw, it creates an unmodeled bias in the evolution of the target vehicle's point cloud while minimally affecting Skynet's estimates of range. The errors are reduced to normal at bearings near $80°$, when it is too far away for the laser rangefinders to produce usable clusters. Statistically, LocalMap ground speed estimation errors are indistinguishable from zero at the 5% significance level in 142 of 284 bearings considered, and 211 of 254 are within 0.1 m/sec. of zero at the same significance level.

Figure 5.9 plots average LocalMap target vehicle relative heading estimation errors over the 11 from-the-right perpendicular intersection encounters. Like

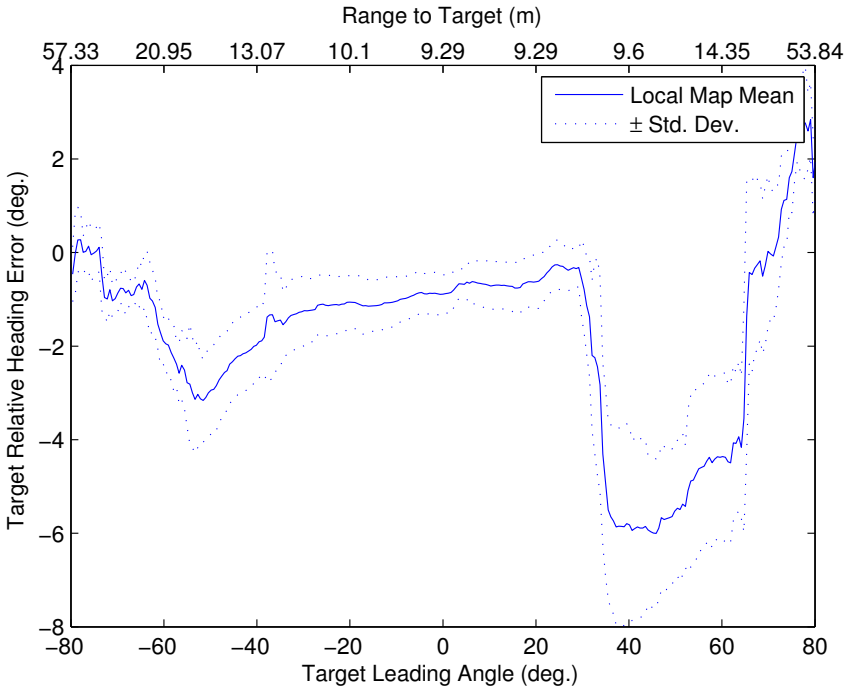

Figure 5.9: LocalMap relative heading tracking errors over 11 perpendicular intersection encounters with a moving target vehicle approaching from the right at 15 mph. The LocalMap's point cloud representation is able to estimate relative heading correctly despite significant changes in target vehicle viewpoint.

the ground speed estimation errors, the target vehicle relative heading estimation errors remain relatively small from first acquisition through the intersection. It is noted, however, that these errors appear slightly biased, perhaps reflecting a minor angular miscalibration in the ego vehicle's right-facing laser. Estimated relative headings also suffer larger errors at target vehicle bearings between 40° and 80° due to Skynet's first left-facing radar. Statistically the overall heading errors are quite small: the LocalMap produces heading errors less than 2° at the 5% significance level in 210 of the 284 bearings considered.

Figure 5.10: In the second experiment, the LocalMap tracks a moving target vehicle as it approaches the moving ego vehicle from the opposite direction. Both vehicles travel at approximately 15 mph for these parallel head-on encounters.

5.3.2 Experiment 2: Parallel Head-On Encounter

The second experiment evaluates the LocalMap's ability to track a moving target vehicle from a moving ego vehicle as they approach each other from opposite directions on parallel tracks. In this experiment, the ego vehicle (Skynet) and the target vehicle approach each other in opposite lanes on a straight road. Both vehicles travel at approximately 15 mph during the experiment, for a combined closing speed of approximately 30 mph. The LocalMap is again run with 50 particles, and each particle is allowed to track as many obstacles as it deems appropriate. Data is collected for 11 such parallel head-on encounters, all conducted on the same road from the same vehicle starting positions. Figure 5.10 shows the experimental setup.

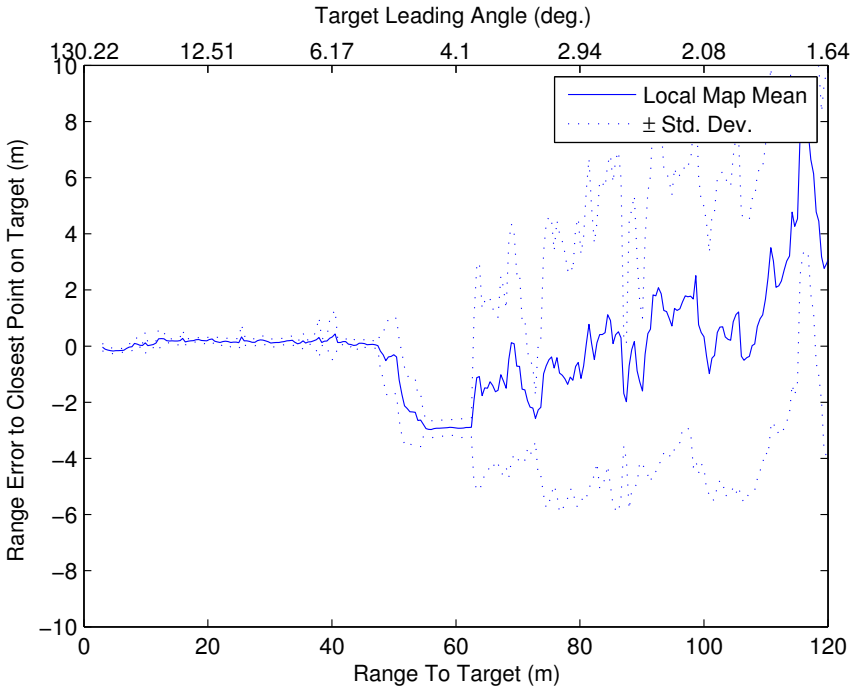

Figure 5.11: LocalMap range tracking errors to the closest point on a moving target vehicle over 11 parallel head-on encounters at 30 mph closing speeds. Error statistics are calculated across encounters according to the true range to the target vehicle. In the 284 true ranges to the target vehicle considered, 153 are statistically equal to zero, and 233 are within 20 cm of zero at the 5% significance level.

Like the first experiment, comparisons in the head-on encounters are made to the obstacles tracked in the LocalMap's MAP estimate of the environment. The same Mahalanobis distance criterion is used to choose the obstacle within this MAP estimate that most closely matches the truth data, and the same distance threshold is applied to discard frames in which the LocalMap was not tracking the target vehicle. The LocalMap experienced 231 of these missed detection frames among the 7264 considered in this experiment. All but one of these missed detection frames occurred when the target vehicle was more than 95 m from the ego vehicle.

LocalMap tracking statistics averaged over the 11 parallel head-on encounters are shown in Figures 5.11, 5.12, and 5.13. In this experiment, tracking data from the trials is aligned by the true range to the closest point on the target vehicle. Each Figure therefore reads right to left; the target vehicle enters detection range at approximately 120 m and approaches the ego vehicle in the oncoming lane until both pass each other. Evaluation of the maneuver ends when the range to the closest point on the target vehicle is at a minimum, approximately 3 m. Figure 5.11 plots average LocalMap errors in range to closest point on the moving target vehicle observed in the 11 parallel head-on encounters. Range errors are largest when the maneuver begins, when only a single radar observes the target vehicle. Range errors drop to near zero at ranges of approximately 50 m, when the target vehicle is first observed by laser rangefinders. The LocalMap also becomes more accurate and consistent with the additional sensor data at these ranges, as shown by its small standard deviations across the trials. Statistically, LocalMap ranging errors are indistinguishable from zero in 153 of the 284 ranges considered, and 233 are within 20 cm of zero at the 5% significance level.

Figure 5.12 plots average LocalMap target vehicle ground speed estimation errors over the 11 parallel head-on encounters. Like the perpendicular intersection encounters, ground speed estimates in the head-on encounters remain accurate at all ranges of the maneuver after initial target acquisition. As in the previous experiment, this accuracy is due to the positioning of the ego vehicle's Delphi radars; one faces forward and measures the speeds of approaching vehicles directly through range rate information in Doppler shifts. Of greater interest is the fact that these accurate speed estimates persist in the closest ranges of the maneuver, as the target vehicle passes to the left of the ego vehicle. At

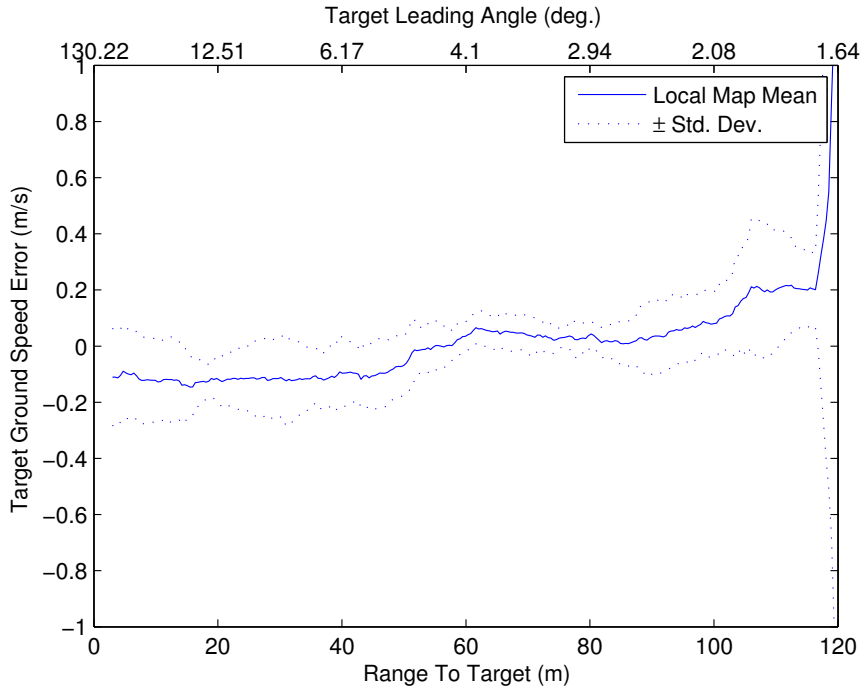

Figure 5.12: LocalMap ground speed tracking errors over 11 parallel head-on encounters at 30 mph closing speeds.

this point in the maneuver the target vehicle is not visible by the forward-facing radar, and its shape as observed by the laser rangefinders varies greatly due to a rapidly changing viewpoint. The LocalMap's point cloud representation suffers no losses from the changing viewpoint, however, and maintains accurate estimates throughout the maneuver. Statistically, LocalMap ground speed estimation errors are indistinguishable from zero at the 5% significance level in 105 of 284 ranges considered in this experiment and within 5 cm / sec. of zero in 246 of the ranges considered.

Figure 5.13 plots average LocalMap target vehicle relative heading estimation errors over the 11 parallel head-on encounters. Like the ground speed estimation errors, the relative heading estimation errors remain small throughout the entire maneuver. At the 5% significance level, these heading errors are in-

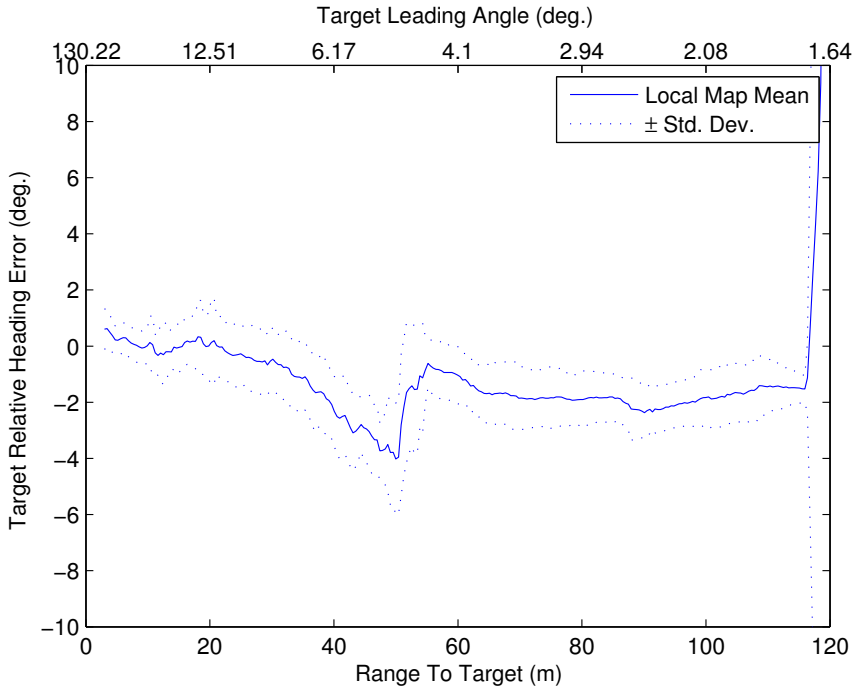

Figure 5.13: LocalMap relative heading tracking errors over 11 parallel head-on encounters at 30 mph closing speeds.

distinguishable from zero in 93 of 284 ranges considered and within 2° of zero in 266 ranges considered. Like the ground speed errors, the heading errors remain accurate even in the closest ranges of the maneuver, when the target vehicle's shape as observed by the laser rangefinder changes most rapidly. The LocalMap's point cloud representation and parameterized rigid body transform resolve these rapid shape changes correctly and without estimation artifacts present in feature extraction approaches.

5.3.3 Experiment 3: Multiple Obstacle Tracking In Heavy Traffic

The third experiment evaluates the LocalMap's consistency in tracking multiple obstacles in a densely-populated, highly dynamic environment. In this experiment, variants of the LocalMap algorithm with different numbers of particles are run on the same segment of logged data, a 19 minute excerpt of a DARPA Urban Challenge qualifying round at George Air Force Base in Victorville, California. The data contains Skynet's sensor measurements of two concentric loops of heavy traffic traveling in opposite directions, recorded as Skynet merged into and out of this traffic multiple times. Figure 5.14 shows the experimental setup.

The Urban Challenge data excerpt is presented to variants of the LocalMap tracking algorithm run with 1, 2, 5, 10, 20, and 50 particles. No truth data is available for the qualifying round, although the merging scenario features repeated instances of the perpendicular intersection encounters and parallel head-on encounters studied in Sections 5.3.1 and 5.3.2. Two new factors are present in this experiment: obstacle occlusion, where one or more obstacles are temporarily blocked from Skynet's view, and large numbers of moving obstacles. This particular data excerpt features large numbers of occlusions, occurring primarily when a vehicle in the inner traffic loop passes alongside a vehicle in the outer traffic loop. In this environment, the LocalMap tracks an average of 32 potentially moving obstacles at each instant in time.

The close interaction between moving traffic vehicles adds data assignment complexity not present in the first two experiments. Though no truth data is available, the effects of the added complexity are reflected in the number of

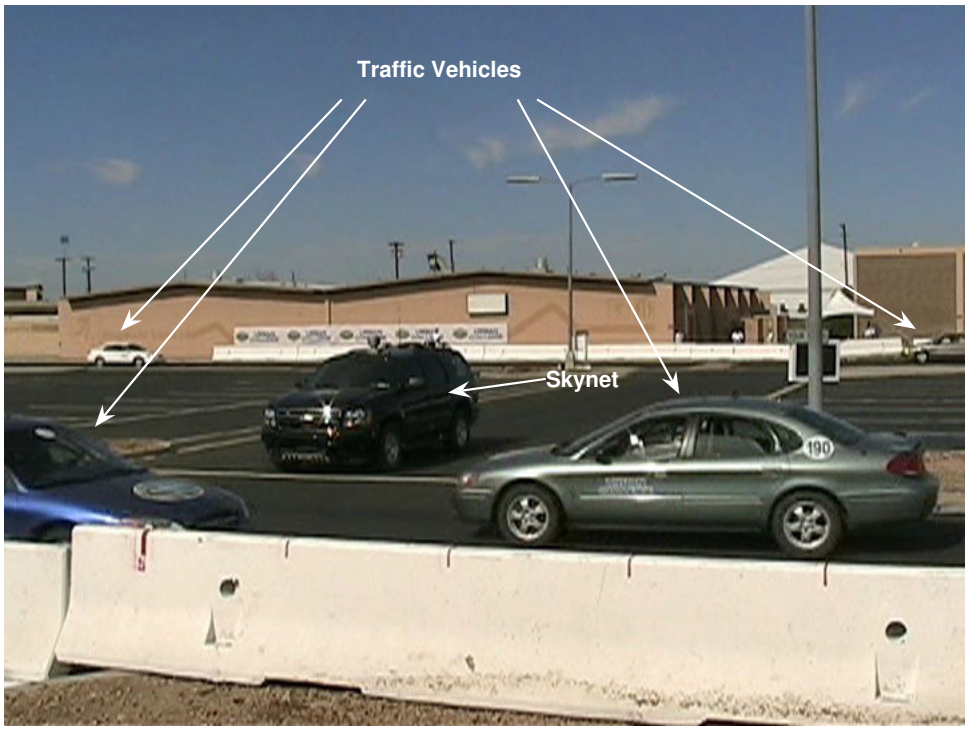

Figure 5.14: In the third experiment, variants of the LocalMap are run on excerpts of data from a DARPA Urban Challenge qualifying round. In this data, Skynet autonomously completes multiple merges into and out of moving traffic across a lane of oncoming vehicles.

obstacles tracked by the LocalMap. Incorrect data assignments, even those resolved quickly in resampling, can temporarily result in a particle with too many or too few obstacles. Accurate Monte Carlo sampling of the data assignments, on the other hand, should yield convergence to a common maximum *a posteriori* estimate as the number of particles increases. This experiment looks for that convergence across LocalMap variants as a measure of data assignment consistency in a complex dynamic environment.

Figure 5.15 plots the sample cumulative distribution function of errors in the number of obstacles tracked in variants of the LocalMap with 1, 2, 5, 10, and 20 particles. Errors are calculated against a LocalMap variant run with 50

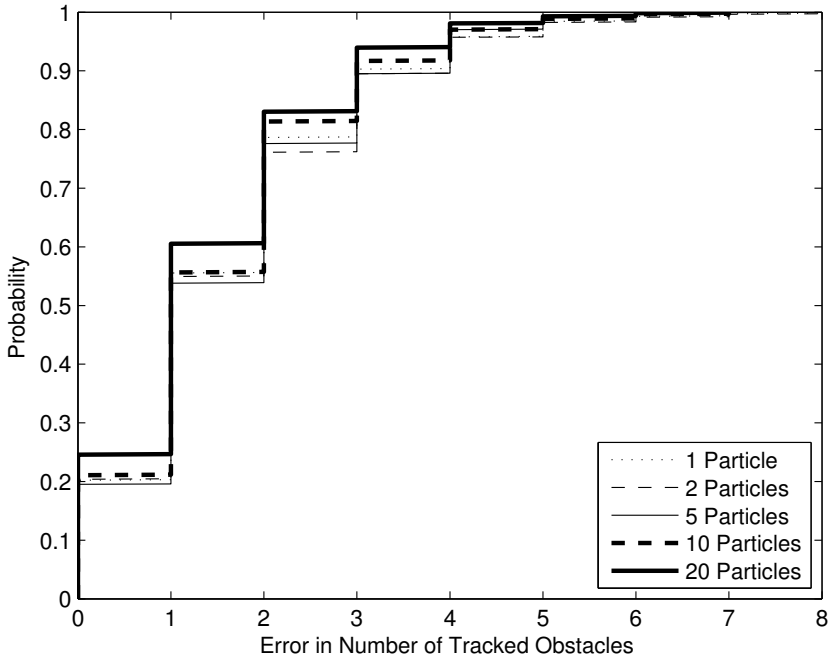

Figure 5.15: Sample cumulative distribution function of errors in the number of obstacles tracked in a heavy traffic scenario in variants of the LocalMap run with 1, 2, 5, 10, and 20 particles. Errors are calculated against a LocalMap variant run with 50 particles. The LocalMap algorithm's convergence to a common number of tracked obstacles as the number of particles increases shows stability in selection of data assignments.

particles, and are only evaluated for the maximum *a posteriori* (MAP) estimate in each LocalMap variant, i.e. the particle with the largest weight. Figure 5.15 shows that the sample cumulative distribution functions for LocalMap variants with 10 and 20 particles are strictly greater than those of variants with fewer particles, indicating convergence to a common number of obstacles as the number of particles increases. More importantly, all the cumulative density functions lie within approximately 5% of each other. From this result it is evident that the LocalMap's point cloud representation and parametric filter representation largely make the data assignments 'obvious,' as even a single particle LocalMap

is capable of achieving results similar to a 50 particle variant throughout most of the experiment.

Although a single particle performs almost as well as 50 in most of the experiment, there are three isolated instances where variants with low numbers of particles result in significant errors. Two of these instances, occurring at $t \approx 12.235$ min. and 18.65 min. into the data excerpt, correspond to times when Skynet just starts to merge across oncoming traffic. The scenario is difficult from a data assignment point of view: all traffic vehicles in the oncoming lane instantly become occluded by the closest oncoming vehicle. A metal light pole behind Skynet further compounds the problem, as it temporarily passes in view of Skynet's rear radars at the same point in the maneuver. False positive detections created by a radar speed detector mounted on the light pole could potentially create false obstacles that amplify the discrepancy. The third instance, at $t \approx 7.57$ min., corresponds to Skynet making a 90° left turning around a tight corner. Here again Skynet's rear radars are likely the source of the error, as a large construction scissors lift temporarily becomes visible during the turn. This lift likely generates multiple Delphi radar tracks, which may be mistakenly assigned to differing numbers of obstacles in LocalMap variants. All three isolated errors track at most eight fewer or eight more objects than the LocalMap variant with 50 particles. In contrast, Figure 5.15 shows that most errors are less than four.

5.4 Summary Of Performance In The DARPA Urban Challenge

The LocalMap tracking algorithm has also been implemented on Skynet in real-time, where it acts as Skynet's sole obstacle detection and tracking system. The LocalMap is implemented in C++ on a single 2.0 GHz Intel Core 2 Duo machine with 2.0 Gb RAM, running Windows Server 2003. Four particles are used in Skynet's LocalMap to ensure real-time processing. A caching scheme for the measurement function is also implemented to avoid recalculation of $h\left(o_{m,k}, P_k\right)$ when obstacle state and point cloud estimates do not change, and assignment likelihoods are only computed for tracked obstacles in thresholded proximity of a measurement. These minor augmentations help offset the computational expense of numerical differentiation performed in the Sigma Point Transforms utilized in the LocalMap.

Skynet relied on the LocalMap for obstacle detection and tracking during the 2007 DARPA Urban Challenge, a 60 mile autonomous urban driving competition held in Victorville, California in November, 2007. The Urban Challenge featured simultaneous interaction of 11 full-sized autonomous robots and approximately 50 human-driven sedans in typical urban traffic scenarios such as merging, parking, intersection queuing, and vehicle following. Many of these encounters tested the practical applicability of the LocalMap's point cloud representation and stable measurements. During one qualifying round of the Urban Challenge, for example, Skynet was required to complete multiple merges into and out of moving traffic across a lane of oncoming vehicles. Skynet made two attempts at this qualifying course, completing 5 successful merges into and out of traffic in the first attempt, and 10 in the second attempt, whose data was utilized in Section 5.3.3. In the finals of the Urban Challenge, Skynet was one of

only six robots to complete the entire 60 mile course. While competing, Skynet's LocalMap tracked a total of 175252 distinct obstacles. On average, the LocalMap tracked 48.5 obstacles in each particle, with a maximum of 209 obstacles per particle in a single iteration [56].

5.5 Conclusion

The LocalMap tracking algorithm has been presented as a computationally feasible, real-time solution to the joint estimation problem of data assignment and dynamic obstacle tracking from a potentially moving robotic platform. The algorithm utilizes a Bayesian factorization to separate the joint estimation problem into an independent data assignment problem and a multiple dynamic obstacle tracking problem conditioned on the data assignments. A particle filter is then used to sample the *a posteriori* distribution of data assignments, and compact and efficient parametric filters are used to estimate the *a posteriori* densities of the obstacles conditioned on the sampled data assignments. The LocalMap algorithm achieves a practical computational burden by using expensive Monte Carlo sampling only over the portion of the state space that most needs it, the data assignment histories. The rest of the states, those of the dynamic obstacles, are estimated with banks of efficient closed-form parametric filters.

The LocalMap algorithm achieves real-time rates through a carefully-selected point cloud obstacle representation and stable measurements. These techniques eliminate estimation artifacts and measurement instability common to sensor data preprocessing techniques such as box and corner detectors, improving the accuracy of the LocalMap's parametric tracking filters even under

substantial changes in obstacle viewpoint. The result makes data assignments 'obvious,' and real-time performance is achieved through commensurate reductions in particle requirements in the data assignment problem.

The LocalMap algorithm has been implemented on Cornell University's 'Skynet,' an autonomous 2007 Chevrolet Tahoe equipped with a position, velocity and attitude estimator, and laser rangefinders, radars, and optical cameras for obstacle tracking. The LocalMap has been validated in three experiments: two experiments in which a single moving target vehicle is tracked under large changes in obstacle viewpoint, and one experiment in which multiple moving target vehicles are tracked in heavy traffic. In these experiments, the LocalMap is shown capable of both determining the number of obstacles and accurately tracking their positions and velocities. Statistics are also presented for Skynet's performance in the DARPA Urban Challenge, where the LocalMap algorithm was implemented in real-time to serve as Skynet's obstacle detection and tracking system. In the DARPA Urban Challenge, the LocalMap ran for 6 hours, allowing Skynet to travel autonomously for 60 miles in moving traffic.

CHAPTER 6

CONCLUSION

This dissertation has presented a complete, real-time, field-proven approach to robotic localization and perception for full-size field robots operating outdoors in static terrain and dynamic urban environments. The approach divided the robotic localization and perception solution into four main components: pose estimation, pose augmentation, static terrain estimation, and dynamic obstacle tracking. Each of these main components was addressed within a formal probabilistic yet efficient, real-time framework to facilitate a rigorous and accurate Bayesian approach to robotic localization and perception without sacrificing computational feasibility. All four of the approach's components have been implemented and validated in real-time on Cornell University's 2005 DARPA Grand Challenge robot and 2007 DARPA Urban Challenge robot. These components helped Cornell's DARPA Grand Challenge robot qualify as one of 23 finalists in the 2005 DARPA Grand Challenge, and they helped Cornell's DARPA Urban Challenge robot complete 60 miles of autonomous driving as one of six robots to complete the 2007 DARPA Urban Challenge.

The first component presented was a tightly-coupled position, velocity, and attitude (pose) estimator built as part of Cornell University's 2007 DARPA Urban Challenge robot, 'Skynet.' Skynet's pose estimator fuses information from GNSS signals with onboard inertial and odometry sensors in an Extended Square Root Information Filter (ESRIF). A sensitivity analysis has been performed on the pose estimator by evaluating its changes in performance as different inputs and design features are removed. The resulting pose estimator variants have been tested on logged data taken at the 2007 DARPA Urban Challenge

National Qualifying Event, with the logged output of the pose estimator used as a baseline solution. This data-driven sensitivity analysis has shown that the pose estimator is most sensitive to differential corrections, though it becomes increasingly less effective as integrity monitoring and other design decisions are reversed as well. Though the pose estimator is statistically most sensitive to differential corrections, it has been argued that precision and robustness are more important than absolute accuracy in autonomous urban navigation. As a result, many tested pose estimator variants have yielded intolerably large discontinuities in their pose solutions. In fact, only variants including filter integrity monitoring were found suitable in this respect, aside from one conservative variant using only pseudoranges.

The second component presented was the PosteriorPose algorithm, a particle filtering approach for augmenting an existing absolute pose solution with relative landmark measurements referenced to a known map. The PosteriorPose algorithm incorporates techniques to correctly fuse absolute navigation signals with measurements of nearby lanes, lane lines, and stop lines in the Bayesian probabilistic framework. These techniques have been augmented with rigorous measurement hypothesis testing to cope with real-world sensor errors, especially non-Gaussian errors in sensors based on computer vision techniques. The PosteriorPose algorithm has been validated experimentally on two driving courses, one with dense road information and one with sparse road information. In each case, the algorithm was allowed 5 minutes of initialization data followed by 30 minutes of test data. The performance of the algorithm was compared to Skynet's tightly-coupled pose estimator under the same GPS signal availability. In full GPS signal availability, the PosteriorPose algorithm proved statistically superior to the standard pose estimator. In two 30 minute extended GPS

blackouts, the PosteriorPose algorithm impressively retained a converged and accurate navigation solution. The standard pose estimator solution, which integrated inertial and odometry sensors in the GPS blackout, suffered from large dead reckoning errors.

The third component presented a terrain mapping and estimation algorithm, which uses Gaussian sum height densities to model terrain variations in a planar gridded elevation model. The algorithm accounts for multiple sources of sensor error in fusing terrain measurements into the map to produce robust and accurate terrain estimates. Most importantly, the algorithm potentially assigns each measurement to multiple locations in the elevation model, to properly account for uncertainty in the in-plane location of each measurement as well as elevation. The algorithm has been compared in simulation to minimum / maximum elevation metrics, and it has been shown to produce more consistent statistical results on a variety of common terrain features. The algorithm has also been validated against experimental data gathered in an environment containing terrain features with known heights.

The final component presented was the LocalMap tracking algorithm, a computationally feasible, real-time solution to the joint estimation problem of data assignment and dynamic obstacle tracking from a potentially moving robotic platform. The algorithm utilizes a Bayesian factorization to separate the joint estimation problem into an independent data assignment problem and a multiple dynamic obstacle tracking problem conditioned on the data assignments. A particle filter is then used to sample the *a posteriori* distribution of data assignments, and compact and efficient parametric filters, specifically designed for tracking full-size moving vehicles, are used to estimate the *a posteriori*

densities of the obstacles conditioned on the sampled data assignments. The LocalMap has been validated in three real-world experiments: two experiments in which a single moving target vehicle is tracked under large changes in obstacle viewpoint, and one experiment in which multiple moving target vehicles are tracked in heavy traffic. In these experiments, the LocalMap is shown capable of both determining the number of obstacles and accurately tracking their positions and velocities.

6.1 Summary of Contributions

Listed below are significant contributions the author believes this dissertation makes to probabilistic and field robotics research:

Chapter 2, 'Tightly-Coupled GPS / INS System Design for Autonomous Urban Navigation'

- A rigorous statistical sensitivity analysis of a tightly-coupled pose estimator implemented for autonomous navigation in a ground vehicle
- An explanation of pose estimator performance and design features in relation to autonomous navigation

Chapter 3, 'Particle Filtering for Map-Aided Localization in Sparse GPS Environments'

- A map-aided pose estimation algorithm, implemented as a particle filter, that uses computer vision algorithms' measurements of painted lanes and stop lines to augment traditional GNSS navigation signals

- A formulation of the map-aiding pose estimation problem that relaxes the common map-aiding assumption / constraint that the quarry remains on the mapped road network
- Techniques for Bayesian incorporation of filtered GPS / INS measurements and computer vision lane and stop line measurements into a particle-filtered localization scheme using map-aiding to identify sensor detection modes and hypothesis tests to reject outliers
- An experimental demonstration of long-term stability of map-aided pose estimates in two 30-minute GNSS blackouts
- A real-time implementation of a map-aided pose estimation algorithm, experimentally validated in 60 miles of autonomous driving in the 2007 DARPA Urban Challenge

Chapter 4, 'A Mixture-Model Based Algorithm for Real-Time Terrain Estimation'

- A dense terrain estimation algorithm that fuses ground measurements into a grid-based elevation map while accounting for multiple sources of sensor uncertainty
- Techniques to generate sufficient statistics for terrain estimation without maintaining histories of sensor measurements
- Identification and proper Bayesian handling of the data assignment problem as it relates to terrain estimation
- A real-time implementation of a grid-based terrain estimation algorithm, experimentally validated in the 2005 DARPA Grand Challenge

Chapter 5, 'Stable and Efficient Tracking of Multiple Dynamic Obstacles Under Large Viewpoint Changes'

- A Rao-Blackwellized Particle Filter multitarget tracking algorithm that robustly estimates obstacle maneuvers in dynamic urban environments despite potentially close obstacle ranges and rapid changes in viewpoints
- A Bayesian rigid body target dynamics model and point cloud geometry representation that eliminates estimation artifacts caused by heuristic feature extraction and matching techniques
- Sets of stable, linearizable measurements for robust obstacle tracking in dynamic urban environments using laser rangefinders, radars, and optical cameras
- An experimental demonstration of accurate, robust, and repeatable obstacle tracking in two common urban driving encounters
- A real-time implementation of a multitarget tracking algorithm fusing information from multiple sensors and multiple sensor types, experimentally validated in 60 miles of autonomous driving in moving traffic in the 2007 DARPA Urban Challenge

BIBLIOGRAPHY

[1] United States 106th Congress. National defense authorization, fiscal year 2001, October 2000. Public Law 106-398, H. R. 4205, Section 220.

[2] F. Ackermann. Airborne laser scanning - present status and future expectations. *ISPRS Journal of Photogrammetry and Remote Sensing*, 54:64–67, 1999.

[3] S. Alban, D. Akos, S. Rock, and D. Gebre-Egziabher. Performance analysis and architectures for INS-aided GPS tracking loops. In *Proceedings of ION National Technical Meeting, ION-NTM*, pages 611–622, 2003.

[4] T. Anderson. *An Introduction to Multivariate Statistical Analysis*. John Wiley & Sons, Inc., New York, 2nd edition, 1984.

[5] K. Arakawa and E. Krotkov. Fractal surface reconstruction with uncertainty estimation: Modeling natural terrain. Technical Report CMU-CS-92-194, School of Computer Science, Carnegie Mellon University, Pittsburgh, 1992.

[6] F. Artes and L. Nastro. Applanix POS LV 200: Generating continuous position accuracy in the absence of GPS reception, August 2005.

[7] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, February 2002.

[8] P. Axelsson. Processing of laser scanner data - algorithms and applications. *ISPRS Journal of Photogrammetry and Remote Sensing*, 54:138–147, 1999.

[9] Y. Bar-Shalom and T. Fortmann. *Tracking and Data Association*. Academic Press, Inc., Orlando, 1988.

[10] Y. Bar-Shalom, T. Kirubarajan, and X. Lin. Probabilistic data association techniques for target tracking with applications to sonar, radar, and eo sensors. *IEEE A&E Systems Magazine*, 20(8):37–56, August 2005.

[11] Y. Bar-Shalom, X. Rong Li, and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation: Theory, Algorithms and Software*. John Wiley & Sons, Inc., New York, 2001.

[12] G. Bierman. *Factorization Methods for Discrete Sequential Estimation*. Academic Press, New York, 1977.

[13] J. Bosen. A formula for approximation of the saturation vapor pressure of water. *Monthly Weather Review*, 88(8):275–276, August 1960.

[14] A. Bridges. $1 million pentagon-sponsored robot race ends as all entries break down. *Associated Press*, March 2004. Online: http://www.space.com/businesstechnology/technology/darpa_race_040313.html.

[15] M. Campbell, E. Garcia, D. Huttenlocher, I. Miller, P. Moran, A. Nathan, B. Schimpf, N. Zych, J. Catlin, F. Chelarescu, H. Fujishima, F.-R. Kline, S. Lupashin, M. Reitmann, A. Shapiro, and J. Wong. Team cornell: Technical review of the DARPA urban challenge vehicle, April 2007.

[16] Y. Cheng and T. Singh. Efficient particle filtering for road-constrained target tracking. In *Proceedings of the Eighth International Conference on Information Fusion*, volume 1, pages 161–168, July 2005.

[17] Y. Cui and S. S. Ge. Autonomous vehicle positioning with gps in urban canyon environments. *IEEE Transactions on Robotics and Automation*, 19(1):15–25, February 2003.

[18] J. Davis, T. Herring, I. Shapiro, A. Rogers, and G. Elgered. Geodesy by radio interferometry: Effects of atmospheric modeling errors on estimates of baseline length. *Radio Science*, 20(6):1593–1607, November-December 1985.

[19] Defense Advanced Research Projects Agency. *Urban Challenge Rules*, July 2007.

[20] A. Doucet, V. Ba-Ngu, C. Andrieu, and M. Davy. Particle filtering for multitarget tracking and sensor management. In *Proceedings of the Fifth International Conference on Information Fusion*, volume 1, pages 474–481, 2002.

[21] A. Doucet, N. de Freitas, K. Murphy, and S. Russell. Rao-blackwellised particle filtering for dynamic bayesian networks. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 176–183, Stanford, 2000.

[22] S. El-Hakim, E. Whiting, L. Gonzo, and S. Girardi. 3-d reconstruction of complex architectures from multiple data. In *Proceedings of the ISPRS Work-*

ing Group V/4 Workshop, Venice-Mestre, Italy, August 2005. ISPRS International Workshop on 3D Virtual Reconstruction and Visualization of Complex Architectures, International Society for Photogrammetry and Remote Sensing.

[23] M. El Najjar and P. Bonnifait. A road-matching method for precise vehicle localization using belief theory and kalman filtering. *Autonomous Robots*, 19(2):173–191, September 2005.

[24] F. Fayad and V. Cherfaoui. Tracking objects using a laser scanner in driving situation based on modeling target shape. In *Proceedings of the 2007 IEEE Intelligent Vehicles Symposium*, pages 44–49, June 2007.

[25] P. Felzenszwalb and D. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[26] L. Fletcher, S. Teller, E. Olson, D. Moore, Y. Kuwata, J. How, J. Leonard, I. Miller, M. Campbell, D. Huttenlocher, A. Nathan, and F.-R. Kline. The mit - cornell collision and why it happened. *Journal of Field Robotics*, 2008. To appear.

[27] K. Fürstenberg and K. Dietmayer. Object tracking and classification for multiple active safety and comfort applications using a multilayer laserscanner. In *Proceedings of the 2004 IEEE Intelligent Vehicles Symposium*, pages 802–807, June 2004.

[28] K. Fürstenberg, K. Dietmayer, and S. Eisenlauer. Multilayer laserscanner for robust object tracking and classification in urban traffic scenes. In *Proceedings of the Ninth World Congress on Intelligent Transportation Systems*, pages 7–16, Chicago, Illinois, 2002.

[29] K. Fürstenberg, K. Dietmayer, and V. Willhoeft. Pedestrian recognition in urban traffic using a vehicle based multilayer laserscanner. In *Proceedings of the 2002 IEEE Intelligent Vehicles Symposium*, volume 1, pages 31–35, June 2002.

[30] I. Gat, M. Benady, and A. Shashua. A monocular vision advance warning system for the automotive aftermarket. In *SAE 2005 Transactions Journal of Passenger Cars: Electronic and Electrical Systems*, pages 403–410, 2005. 2005-01-1470.

[31] M. George and S. Sukkarieh. Tightly coupled INS / GPS with bias estima-

tion for UAV applications. In *Proceedings of the 2005 Australasian Conferences on Robotics and Automation*, December 2005.

[32] GPS Joint Program Office. Navstar gps space segment / navigation user interfaces, March 2006. IS-GPS-200D.

[33] G. Grisetti, C. Stachniss, and W. Burgard. Improving grid-based slam with Rao-Blackwellized particle filters by adaptive proposals and selective resampling. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2432–2437, 2005.

[34] D. Hähnel, W. Burgard, and S. Thrun. Learning compact 3d models of indoor and outdoor environments with a mobile robot. *Elsevier Science Special Issue Eurobot '01*, pages 1–16, 2004.

[35] J. Hooper. From DARPA grand challenge 2004: DARPA's debacle in the desert. *Popular Science*, 2004. Online: http://www.popsci.com/scitech/article/2004-06/darpa-grand-challenge-2004darpas-debacle-desert.

[36] E. Huising and L. Pereira. Errors and accuracy estimates of laser data acquired by various laser scanning systems for topographic applications. *ISPRS Journal of Photogrammetry and Remote Sensing*, 53:245–261, 1998.

[37] S. Julier and J. Uhlmann. A general method for approximating nonlinear transformations of probability distributions. Technical report, Department of Engineering Science, University of Oxford, Oxford, November 1996.

[38] S. Julier and J. Uhlmann. A new extension of the Kalman filter to nonlinear systems. In *Proceedings of the SPIE Volume 3068: Signal Processing, Sensor Fusion, and Target Recognition VI*, pages 182–193, 1997.

[39] E. Kaplan, editor. *Understanding GPS: Principles and Applications*. Artech House, Boston, 1996.

[40] T. Kirubarajan and Y. Bar-Shalom. Probabilistic data association techniques for target tracking in clutter. *Proceedings of the IEEE*, 92(3):536–557, March 2004.

[41] S. Lacroix, D. Mallet, G. Bonnafous, S. Bauzil, S. Fleury, M. Herrb, and R. Chatila. Autonomous rover navigation on unknown terrains: Functions

and integration. *Interjational Journal of Robotics Research*, 21(10-11):917–942, 2002.

[42] J. Leal. *Stochastic Environment Representation*. PhD Thesis, University of Sydney, January 2003. Australian Centre for Field Robotics.

[43] K. W. Lee, W. S. Wijesoma, and J. Ibañez-Guzmán. Map aided slam in neighbourhood environments. In *Proceedings of the 2004 IEEE Intelligent Vehicles Symposium*, pages 836–841, June 2004.

[44] F. Lemoine, S. Kenyon, J. Factor, R. Trimmer, N. Pavlis, D. Chinn, C. Cox, S. Klosko, S. Luthcke, M. Torrence, Y. Wang, R. Williamson, E. Pavlis, R. Rapp, and T. Olson. The development of the joint NASA GSFC and NIMA geopotential model EGM96, July 1998. NASA/TP-1998-206861.

[45] P. Lohmann, A. Koch, and M. Schaeffer. Approaches to the filtering of laser scanner data. *Interanational Archives of Photogrammetry and Remote Sensing*, 33, 2000.

[46] J. Lundberg and B. Schutz. Recursion formulas of legendre functions for use with nonsingular geopotential models. *Journal of Guidance, Control, and Dynamics*, 11(21):31–38, January-February 1988.

[47] R. MacLachlan and C. Mertz. Tracking of moving objects from a moving vehicle using a scanning laser rangefinder. In *Proceedings of the IEEE Intelligent Transportation Systems Conference 2006*, pages 301–306, September 2006.

[48] M. Maehlisch, T. Kauderer, W. Ritter, and K. Dietmayer. Feature-level video and multibeam lidar sensor fusion for full-speed acc state estimation. In *Proceedings of the 4th International Workshop on Intelligent Transportation*, 2007.

[49] M. Maehlisch, W. Ritter, and K Dietmayer. De-cluttering with integrated probabilistic data association for multisensor multitarget acc vehicle tracking. In *Proceedings of the 2007 IEEE Intelligent Vehicles Symposium*, pages 178–183, Istanbul, Turkey, 2007.

[50] M. Maehlisch, R. Schweiger, W. Ritter, and K. Dietmayer. Multisensor vehicle tracking with the probability hypothesis density filter. In *Proceedings of the 9th International Conference on Information Fusion*, pages 1–8, Florence, Italy, 2006.

[51] J. Marsden and A. Weinstein. *Calculus 1*. Springer-Verlag, New York, 2nd edition, 1985.

[52] M. Martin and H. Moravec. Robot evidence grids. Technical Report CMU-RI-TR-96-06, The Robotics Institute, Carnegie Mellon University, Pittsburgh, March 1996.

[53] A. Mendes, L. Conde Bento, and U. Nunes. Multi-target detection and tracking with a laserscanner. In *Proceedings of the 2004 IEEE Intelligent Vehicles Symposium*, pages 796–801, June 2004.

[54] I. Miller and M. Campbell. Rao-blackwellized particle filtering for mapping dynamic environments. In *Proceedings of the 2007 International Conference on Robotics and Automation*, pages 3862–3869, April 2007.

[55] I. Miller and M. Campbell. Particle filtering for map-aided localization in sparse gps environments. In *Proceedings of the 2008 International Conference on Robotics and Automation*, pages 1834–1841, May 2008.

[56] I. Miller, M. Campbell, D. Huttenlocher, A. Nathan, F.-R. Kline, P. Moran, N. Zych, B. Schimpf, S. Lupashin, E. Garcia, M. Catlin, J. Kurdziel, and H. Fujishima. Team cornell's skynet: Robust perception and planning in an urban environment. *Journal of Field Robotics*, 25(8):493–527, August 2008.

[57] I. Miller, E. Garcia, and M. Campbell. To drive is human. *Computer*, 39(12):52–56, December 2006.

[58] I. Miller, S. Lupashin, N. Zych, P. Moran, B. Schimpf, A. Nathan, and E. Garcia. Cornell University's 2005 DARPA Grand Challenge Entry. *Journal of Field Robotics*, 23(8):625–652, 2006.

[59] I. Miller, B. Schimpf, J. Leyssens, and M. Campbell. Tightly-coupled gps / ins system design for autonomous urban navigation. In *Proceedings of the 2008 IEEE / ION Position, Localization, and Navigation Symposium*, May 2008.

[60] S. Mohiuddin and M. Psiaki. Satellite relative navigation using carrier-phase differential gps with integer ambiguities. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, San Francisco, California, August 2005. Paper AIAA-2005-6054.

[61] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In

Proceedings of the AAAI National Conference on Artificial Intelligence, Edmonton, Canada, 2002. AAAI.

[62] F. Moon. *Applied Dynamics With Application to Multibody and Mechatronic Systems*. John Wiley and Sons, New York, 1998.

[63] K. Morin. *Calibration of Airborne Laser Scanners*. PhD thesis, University of Calgary, November 2002.

[64] R. Murray, Z. Li, and S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Boca Raton, 1994.

[65] A. Niell. Global mapping functions for the atmosphere delay at radio wavelengths. *Journal of Geophysical Research*, 101(B2):3227–3246, 1996.

[66] NIMA. Digital terrain elevation data (DTED), 2000. http://www.fas.org/irp/program/core/dted.htm.

[67] Northrop Grumman Navigation Systems Division. *LN-200 Fiber Optic Inertial Measurement Unit*, 2000.

[68] E. Ohlmeyer, T. Pepitone, B. Miller, D. Malyevac, J. Bibel, and A. Evans. Gps-aided navigation system requirements for smart munitions and guided missiles. In *AIAA Guidance, Navigation, and Control Conference Collection of Technical Papers*, pages 954–968. American Institute of Aeronautics and Astronautics, Reston, August 1997.

[69] K. Olin and D. Tseng. Autonomous cross-country navigation: An integrated perception and planning system. *IEEE Expert: Intelligent Systems and Their Applications*, 6(4):16–30, August 1991.

[70] D. Pagac, E. Nebot, and H. Durrant-Whyte. An evidential approach to map-building for autonomous vehicles. *IEEE Transactions on Robotics and Automation*, 14(4), August 1998.

[71] M. Psiaki and S. Mohiudden. Modeling, analysis and simulation of GPS carrier phase for spacecraft relative navigation. *Journal of Guidance, Control, and Dynamics*, 30(6):1628–1639, November-December 2007.

[72] M. Psiaki, J. Theiler, J. Bloch, S. Ryan, R. Dill, and R. Warner. ALEXIS spacecraft attitude reconstruction with thermal / flexible motions due to

launch damage. *Journal of Guidance, Control, and Dynamics*, 20(5):1033–1041, September-October 1997.

[73] S. Russell. DARPA grand challenge winner: Stanley the robot! *Popular Science*, January 2006. Online: http://www.popularmechanics.com/science/robotics/2169012.html.

[74] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, Inc., Upper Saddle River, 2nd edition, 2003.

[75] J. Saastamoinen. Atmospheric correction for the troposphere and stratosphere in radio ranging of satellites. In S. Henriksen, A. Mancini, and B. Chovitz, editors, *Geophysical Monograph Series*, volume 15, pages 247–251, 1972.

[76] S. Särkkä, A. Vehtari, and J. Lampinen. Rao-blackwellized monte carlo data association for multiple target tracking. In P. Svensson and J. Schubert, editors, *Proceedings of the Seventh International Conference on Information Fusion*, volume 1, pages 583–590, Mountain View, CA, 2004.

[77] D. Satale and M. Kulkarni. Lidar in mapping. In *Proceedings of the 2003 Map India Poster Session*, New Delhi, India, 2003. Map India 2003 Poster Session, GISdevelopment.

[78] P. Savage. Strapdown inertial navigation integration algorithm design part 1: Attitude algorithms. *Journal of Guidance, Control, and Dynamics*, 21(1):19–28, January-February 1998.

[79] P. Savage. Strapdown inertial navigation integration algorithm design part 2: Velocity and position algorithms. *Journal of Guidance, Control, and Dynamics*, 21(2):208–221, March-April 1998.

[80] P. Savage. Errata: Strapdown inertial navigation integration algorithm design part 2: Velocity and position algorithms. *Journal of Guidance, Control, and Dynamics*, 27(2):318, March-April 2004.

[81] B. Scherzinger. *Precise Robust Positioning with Inertial / GPS RTK*. Applanix Corporation, 2000.

[82] R. Schulz and K. Fürstenberg. Laserscanner for multiple applications in passenger cars and trucks. In *Advanced Microsystems for Automotive Applications 2006*, pages 129–138, 2006.

[83] Septentrio Satellite Navigation. *PolaRx2 User Manual Version 2.6*, June 2005.

[84] SICK AG Division Auto Ident. *LMS 200, LMS 211, LMS 220, LMS 221, LMS 291 Laser Measurement Systems Technical Description*, June 2003.

[85] J. Stoker. Voxels as a representation of multiple-return lidar data. In *ASPRS Annual Conference Proceedings*, Bethesda, MD, May 2004. American Society for Photogrammetry and Remote Sensing.

[86] D. Streller and K. Dietmayer. Object tracking and classification using a multiple hypothesis approach. In *2004 Intelligent vehicles Symposium*, pages 808–812, June 2004.

[87] D. Streller, K. Fürstenberg, and K. Dietmayer. Vehicle and object models for robust tracking in traffic scenes using laser range images. In *Proceedings of the IEEE 5th International Conference on Intelligent Transportation Systems*, pages 118–123, September 2002.

[88] S. Sukkarieh, E. Nebot, and H. Durrant-Whyte. A high integrity imu/gps navigation loop for autonomous land vehicle applications. *IEEE Transactions on Robotics and Automation*, 15(3):572–578, June 1999.

[89] Team Cornell. Technical review of team cornell's spider: Darpa grand challenge 2005, December 2005. http://www.darpa.mil/grandchallenge05/TechPapers/TeamCornell.pdf.

[90] S. Thrun. Robotic mapping: A survey. Technical Report CMU-CS-02-111, School of Computer Science, Carnegie Mellon University, Pittsburgh, February 2002.

[91] S. Thrun, W. Burgard, and D. Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, San Francisco, CA, 2000. IEEE.

[92] S. Thrun, Y. Liu, D. Koller, A. Ng, Z. Ghahramani, and H. Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *The International Journal of Robotics Research*, 23(7-8):693–716, 2004.

[93] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley,

M. Palatucci, V. Pratt, and P. Stang. Stanley: The robot that won the DARPA grand challenge. *Journal of Field Robotics*, 23(9):661–692, 2006.

[94] Trimble. *AgGPS 252 Receiver User Guide Version 1.00, Revision A*, February 2004.

[95] U.S. Department of Transportation Federal Aviation Administration. U.s. department of transportation federal aviation administration specification for the wide area augmentation system (waas), August 2001. FAA-E-2892b.

[96] C. Wang. *Simultaneous Localization, Mapping and Moving Object Tracking*. PhD Thesis, Robotics Institute, Carnegie Mellon University, April 2004. CMU-RI-TR-04-23.

[97] J. Weingarten and R. Siegwart. Ekf-based 3d slam for structured environment reconstruction. In *Proceedings of the IEEE / RSJ International Workshop on Intelligent Robots and Systems*, pages 3834–3839, Los Alamitos, 2005. Institute of Electrical and Electronic Engineers.

[98] S. Wender and K. Dietmayer. 3d vehicle detection using a laser scanner and a video camera. In *Proceedings of the 6th European Congress on ITS*, June 2007.

[99] W. S. Wijesoma, K. W. Lee, and J. Ibañez-Guzmán. Motion constrained simultaneous localization and mapping in neighbourhood environments. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 1085–1090, April 2005.