System Architectures and Environment Modeling for High-Speed Autonomous Navigation

Thesis by

Lars B. Cremean

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

2006

(Defended May 12, 2006)

Acknowledgements

Many thanks go out to the people that have provided guidance and support along the way to me finishing this thesis. First and foremost, I would like to thank my advisor, Richard Murray, for allowing me the latitude to explore a variety of directions in my research, for his support in my intensive participation in Caltech's entries in the DARPA Grand Challenge, and for the encouragement to run with this effort as my thesis project. His enthusiasm, energy, and seeming ability to pack twenty-five hours into each of his days is an inspiration. I also would like to thank the other members of my thesis committee: Joel Burdick, Erik Antonsson, and Pietro Perona, who were each a valuable source of expertise and guidance to me in my research and project leadership roles.

Bob and Alice, the experimental platforms that enabled much of the research in this thesis, would not have achieved so much success in the DARPA Grand Challenge events (let alone exist in the first place) were it not for the effort of a large number of undergraduates who made them a reality. It has been a pleasure working with each and every one of them. Special recognition goes to Jeff Lamb and Tully Foote, who led the vehicle and actuation activities for both vehicles, Laura Lindzey for implementing the trajectory follower on Alice, Alex Stewart for his relentless effort implementing supervisory control (and for late night coffee runs) and Jeremy Gillula for leading Alice's sensor fusion and mapping efforts (and supporting my data collection runs to the desert). Thanks also to the rest of the planning teams for their extraordinary efforts, especially Dima Kogan, whose enthusiasm and dedication was an inspiration. The planner truly is awesome.

Thanks also to my officemate, Kristo Kriechbaum, and his wife Maria, for their friendship while sharing the full Caltech experience with me, including surfing, SOPS softball, Thomas Lunch Group, Dodger games and countless Ernies.

Special and loving thanks go to my parents, my brothers Michael and Nels, and Robert, who have provided consistent love and encouragement for all of my endeavors. Finally, loving thanks to Marissa, who has provided immense support for me over the last three years, even and

especially through bouts of sleeplessness, delirium and one-of-a-kind trips to Fontana, Barstow, and Primm.

Abstract

Successful high-speed autonomous navigation requires integration of tools from robotics, control theory, computer vision, and systems engineering. This thesis presents work that develops and combines these tools in the context of navigating desert terrain.

A comparative analysis of reactive, behavior-based, hybrid, and deliberative control architectures provides important guidelines for design of robotic systems. These guidelines depend on the particular task and environment of the vehicle. Two important factors are identified which guide an effective choice between these architectures: dynamic feasibility for the vehicle, and predictability of the environment. This is demonstrated by parallels to control theory, illustrative examples, simulations, and analysis of Bob and Alice—Caltech's full-scale autonomous ground vehicle entries in the 2004 and 2005 Grand Challenge races, respectively.

Further, new model-based methods are developed for constructing and maintaining estimates of terrain elevation and road geometry. These are demonstrated in simulation and in fully autonomous operation of Alice, including accurate detection and tracking of the centerline of desert roads at speeds up to 5 m/s. Finally, Alice's navigation architecture is presented in full along with experimental results that demonstrate its capabilities.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis is concerned with developing and analyzing the mathematical, algorithmic, and systemic tools needed to provide increased autonomy for robotic systems. Broadly, such tools are needed for both semi- and fully autonomous systems and have applications in a variety of areas including underwater exploration; disaster rescue and recovery; human and cargo transport; military transport and reconnaisance through land, sea, and air; assistive robotics; and planetary exploration.

The tools developed in this thesis are applied in particular to high-speed autonomous ground navigation, a field that has built upon decades of research in a number of fields including control theory, robotics, artificial intelligence, computer vision, dynamical systems, systems engineering, and mechanics.

This thesis focuses on two important considerations in the design of systems for high-speed autonomous navigation: dynamic feasibility and predictability. For the purposes of this work, high-speed is defined as in the range 5–15 m/s, depending on terrain. Testing was performed in low-structure desert environments, primarily over unimproved desert roads.

1.1 Motivation

Autonomy is a keystone for technological advancement in many commercial, military and human exploration endeavors, and the ability for a vehicle or robot to successfully navigate through its environment is essential feature for many of such endeavors. Specific and immediate goals underscore the need for such advancement:

- In 2001, Congress set as a goal that "one-third of the operational ground combat vehicles of the Armed Forces" will be unmanned by the year 2015.[1] In 2003, it authorized defense

[1] National Defense Authorization Act for Fiscal Year 2001 (S. 2549, Sec. 217)

agencies to provide "cash prizes in recognition of outstanding achievements ... designed to promote science, mathematics, engineering, or technology education."[2]

- Since January 2004 to the time of this writing, NASA's *Spirit* and *Opportunity* rovers have been exploring and collecting science data on the Martian surface, primarily through remote control from the (Earth's) ground and through partial autonomy. In 2009, a much larger and more capable rover called the Mars Science Laboratory is planned for exploration of Mars. The degree of autonomy on this rover will have a dramatic impact on the ability of this rover to efficiently collect scientific data [29, 30].

Successfully meeting ambitious goals such as those of unmanned military transport and interplanetary exploration requires technological advancements and improved fundamental capabilities in research areas of robotics, control theory, computer vision, and systems engineering.

To spur innovation and development toward the goal stated above for ground combat vehicles, DARPA sponsored two races for autonomous ground vehicles through the Mojave and Nevada deserts, called the DARPA Grand Challenge. Caltech entered vehicles in each of these races, which serve as the primary testbeds and examples for the work presented here, which is generally applicable to autonomous navigation on Earth as well as Mars.

1.2 Themes

There are, of course, many considerations for the design of any autonomous system. The importance of such considerations depends on the specific properties of the system and requirements of the task at hand. Two central themes emerge from a thoughtful analysis of these considerations: *dynamic feasibility* properties of the system, and *predictability* properties of the situated environment in which the system operates.

1.2.1 Dynamic Feasibility

Dynamic feasibility refers to a family of issues in control including stability, constraints on system state, input constraints, and constraints imposed by the dynamics of the system itself. These issues are embedded in the concept of *flight envelopes* in aircraft flight and of *reachability spaces* in control and dynamical systems, which is the set of states that are achievable by a dynamical system in some specified time. The relative importance of dynamic feasibility is related to the operating condition of the system and how close the aircraft is to operating at the edge of the

[2] National Defense Authorization Act for Fiscal Year 2003 (H.R. 4546, Sec. 2374b)

flight envelope. Generally, dynamic feasibility increases in relative importance the closer the system is to the edge of the operating envelope.

For some systems and autonomous tasks, dynamic feasiblity is not a major concern. For example, in navigating a two-wheeled robot at slow speeds in an uncluttered environment, the operating conditions of the system and task are such that the robot can safely stop and take time to determine the best course of action. For unmanned aircraft, which are open-loop unstable, stopping is not an option, but the vehicle may be able to cruise while determining the best course. For both examples, if the low-level control (motor control for the robot or servo control for the unmanned plane) can be trusted to provide stability and a sufficient "flight envelope" exists for the specified task, dynamic feasibility is not as significant a concern in deciding the proper course of action.

The nature of the environment has a big impact on the importance of dynamic feasibility in accomplishing a robot task. Consider giving our hypothetical two-wheeled robot and (small) unmanned plane the same task of navigating an indoor corridor with turns to take a photograph of a target at the end while avoiding hitting the walls. Clearly, it is a greater challenge for the plane to accomplish this task; traveling too slowly will cause it to stall and crash, and traveling too quickly might make it difficult or impossible to make the turns. Spatial constraints of the environment and temporal constraints of the vehicle dynamics combine to make dynamic feasibility an essential consideration for certain tasks.

1.2.2 Predictability

Predictability refers to the degree to which a system can completely and accurately model its surroundings at the current time and into the future, and it is the primary important consideration for deciding the appropriate methodology for a wide variety of robotic tasks.

Consider the example of the previous section. Navigating the same hallway while there are people wandering it provides an even greater challenge to both of our vehicles. Two approaches can help our vehicles in their attempt to satisfy this more difficult task. One is to equip them with complete spatio-temporal motion models of the people in the hallways, from the beginning to the end of the experiment, and instruct the vehicle to avoid people based on this model. Another is to equip them with fast and accurate sensing to rapidly and accurately detect the locations of all of the people in the hallway at any given time.

Clearly, the complete motion model is preferable if it is guaranteed to perfectly reflect the state of the people in the hallway at all times in the interval of interest. With this guarantee,

fast, accurate, and global sensing is not necessary; it only provides a sliver in time of what is already available in the model! Further, while probably sufficient for the robot, the sliver might not be enough information for the plane to satisfy the spatio-temporal constraint of avoiding the people.

In reality, of course, the guaranteed complete motion model is unavailable. If the first option is to be pursued, the motion model must be constructed, completely or partially, based on a history of sensor measurements. If the system is highly predictable, then this approach is more appropriate for allocation of sensing and computational resources. The second option, to act based solely on the current sensory data, is the appropriate course if the system is not predictable at all.

In most real-world applications, there is neither zero nor perfect predictability, and practical decisions must be made regarding whether and how to model the vehicle, sensors, and environment. Additional important decisions regard how far into the past one should go to use sensory data and how far into the future to plan the vehicle's actions.

1.2.3 Bridging of Disciplines

Successful high-speed autonomous navigation requires development and integration of tools from control theory, robotics, computer vision, and systems engineering. Risking oversimplification, each of these fields brings a strength that is largely neglected in the others:

- Control theory deals with modifying the behavior of dynamical systems, and provides a huge body of theory and experimental results in stability, performance and robustness of constrained systems that have adequate mathematical models.
- Robotics provides a wealth of experience and literature in mapping and navigation for mobile robots, especially for low-speed operation in real-world environments using uncertain sensory data.
- Computer vision is the enterprise of enabling machines with the ability to sense, detect, recognize and understand the elements of the environment, especially through processing of images.
- Systems engineering is a structured design and development process for complex, interdisciplinary engineered systems.

Developing reliable systems for high-speed autonomous navigation suggests the need to develop and apply tools from these disciplines outside of their historical scope. For example,

- much of the early research in robotics has been indoors and at slow speeds, reducing the need for considerations of stability and dynamical constraints;
- a large portion of control theory assumes certain environmental conditions, or that uncertainties can be simply modeled; and
- a great deal of computer vision deals with understanding of scenes instantaneously, and do not consider motion models of the camera or the environment.

To be sure, there is much past and current research in each of these fields that do not make these simplifying assumptions. The intersection and integration of these provide the "hard" problems of tomorrow's research. This thesis represents a starting point for such integration.

1.3 Contributions and Organization

The major contributions of this thesis can be summarized as

- a comparative exploration of different control architectures for high-speed autonomous robots, informed by the DARPA Grand Challenge;
- new, model-based, predictive techniques for LADAR-based road following in particular and environment estimation in general; and
- detailed description of Caltech's two entries in the 2004 and 2005 DARPA Grand Challenge, focused on software architecture, technical implementation and overall performance.

The thesis is organized as follows: chapter 2 presents a principled comparison of the major architectures for robot navigation and provides results that illustrate the relative merits of the different approaches, in the context of the themes presented above, through simulation and real-world examples. Chapter 3 presents experimental results of new methods for dynamic model-based estimation of terrain elevation and road geometry. Chapter 4 presents the system design and experimental results from Alice, autonomous ground vehicle finalist in the 2005 DARPA Grand Challenge. A thesis summary and discussion of future research directions are presented in chapter 5.

Chapter 2

System Architectures for Autonomous Navigation

Design and implementation of any robotic or control system involves decisions about how to take the sensory data of the system and compute an input to the system that enables it to achieve the desired task, within acceptable tolerances of stability, performance, and robustness. For simple control systems, these decisions have been well analyzed and a wealth of theoretical and practical research exists for solving such design problems, e.g., for linear single-input single-output (SISO) control systems.

Many real-world research problems in robotics and controls rely on heavily multi-input, information-rich architectures in order to operate effectively in complex, dynamic, and uncertain environments. This typically involves taking very large amounts of raw sensory data and processing them down to a small number of actuation commands. For example, a robot well equipped with image, state sensing, and range detection sensors might have access to on the order of one gigabit (2^{30} bits) per second of raw data, and it might send actuation commands at a rate on the order of one kilobit (2^{10} bits) per second. This represents a remarkable reduction in data by a factor of *one million*, equivalent to halving the data rate twenty times in the process of computing the desired control.

The choices between different approaches to this sensory convergence problem are often made without careful consideration, in large part because there are no formal guidelines (and few informal ones) for making these decisions. The goals of this chapter are (1) to clearly define the major alternative approaches in robot control architectures, (2) to present several important considerations for choosing between them, (3) to provide metrics and a methodology for comparing robot architectures and systems, and (4) to provide real-world case studies, thought experiments, and a mathematical framework for illustration of the relative merits of different control architectures in the context of autonomous navigation. This discussion should clarify

the fundamental differences between different architectures and provide a reference for future research bridging control theory and robotics. Illustrative real-world examples are drawn from the field of entrants in the 2004 and 2005 DARPA Grand Challenge races for autonomous ground vehicles.

2.1 Primary Robot Architectures for Navigation

Specifying a robot architecture imposes constraints on the solution to the robot task at hand [34]. This is advantageous and often necessary for tractability and implementation, but it is important to consider the implications of such constraints on the achievable performance from the system. Whether differences in approaches are significant in terms of the performance of the robot depends on several factors including vehicle speed, constraints, stability, and task difficulty and complexity.

There are four predominant approaches to designing robot control architectures, each of them with unique advantages and disadvantages. The fundamental differences between them lie in the manner in which a large amount of sensory data is reduced to a small set of actuation decisions. These four architectures are *reactive control*, *deliberative control*, *hybrid control*, and *behavior-based control*.

In the late 1960s and 1970s much of the artificial intelligence and robotics research was done in a model-based way and was demonstrated in simulated "blocks" worlds; the predominant approach was to represent the environment completely and explicitly [10, 50]. This top-down *deliberative control* approach relies on perfect models of the environment for perfect completion of robot tasks and reasonably accurate models of the environment for effective completion of robot tasks.

Through the 1980s, the robotics research pendulum swung significantly away from this model-based approach and toward *reactive control* approaches that connected sensor signals directly to actuators and *behavior-based control* methods that connected such reactive systems in parallel. This swing away from deliberative control was in large part due to the influence of Rodney Brooks in his attempts to bring a more balanced approach to artificial intelligence [9, 10]. His influence, along with that of Braitenberg's *Vehicles* [7], led to an explosion in research in bottom-up behavior-based control approaches.

Current research tends to be more balanced between the different control approaches, but too often proceeds without clear principles for the choice of control architecture used. This

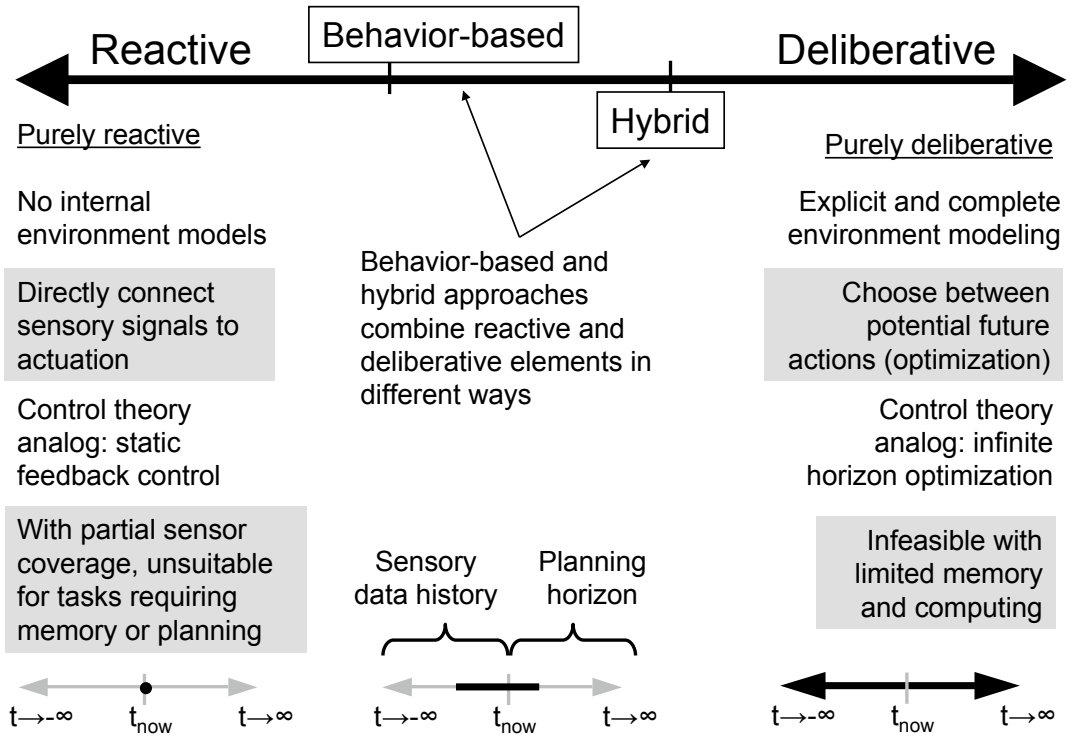

Figure 2.1. Robot control spectrum from purely reactive to purely deliberative control. Position along the spectrum is characterized by the horizon length of past measurements and horizon length of future plans.

chapter clarifies the fundamental properties of, and distinctions between, these approaches so that informed and principled decisions can be made regarding the design of robot architectures.

Fig. 2.1 summarizes the essential properties of the reactive and deliberative extremes of the robot control spectrum. Moving toward the deliberative end of the spectrum means increasing use and dependence on internal models of the environment and on reliable localization. Moving toward the reactive end means increased focus on computing robot action based on current sensory signals.

The degree to which a given robot control architecture is reactive or deliberative is directly related to how far into the past sensory measurements are fused in order to represent the environment. This horizon into the past is denoted T_s here and is measured in seconds. Because purely reactive systems immediately connect current sensory signals to actuation, there is no memory built based on a history of sensory measurements, and T_s is effectively zero. Purely deliberative systems require complete environment models, and in the extreme they build such models over the entire history of measurements ($T_s \rightarrow -\infty$). Practical implementations of

deliberative architectures build models based on a finite but relatively large T_s.

Similarly, reactive and deliberative architectures are distinguished by how much time into the future actions are planned, denoted T_p. Purely reactive architectures plan only the current action, so for these T_p is zero. Purely deliberative control in the extreme is equivalent to infinite horizon optimization, where $T_p \to \infty$. The horizons T_p and T_s provide two primary metrics for evaluating the degree to which a robot control system is reactive or deliberative.

Robot control architecture design in practice typically combines both reactive and deliberative elements, and there are different ways in which this can be done. Two primary methods for combining reactive and deliberative components are *hybrid* methods and *behavior-based* methods.

Reactive, deliberative, hybrid, and behavior-based control architectures are described in detail through the rest of this section. The next sections (sections 2.2 and 2.3) present connections to control theory and a mathematical framework for describing these architectures, respectively. Section 2.6 provides a qualitative comparison of architectures, and section 2.4 presents a framework and illustrative examples for quantitative comparison of architectures.

2.1.1 Reactive Control

Definition 2.1.1. *Reactive control architectures* are those that connect sensory data directly to actuation commands without internal models of the environment or explicitly planning or evaluating between alternative actions.

The reactive control approach is summarized by Matarić as "Don't think, react!" [36]. It provides a direct pathway between sensors and actuators without the expense of maintaining a model of the environment and without planning actions into the future. It "lives in the now" in the sense that it does not organize a past history of measurements and does not evaluate between possible future states and actions. As such, this approach tends to minimize the computational burden and latency associated with mapping and planning.

Rodney Brooks advocated in the early 1990s for designing systems composed of such reactive components, arguing that "explicit representations and models of the world simply get in the way" and that it is better to "use the world as its own model" [10] in designing artificially intelligent systems. Inspiration that provides circumstantial support for this viewpoint comes from biological systems, where behavior such as insect flight is fairly well believed to be a reactive process [21].

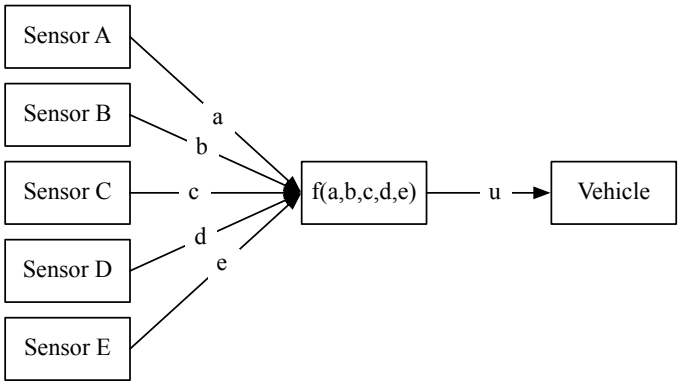

Figure 2.2. Prototypical reactive architecture diagram. Sensory signals are directly converted to actuation commands without internal representation or planning.

It is quite possible, in fact, that biological systems have evolved such reactive components—along with the requisite sensing and actuation—because fast sensor-to-actuator processing is a necessary ingredient for their survival. The field of neuroethology provides a basis for studying the neurology and physiology of such animal behavior. It is clear that humans are not purely reactive beings, though, given their capacity for spatial skills and reasoning, and so it is clear that reactive systems are not the complete path to understanding and emulating human intelligence.

Fig. 2.2 depicts the structure of a reactive control system, characterized by the immediate convergence of sensory data for rapid computation of the system's control signal. In insect flight, this phenomenon is referred to as sensorimotor convergence or visuomotor convergence [21].

2.1.2 Deliberative Control

Definition 2.1.2. *Deliberative control architectures* are those that build a unified model of the environment using a history of sensory data and choose between different future actions by evaluating them against that model.

Deliberative control is characterized by the maintenance of an explicit model of the environment and planning future state and events based on this model. An appropriate analogy is a person making use of a physical map to decide how to navigate through a city to a destination, updating that map as new aspects of the environment are discovered.

In this approach, it is useful to distinguish between two types of sensors used to determine the state of the environment – including the state of the robot with respect to it, or vice versa. These two types are *exteroceptive sensors* and *proprioceptive sensors*, terms borrowed from

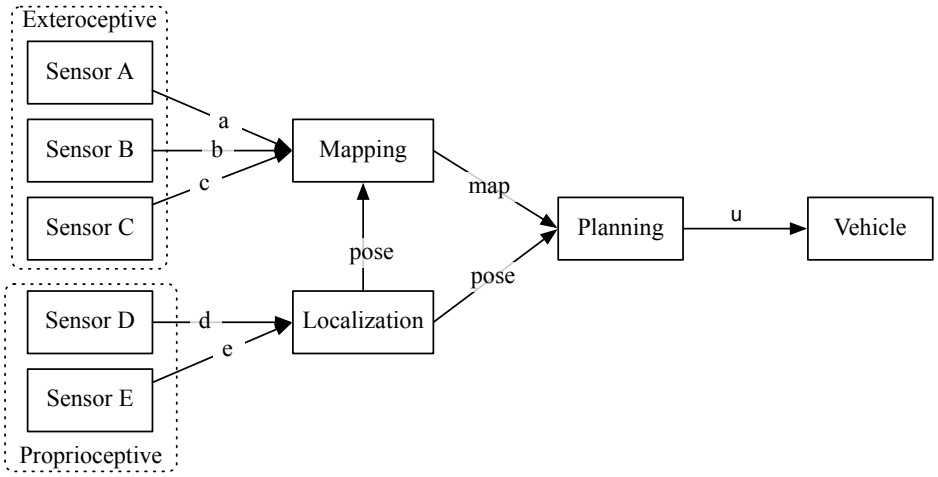

Figure 2.3. Prototypical deliberative architecture diagram. Mapping and planning comprise the defining characteristics of this architecture. Robot action is computed directly from the planning output.

animal physiology. Exteroceptive sensors are those that receive stimuli originating from outside the body; robot examples include laser detection and ranging (LADAR), radio detection and ranging (RADAR), monocular vision, and stereo vision. Proprioceptive sensors refer to those receiving stimulus originating inside the body; robot examples include inertial sensing using accelerometers and gyroscopes, temperature sensors, and odometric sensors.

Fig. 2.3 depicts a standard deliberative control architecture, in which a history of measurements from all sensors is gathered to update a single map of the environment. The vehicle pose (position and orientation) is determined with respect to this map, and planning is done to choose a suitable or best action through this environment.

Deliberative control as described above calculates robot action directly from the planning output. This is a *feedforward* approach to robot control, in which uncertainty in the vehicle dynamics and changes in the environment are handled exclusively through regularly recomputing new plans for the vehicle to execute. This approach typically requires accurate models of vehicle dynamics and fast replanning for stable and high-performance navigation.

A basic feature of deliberative control is optimization, since the planning component chooses between a set of possible actions to pick the best one. The parameterization of the possible options for optimization is a choice; much of the robotics literature frames this optimization as a choice over a finite set of options, which could be small or large. However, continuous parameterizations are also possible and may find more optimal solutions.

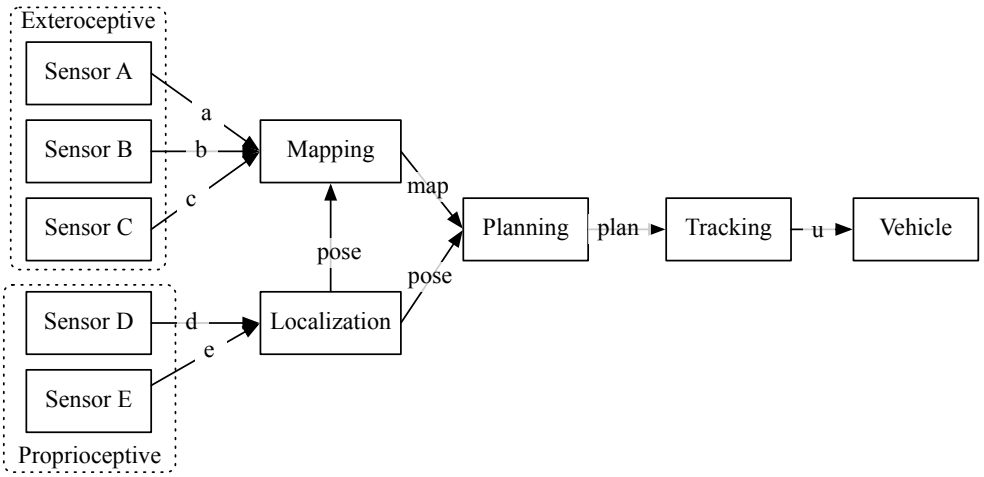

Figure 2.4. Prototypical hybrid architecture diagram. This architecture combines a high-level deliberative component for computing plans and a low-level reactive component for tracking these plans.

Path planning is a central research topic in deliberative control, and significant research contributions in this area have been made by researchers in both robotics and control theory. In the control and dynamical systems community, recent advancements have been made in computing and algorithms for optimal path planning that handle real-time dynamic and spatial constraints in the form of receding horizon control [42]. These results leverage a combination of ideas from control theory and tools for constrained nonlinear optimization, and can be tailored for high-speed autonomous navigation, where dynamic real-time constraints and stability considerations must be taken into account to avoid system failures such as crashes and rollover.

2.1.3 Hybrid Control

This strictly deliberative control approach of section 2.1.2 relies on accurate models of the vehicle and environment for high performance. In real-world applications there is always uncertainty in these models. Environment map uncertainty is inherent due to imperfect sensing and approximate map representations, and vehicle models are necessarily approximate due to the complexity of detailed modeling of vehicle-terrain interaction.

Feedback control provides an effective means to manage and compensate for model uncertainties in vehicle dynamics, by computing actuation based on the error between the desired plan and the current vehicle state. The combination of high-level planning with such low-level tracking is an example of synthesizing reactive and deliberative components, and is called hybrid

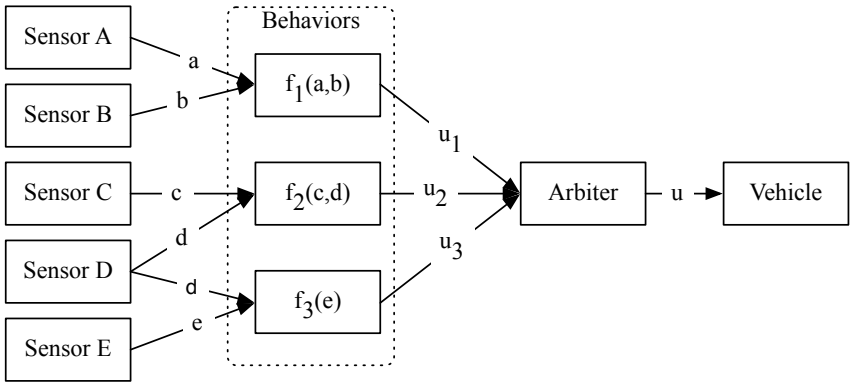

Figure 2.5. Behavior-based architecture diagram. Reactive and/or deliberative components (behaviors) are connected in parallel, and robot control is computed from the outputs of these behaviors.

control [36].

Definition 2.1.3. *Hybrid control architectures* are those that combine high-level deliberative planning with low-level tracking of the desired plan.

Fig. 2.4 depicts the organization of a hybrid control architecture. It is identical to the deliberative control architecture of fig. 2.3 with the addition of the low-level tracking component. The use of low-level reactive control in hybrid architectures is important when there is *uncertainty* in the models used to perform the high-level planning. The fundamental role of this low-level feedback control is to compensate for such uncertainty, and the degree to which reactive control elements are important is proportional to this uncertainty. Important connections of this hybrid approach to inner-outer loop control design are discussed in section 2.2.

2.1.4 Behavior-Based Control

The other major approach for combining reactive and deliberative elements for autonomous navigation is *behavior-based* control. Behavior-based architectures are commonly associated with robotics and artificial intelligence research [4, 10]. They consist of a set of behaviors that each seek to achieve distinct goals. An example set of behaviors might include waypoint following, obstacle avoidance, road following, and rollover prevention.

Definition 2.1.4. *Behavior-based control* consists of a number of goal-based behaviors that connect sensory data to robot action. These behaviors can be reactive or deliberative, and robot control is computed as some function of the outputs of all behaviors.

The output of each of the behaviors must be combined in order to compute a single set of robot actions at any given time. One leading method for doing this synthesis is with an arbiter framework. The arbiter takes votes from each behavior over a parameterized space of commands, takes a weighted average of the votes, and selects the best command from the weighted average. This approach is presented in a formal framework in section 2.3, and [48] is a comprehensive reference. Another popular method for combining behavior outputs is to prioritize the behaviors and to choose the behavior output to command the robot based on the current operating condition. This behavior switching method is called *subsumption* and was first presented by Brooks [8].

Behavior-based robotics is historically associated with purely reactive behaviors, that is, those that do not maintain an explicit internal representation of the environment. However, behavior-based research can and often does include behaviors in which an internal state is maintained based on some subset of sensory data [35, 48]. Fig. 2.5 represents a behavior-based robot control architecture, where some of the behaviors f_i might be reactive and others might be deliberative. Behavior-based architectures combine these different components in parallel, rather than the serial nature of hybrid architectures.

A central challenge in either form of behavior-based robotics is to understand and manage the interaction between different individual behaviors in order to exhibit desired overall performance. This is especially difficult when inertial and dynamic effects are significant in the face of state and input constraints. The examples of section 2.6 illustrate this consideration.

2.1.5 Recap: Methods of Convergence

The architectures presented above differ in the manner in which they condense the large amount of sensory data to a small amount of control data. This process is referred to as *convergence*. Reactive systems perform convergence immediately and without intermediate representation, and therefore may perform poorly if the information required for effective completion of the robot task is not provided all at once (and continuously).

Deliberative systems condense all of the sensory data into an intermediate representation, and the vehicle's action is decided based on that representation. This places a heavy burden on the sensor fusion task and the maintenance of faithful state estimates and maps.

Behavior-based systems perform convergence in two stages. The first combines data from (potentially multiple) sensors into single behaviors, and the second combines commands or functions of commands. These behavior-based systems can be said to be primarily performing

command fusion rather than the *sensor fusion* of deliberative systems.

Another alternative for behavior-based systems with deliberative behaviors is so-called *utility fusion* [47], in which behaviors evaluate over future states of the system and an arbiter determines a temporally consistent sequence of actions to achieve those states. A final, as yet not fully explored, option for deliberative behaviors is to evaluate over paths and perform *path fusion* in the arbiter to determine vehicle control. It is unclear what advantages this approach might have over deliberative methods.

2.2 Robot Architectures and Control Theory

Robotics and control theory are closely related fields. Some of the rich set of tools in control theory and dynamical systems see daily use in commercial robotics. For example, PID control is employed in almost every motor used in robotics applications, and a lot of results in control of nonholonomic kinematic systems fall in the overlap between research in control and robotics. Developments in each field have interesting connections with each other, some of which present a rich source of further investigation. In fact, the robotic architecture approaches presented above have direct analogs in the language of control and dynamical systems.

Remark 1. The reactive approach of robotics is equivalent to the notion of control computation via *state feedback* or *output feedback* for dynamical systems.

Consider the linear system described by eqns. (2.1). Here $x \in \mathbb{R}^n$ is the state of the system, $u \in \mathbb{R}^m$ is the input to the system, and $y \in \mathbb{R}^p$ is the output of the system. Pure state feedback for control (to the origin) of such a system is described by $u = Kx$ or $u = Ky$, where K takes the appropriate dimension. In the case of state feedback, an estimation process constructs an estimate of the state, $\hat{x}$, based on the measurement y.

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{2.1}$$

The state feedback controller is reactive in the sense that it is simply computing the output based on the current state of the system, without maintaining or using any internal memory for the control computation, and without evaluating and choosing over potential future states of the system.

In controls parlance, the state feedback controller is an example of *static feedback*, because

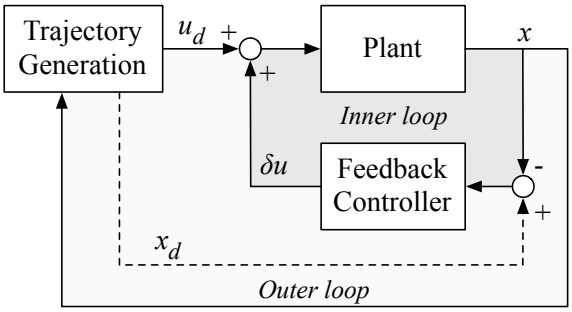

Figure 2.6. The two degree of freedom design for real-time trajectory generation and control. The input to the plant is composed of a feedforward term u_d from the properties of the trajectory and a feedback term δu based on the error between desired and actual state.

there are no dynamics associated with the control computation. In dynamic feedback control, the controller itself possesses dynamics and updates an internal state vector used in control computation. Such a dynamical system can be viewed as something more akin to deliberative control, although the internal model does not represent the geometry of the environment.

The deliberative control framework is equivalent in controls to pure receding horizon control, where plans are made out to some finite time horizon, and vehicle control is computed directly from these plans. As the horizon extends to infinity, the local optimal control problem becomes a global optimization; this is exactly analogous to global D* graph search methods in robotics.

In control theory as well as robotics, uncertainty in vehicle dynamics and in the environment can affect navigation performance for purely deliberative systems. In robotics, coupling such high-level planners with low-level reactive control is a type of hybrid robot control. This is equivalent to the "two degree of freedom" design approach that couples an outer trajectory generation loop with inner feedback control to the trajectory. This approach is illustrated in fig. 2.6.

2.3 A Mathematical Framework for Planning Architectures

A mathematical framework is presented here to provide tools for analysis of the comparative qualities of reactive, behavior-based, hybrid, and deliberative control. The framework is presented from the perspective of dynamical systems and control theory, but gives some useful insights into the design of architectures for autonomous navigation.

In all of the presentation below, the path planning and execution is presented in the context

of a system with dynamical description

$$\begin{aligned} x_{k+1} &= f(x_k, u_k) \\ y_k &= g(x_k). \end{aligned} \tag{2.2}$$

Generally speaking, the state vector x_k will encode the state of the vehicle as well as the state of the world. Both state vector and measurement vector might be very high dimensional, e.g., encoding hundreds of individual state elements and measurements. In addition to the dynamic constraints represented by eqns. (2.2), constraints on the input and state of the system are generally also present, as well as constraints that couple the state and input:

$$\begin{aligned} x_k &\in \mathcal{X} \\ u_k &\in \mathcal{U} \\ g(x_k, u_k) &\leq 0. \end{aligned}$$

In a behavior-based system, the set of behaviors can be interpreted as evaluating a cost function over the space of input values u_k. Let this cost function for behavior i be denoted

$$\mathcal{C}_{B_i} = \mathcal{C}_{B_i}(x_k,\ u_k).$$

In general, a behavior might evaluate over the set of input values, $\{u_i\}_{i=k}^{i=k+N-1}$, from the current time out to some future time $k+N-1$. The solution that minimizes $\mathcal{C}_{B_i}$ as $N \to \infty$ is the globally optimal strategy for the vehicle with respect to that behavior. Computing such a solution is computationally intractable except in special cases; for example, if the system described by eqns. (2.2) is linear, the cost $\mathcal{C}_{B_i}$ is quadratic in the state and input, and constraints are absent, then the linear quadratic regulator is the optimal strategy for controlling the state of the system to the origin. Note that constraints as described by eqns. (2.3) can be included as regions of infinite cost.

Reactive systems by definition do not maintain an internal representation of the environment, but rather connect only the current measurements available to actuation according to the cost function $\mathcal{C}_{B_i}$.[1] Additionally, since reactive systems do not evaluate over future possibilities in the state and input space, the behavior cost is also only evaluated for inputs at the current time $i = k$. In words, reactive behaviors have no express notion of either history or memory, nor

[1] Though the cost function as stated does not include the measurements, in reactive behaviors the current measurement set can be written as a subset of the state vector that does not have any modeled dynamics.

of future possibility. This is a particular limitation of reactive behaviors, especially in the case where there are important dynamical constraints that need to be considered in the operating envelope of the controlled system.

Generally, behaviors in a behavior-based system can be deliberative as well as reactive, both encoding a representation of the environment as well as evaluating over future possible states of the vehicle depending on the input applied. The output of all of the behaviors must be combined in some intelligent way for the vehicle to decide which input commands to send to the vehicle. A common way of doing this is the arbiter framework, which is described above and also presented in several papers in the literature, including the distributed architecture for mobile navigation (DAMN) as presented by Rosenblatt ([47, 48]), experimental testbeds including the Intelligent Vehicle Safety Technologies DARPA Grand Challenge entry [26], and for autonomous navigation of the Mars exploration rovers *Spirit* and *Opportunity* [18, 31]. A common approach for the arbiter is to combine the costs evaluated from all of the behaviors with a weighted average:

$$\mathcal{C}_B(u_k) = \sum_i \alpha_i \, \mathcal{C}_{B_i}(x_k, \; u_k).$$

The process of combining the outputs of behaviors in this way can be interpreted as *command fusion* as opposed to sensor fusion [48]. The control applied by the arbiter is in this case the result of taking the minimum of the combined cost:

$$u_k^* = \min_{u_k \in \mathcal{U}} \; \mathcal{C}_B(u_k). \tag{2.4}$$

Deliberative planning and control strategies, on the other hand, are essentially a sensor fusion approach as opposed to command or utility fusion. With a deliberative approach, instead of the cost being encoded in a distributed fashion, it is encoded centrally in a map of the environment. In the design for Alice, the map is a spatially encoded constraint set where each cell encodes the maximum speed at which any part of the vehicle can traverse the cell. The cost is evaluated over a family of continuously parameterized paths and the lowest cost path is chosen from this family.

The deliberative strategy for navigation can be written as

$$\{u_i^*\}_{i=k}^{i=k+N-1} = \min_{\{u_i\} \in \mathcal{U}^N} \; \mathcal{C}_D \left(\{u_i\}_{i=k}^{i=k+N-1} \right). \tag{2.5}$$

Side-by-side consideration of eqn. (2.4) and eqn. (2.5) indicate that both are essentially performing an optimization over the (parameterized) space of input values to the vehicle. The fundamental difference between the two approaches is that the deliberative approach uses a longer horizon length for its optimization. The deliberative approach can optimize both vehicle state and input values out to a significant horizon that is on the order of the sensing range, which allows it to take advantage of all of the sensory data available. The behavior-based arbiter approach collects votes over the input space for only the current time step. Compared to the deliberative approach, this can lead to sequences of commands that are suboptimal or even dynamically infeasible.

Since optimizing over a longer horizon is known to provide a more globally optimal solution while satisfying dynamic constraints, one might be tempted to say that the deliberative approach is always superior. However, there are practical limitations of large-scale optimization that might make the deliberative approach difficult to implement efficiently, as compared to reactive or behavior-based strategies. Further, in those situations where there is high uncertainty in environment models, plans may be optimal with respect to the model but suboptimal with respect to the environment itself.

2.4 Quantitative Architecture Analysis

This section provides a framework for analysis of different robot control architectures and metrics for dynamic feasibility and predictability that can help guide this analysis.

The overall performance of an autonomous system is given by its ability to perform its given tasks. For autonomously navigating robots, the most fundamental task is to quickly and safely travel from a start position to a goal position. Total mission performance with regards to this task can be evaluated using metrics such as percent mission success, mission duration, and average mission speed.

A difficulty with formal design of robot architectures is that the map between detailed system parameters (such as sensor rate or actuation speeds) and overall performance is both unknown and high-dimensional. This difficulty motivates the identification of salient intermediate metrics that we can relate to overall performance for a given control strategy. Such intermediate metrics, which depend on factors of the environment and of the vehicle, can be assigned to the previously identified concepts of predictability and dynamic feasibility (a.k.a. agility).

2.4.1 Predictability Metrics

As stated previously, predictability reflects a robot's ability to completely and accurately model the environment. Notionally, predictability will scale inversely to the complexity of the environment and directly with the vehicle's mapping ability, i.e.,

$$P \sim \frac{\text{mapping ability}}{\text{environment complexity}}.$$

Mapping ability can be assessed by comparing estimated maps of the environment with a map containing the "true" state of the environment. Examples of such comparisons are presented in figs. 2.7 and 2.8. Fig. 2.7 shows the map obtained by Alice in an early section of the 2005 DARPA Grand Challenge and the presumed ground truth based on the speed limits provided by DARPA in the route description. The difference between actual and estimated speeds is zero for most cells in the map because the Alice's estimated maximum driving speed over this terrain is greater than the DARPA-imposed speed limit. The greatest deviation appears in the cells for which no range data has been returned, through which Alice is programmed to drive slowly.

Several metrics are available for evaluating the quality of the mapping. One such metric is the integrated and normalized error over all the N cells in the region of interest $\mathcal{I}$,

$$J = \frac{1}{N} \sum_{(i,j)\in\mathcal{I}} \left| L^*(i,j) - \hat{L}(i,j) \right|, \tag{2.6}$$

where $L*$ is the actual value at that location in the map (e.g., of maximum traversable speed or of elevation) and $\hat{L}$ is the estimated value. For the region presented in the rightmost plot of fig. 2.7, the metric above evaluates to 0.55 cm/s average absolute error.

Fig. 2.8 shows the output of the same system at a later part of the 2005 Grand Challenge course. The mapping metric of eqn. (2.6) for this map evaluates to 44.2 cm/s average absolute error. Knowledge of additional circumstances leading up to this state provides insight into the factors affecting predictability. In the case of both fig. 2.7 and fig. 2.8, no obstacles actually existed on the interior of the route boundary in global coordinates. The appearance of obstacles in the estimate of fig. 2.8 is due to massive state estimate drift, which critically affected the vehicle's mapping ability and therefore predictability. More details on this failure mode are presented in chapter 4.

The predictability metric of eqn. (2.6) has some limitations, and alternative metrics can be devised that more accurately reflect the considerations for predictability. In particular,

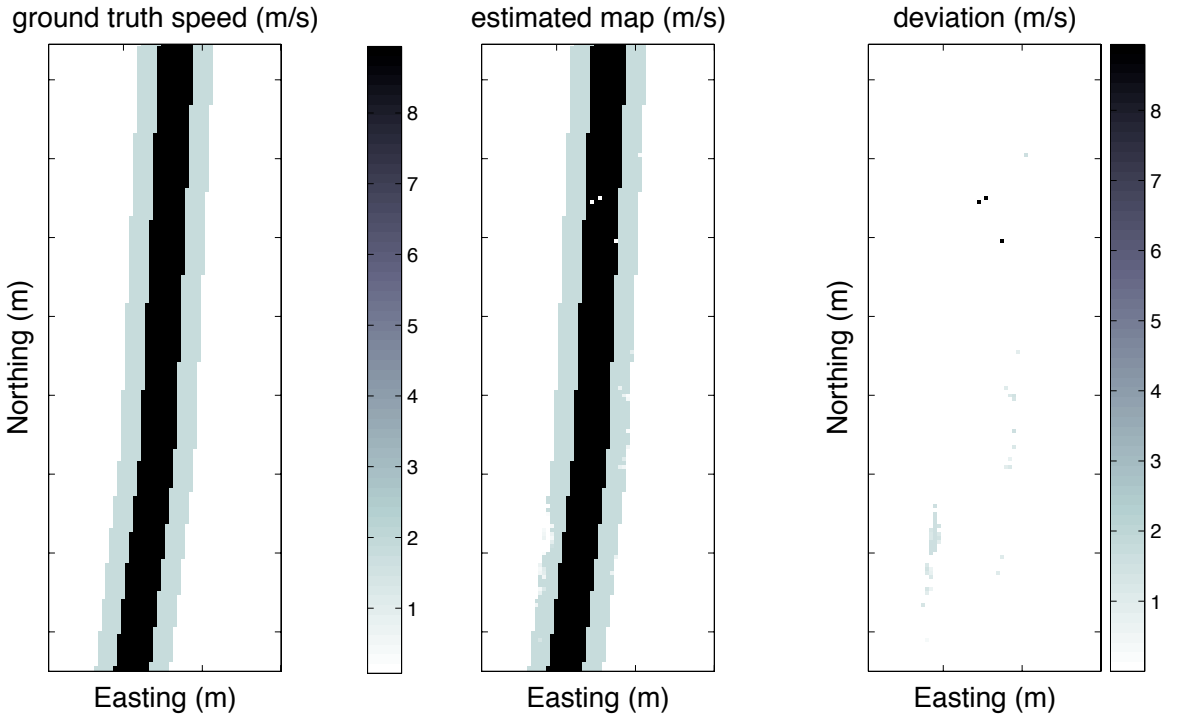

Figure 2.7. Presumed ground truth and estimated speed map for high predictability area. These experimental data are from an early section of Alice's run in the 2005 DARPA Grand Challenge, near waypoint 22 and approximately two minutes into the race. The right plot shows actual minus estimated speeds.

environment complexity does not appear in this metric. However, environment complexity could be interpreted as equal for the two examples presented since no obstacles appeared on the interior of the course.

A fundamental limitation of the metric of eqn. (2.6) is that it does not take into account the essential consideration of the past time horizon of sensory measurements. It provides only an aggregate measure of the final mapping ability, but does not connect this aggregate measure to specific design parameters so that these can be chosen to maximize mapping ability. Factors such as sensor range, angular field of view, accuracy, and time horizon T_s for integration are all important to consider in designing a system that is able to map its environment, but few formal connections between these design variables and predictability are established. One intuitive hypothesis is that better sensor coverage over small time scales will provide better predictability, especially in dynamic environments. Such systems would also allow persistent sensing of static and moving obstacles in order to better distinguish the two.

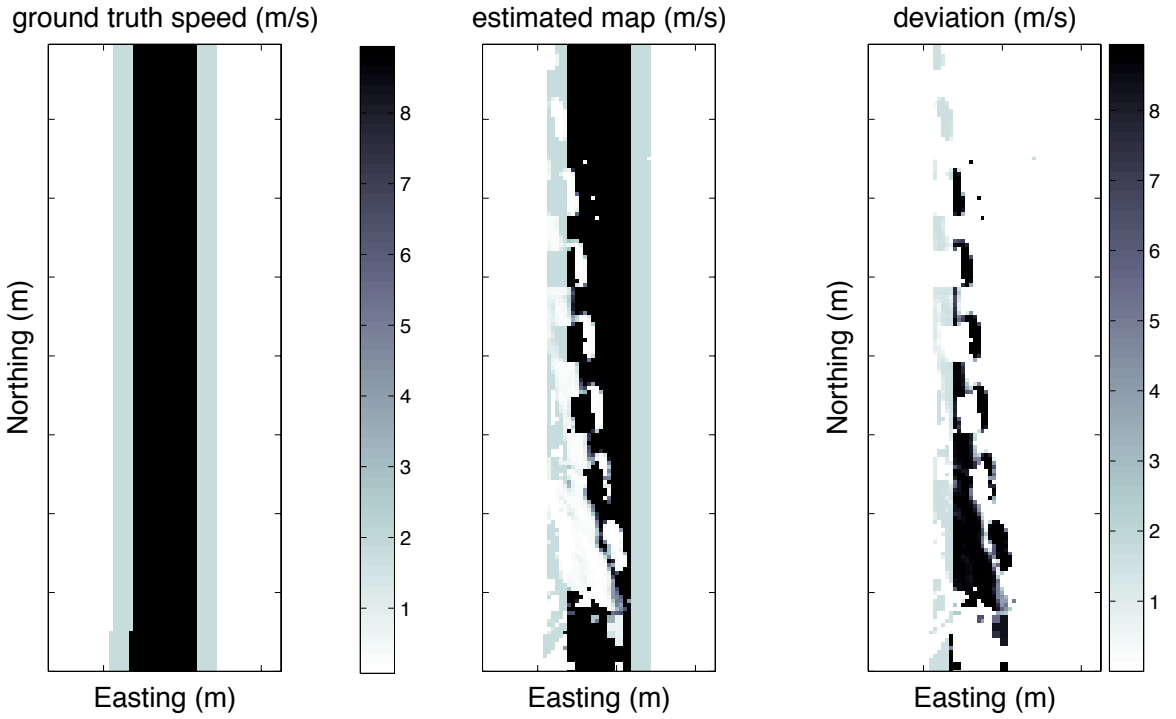

Figure 2.8. Presumed ground truth and estimated speed map for low predictability area. These experimental data are from the final moments of Alice's run in the 2005 DARPA Grand Challenge, near waypoint 174 and approximately 32 minutes into the race. The right plot shows actual minus estimated speeds.

2.4.2 Agility Metrics

Similar to predictability, agility represents an intermediate metric that can be used to inform the choice of robot control architectures for a given task or set of tasks. Notionally, agility refers to the ability of a vehicle to avoid an obstacle that suddenly appears in the obstacle field of view. This ability is inherently dependent on the dynamics of the vehicle, so it is useful to begin with a simple dynamical model of vehicle motion. One such model is

$$\dot{x} = v\cos\theta \tag{2.7}$$

$$\dot{y} = v\sin\theta \tag{2.8}$$

$$\dot{\theta} = \frac{v}{L}\tan\phi \tag{2.9}$$

$$\dot{\phi} = \omega, \tag{2.10}$$

where (x, y) are the position of the vehicle's rear axle in the plane, θ is the vehicle heading, and ϕ is the vehicle steering angle. Consistent with typical cars, constraints are imposed on steering

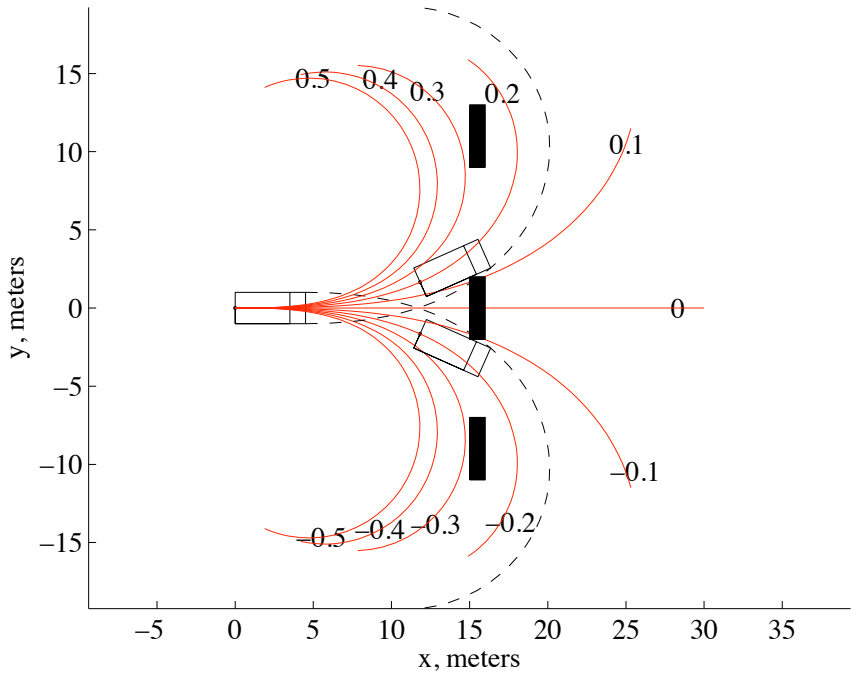

Figure 2.9. Illustration of maximum swerve maneuver for a range of steering rate constraints. Paths of the center of the rear axle (solid) and the outside corner of the vehicle (dashed) are shown for various maximum steering rates.

($\phi \in [-\phi_{max}, \phi_{max}]$) and steering rate ($\omega \in [-\omega_{max}, \omega_{max}]$).

One possible metric for agility is the lateral distance achievable in a maximum swerve maneuver from a straight steering angle. This situation is depicted in fig. 2.9. The vehicle starts at the origin traveling at some assumed constant speed in the x-direction. At some fixed distance, an obstacle is detected and a maximum steering rate is applied. At this fixed distance, the maximum lateral deviation of the vehicle depends on the maximum steering rate and the vehicle speed.

The dependence of swerve avoidance performance through a series of simulated examples is shown in fig. 2.10. The simulations are performed on a model based on eqns. (2.7–2.10) at various speeds and maximum steering *rates* for a fixed detection distance to the obstacle of 12 meters. The vertical axis depicts the maximum lateral distance that the entire vehicle (assumed to be 2 m wide and 4.5 m long from rear axle to front) can clear for the given values of maximum steering rate and speed.

This example clearly illustrates that agility decreases with increasing speed and increases with increasing maximum steering rate. This suggests an alternative metric (other than the ag-

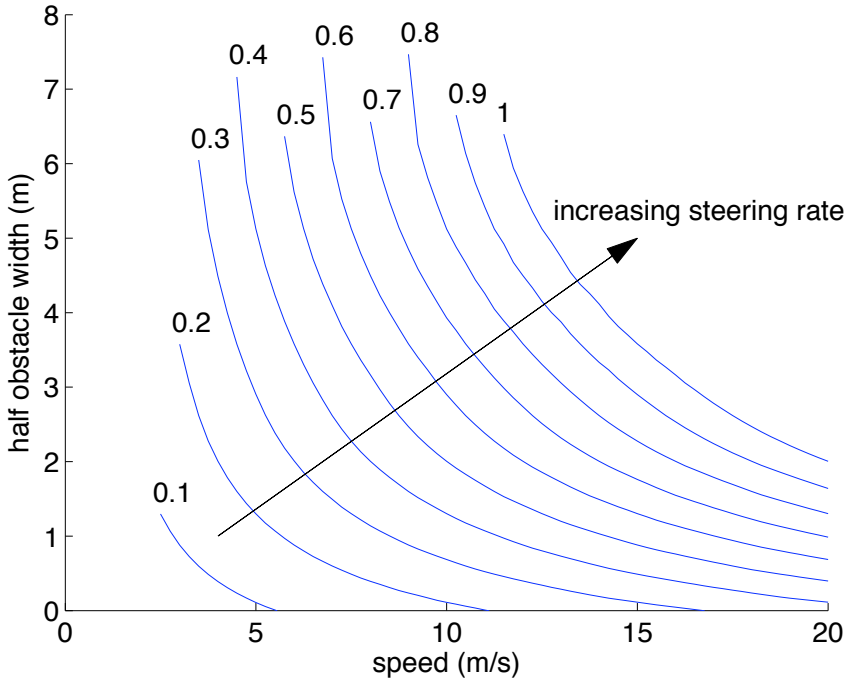

Figure 2.10. Relationship between various factors related to agility. The ability to swerve around obstacles of various widths is shown as a function of speed and maximum steering rate.

gregate metric of maximum lateral swerving distance) that depends on more specific parameters, namely,

$$A = c\frac{\omega_{max}}{v}, \tag{2.11}$$

where c is a nondimensionalizing constant. This agility metric can also be interpreted as relating the "time to avoid" a fixed width obstacle by swerving. A similar metric could be applied with a time to avoid by stopping criterion by replacing ω_{max} with a_{max}, the maximum longitudinal deceleration.

A limitation of eqn. (2.11) as a metric for agility is that it does not include the significant effect of non-zero reaction time on agility. The size of this reaction time (including sensing, detection, mapping, and replanning) is a critical consideration for influencing a vehicle's agility, and other choices of metrics for agility could include this time delay as a factor.

Alice's maximum physical steering rate was about 0.45 rad/s which, at the 9 m/s speed limit indicated in the map of fig. 2.8, evaluates to an agility measure of 0.05 according to the metric of eqn. (2.11). The concrete barrier that she collided with was detected by the short range LADAR a mere 0.5 seconds before impact (midrange sensors had failed earlier and did

not detect the barrier, and the long range sensor misregistered the location of the barrier due to bad state estimate). The combination of limited agility with severely limited predictability served as the downfall for a highly capable autonomous vehicle.

2.5 DARPA Grand Challenge: Case Studies

To provide practical illustrations of the differences between these two approaches, the following sections include basic descriptions of the architectures designed and implemented for the two vehicles entered by Team Caltech into the 2004 and 2005 DARPA Grand Challenge autonomous vehicle races. The 2004 race entry, Bob, used a behavior-based solution to the navigation problem, and the 2005 entry, Alice, used an hybrid approach with several novel contributions in its implementation.

There are, of course, limitations to drawing conclusions about robot architectures solely from the performance of these two vehicles (and others in the Grand Challenge races), but they are useful example to illustrate and guide a comparative analysis of the two approaches. To this end, the design of Bob's navigation software is described in the next section along with a brief overview of the navigation architecture for Alice.[2]

2.5.1 Bob: Behavior-Based Navigation

Bob is an autonomous vehicle designed and built by Caltech undergraduates for competition in the 2004 DARPA Grand Challenge. It was built on a 1996 Chevrolet Tahoe chassis and was equipped with computer-controlled actuation for throttle, brake, steering, and transmission. Additionally, a suite of laser detection and ranging (LADAR) and Firewire camera sensors were mounted on the front of the vehicle and connected to a set of computers. Bob's navigation software used a behavior-based architecture in the pathway from raw sensory data to actuation commands.

Fig. 2.11 depicts the software architecture used by Bob in the 2004 Grand Challenge. Four types of behaviors were implemented and run in a parallel architecture. Each behavior output a quality measure ("goodness") for a set of 15 steering commands according to the behavior's goals.

1. A waypoint-tracking behavior. This behavior assigned steering goodness according to how well that steering command led the vehicle to the next waypoint at some minimum distance

[2] A detailed description of Alice's navigation algorithms is presented in chapter 4, with particular focus on the novel contributions therein.

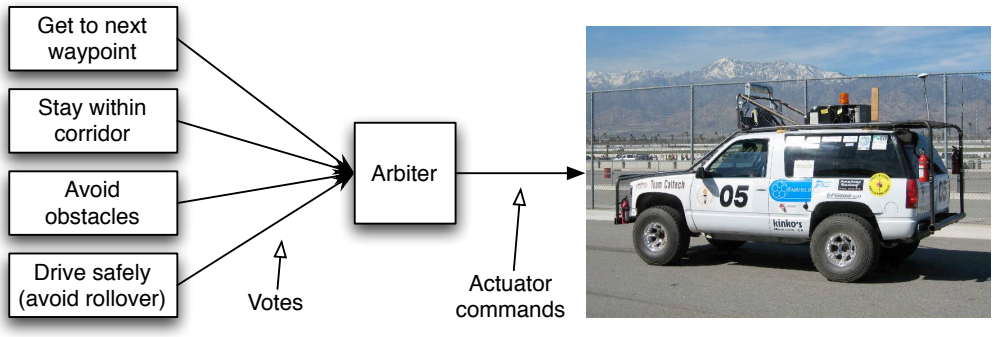

Figure 2.11. Summary of the behavior-based architecture implemented for Bob, Team Caltech's 2004 DARPA Grand Challenge finalist entry (shown at right).

ahead of the vehicle.

2. An obstacle-avoidance behavior. This behavior assigned steering goodness according to the distance over which a given steering command enabled the vehicle to avoid obstacles.

3. A corridor following behavior. This behavior interpreted the corridor boundaries as obstacles and evaluated steering commands in the same way as the obstacle avoidance behavior.

4. A dynamic-feasibility behavior. This behavior assigned quality to steering actions according to the chance of rollover given the vehicle speed and roll angle.

Each behavior evaluated a common, finite set of steering angle commands and reported the goodness for each steering command in the set as well as a maximum safe speed according to that behavior. An arbiter component took a weighted average goodness over the steering angles, at 10 Hz. The arbiter selected the steering angle corresponding to the maximum goodness and selected actuation commands to track the minimum reported safe speed at that steering angle.

Practical Lessons from Bob

Several lessons came from the experiences of designing Bob for the 2004 Grand Challenge.

- A systems engineering approach is necessary for a project of this scope.

- Emphasis should be placed on thorough and extensive testing in real-world conditions, through a series of tests designed to demonstrate improved capabilities, ideally while retaining all previously demonstrated capabilities. These tests should represent realistic goals that have quantifiable metrics.

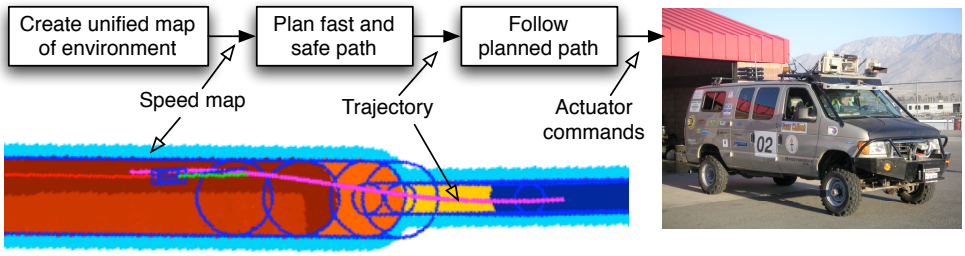

Figure 2.12. Simplified depiction of hybrid architecture used for navigation on Alice.

- Balance should be made between the time spent discussing versus deciding among options. Overly ambitious goals sometimes led to choices that didn't need to be made.

The performance of Bob in the race led to two additional conclusions. One was that thorough testing of individual components is critical to overall success, and a second is that dynamic feasibility considerations are necessary for high-speed operation. In particular, Bob's obstacle avoidance evaluated over arcs in the terrain, while rate limit constraints on the steering meant that at higher speeds the arcs were not all feasible. In the higher speed regime, clothoids[3] satisfying the initial steering angle provide paths for evaluation that better reflect the vehicle's capabilities.

2.5.2 Alice: Hybrid Navigation

Fig. 2.12 is a simplified depiction of the architecture used for Alice, Team Caltech's 2005 Grand Challenge entry. The navigation software was built around central mapping and planning modules, both hallmarks of deliberative navigation, and low-level reactive trajectory tracking. The mapping module maintained a unified representation of the environment around the vehicle by processing sensory data, and the planning module optimized over a parameterized family of future vehicle paths. Alice's navigation software is described in detail in chapter 4.

Not shown in this diagram are the various sensors, which are configured essentially along the lines of the deliberative architecture of fig. 2.3. Also not shown are the supervisory control components, described in more detail in chapter 4, which served as a safeguard when certain assumptions of the deliberative model were violated. Supervisory control compensated for the highest levels of unpredictability of our perception process, for example, when spurious obstacles were placed in the map and falsely indicated that no route forward was possible.

[3] Clothoids are curves with piecewise constant rate of change of curvature.

Practical Lessons from Alice

The experience of designing and implementing Alice for the 2005 Grand Challenge led to several important lessons. One is that designing a system from the ground up for rapid development and testing pays huge dividends; this was a major difference between Bob and Alice. Another is that caution should be kept against system overdesign. A lot of effort in Alice's design went into developing advanced capabilities so that there was very little reliance on the RDDF (or on any other *a priori* information); the mapping and planning capabilities were successful in navigating tight obstacle fields at significant speeds where dynamic constraints are important, but these capabilities far surpassed what was necessary for completion of the course, and in the end an unforeseen state estimate glitch led to the vehicle's demise on race day.

Another fundamental lesson is that there are essential differences between control architectures that must be considered in the design of autonomous systems; these differences are the subject of this chapter. A summary is that deliberative methods are necessary for some systems in order to satisfy spatio-temporal constraints, and that behavior-based methods are useful for dealing with uncertainty in sensing and environment models. Deliberative control methods are superior in many cases when accurate and reliable models of the system are available and dynamic constraints must be satisfied, but they are fragile when model assumptions are violated. This point underscores the need for both high-level (behavior-based) supervisory control and low-level (reactive) trajectory control.

2.5.3 Other Teams' Approaches

The design approach for Alice was shaped by the lessons learned from fielding a team for the 2004 Grand Challenge race, and by the shared experiences of other teams in that event, notably the technical report published by the Red Team [52, 53] and the relative overall success of path-centric versus behavior-based approaches.

The hybrid approaches to solving the Grand Challenge centered on building a grid-based or obstacle-based map of the environment and performing a search through that map for an optimal path. The field of finalists for the 2005 race partially reflected a convergence of system-level architectures to this approach; 17 of the 23 team technical papers [5] (including those from the five vehicles that completed the course) describe various hybrid implementations.

The top three finishing teams from Stanford and from Carnegie Mellon (Red Team and Red Team Too) represented very similar hybrid architectures, with occupancy grid-based methods to determine regions in the map that are and are not traversable. Two primary differences

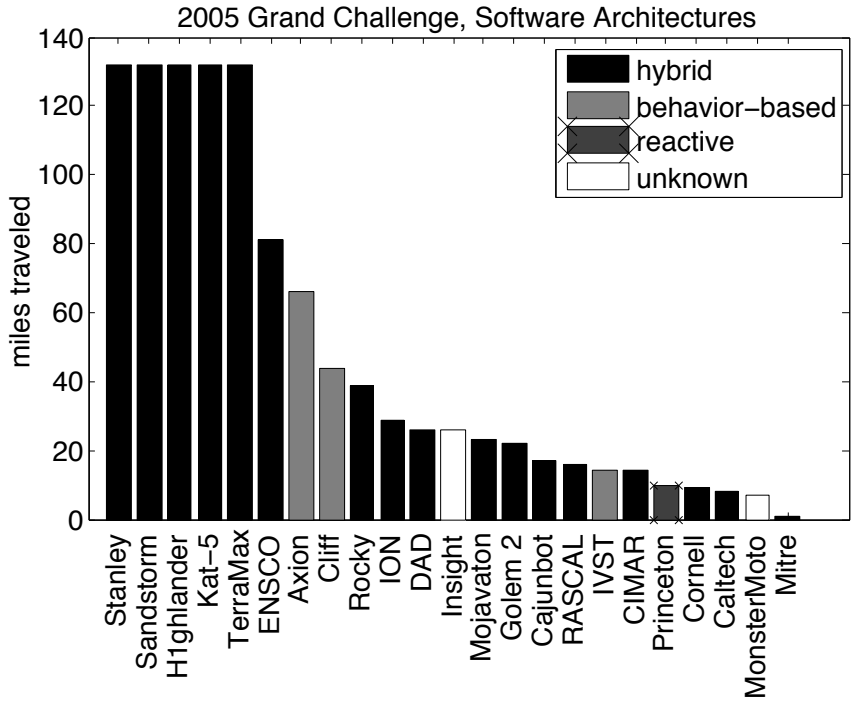

Figure 2.13. Distances traveled and control architectures used in the 2005 Grand Challenge Event, based on official race results and team technical papers.

between these approaches were the gimbaling of sensors and extensive use of *a priori* data and route preplanning by the Red Teams.

See fig. 2.13 for a chart of distance traveled and architecture employed for each of the race vehicles. Based on the technical papers, three teams (Axion, Virginia Tech's Cliff, and IVST) implemented a primarily behavior-based navigation architecture, and Princeton University implemented a purely reactive architecture. These alternative approaches are a source of valuable experience and experimental data, and might provide some insight into the relative merits of different approaches.

One particularly novel sensor suite implementation in the 2005 Grand Challenge attempted to provide sufficient instantaneous coverage of the environment. Team Digital Auto Drive's entry employed a single package of 64 LADAR sensors arranged in a 360 degree horizontal and 20 degree vertical field of view. This sensor suite was mounted on the top of a pickup truck cab and rotated about a vertical axis at 600 RPM. Because of this ubiquitous sensor coverage, it was not necessary for the vehicle's navigation software to build a model of the environment over any significant period of time. As a result, there was more flexibility in choice of planning

architecture. Despite the advantage of not requiring a model of the environment, the team's technical report does indicate that map building took place over multiple poses of the robot; more details of this sensor implementation and employment are available at [20].

2.6 Qualitative Comparison of Architectures

Several conclusions can be made from the recent research experiences of Team Caltech with both behavior-based and hybrid architectures. These are not only particularly relevant to the Caltech team's experience in the DARPA Grand Challenge, but are also generally applicable to research in autonomous robotic navigation. Practical considerations for

Computation

Maintaining an internal representation of the environment and performing optimization over potential robot paths imposes a considerable cost in terms of memory and computation as opposed to reactive behavior-based systems. A combination of efficient representations and modern-day computing capabilities can address these costs, but at the expense of additional system latencies. Reactive control provides an advantage in terms of system latency (as in the example of insect flight), but it is not as "smart" in that it does not consider the future, remember the past, or explicitly consider temporal constraints. The evolution of insect flight, to follow the example, has been such that sensory and motor capabilities more than compensate for these limitations for basic navigation and flight control.

Uncertainty Management

Deliberative/hybrid architectures tend to be connected in a serial nature ("sense, map, plan, act"), and in certain environments overall performance of the vehicle can be better predicted and managed as compared to behavior-based architectures. Because of this serial nature, the connections between components can be well defined and understood, as there are fewer components connected to the (uncertain) environment. Uncertainty in deliberative systems is primarily situated in the model of the environment rather than the control system, partly due to well understood properties of optimal and feedback control systems.

In contrast, managing and understanding the interaction of behaviors in behavior-based systems can be more difficult. In these systems, concerns about uncertainty are shifted from the environment to the control system itself. Behaviors can sometimes have conflicting goals,

and reactive systems often behave in a near-sighted manner without consideration for long-term consequences. Also, behavior-based systems are sometimes characterized by "emergent" behavior (flocking is a well-known example) that is often unexpected and at times undesirable. This phenomenon is of particular interest in artificial intelligence research and attempts to understand biological systems, but can lead to underperformance or failure when the emergent behavior is undesired.

In hybrid control, action is transformed onto a representation of the environment, allowing application of well-developed and well-understood tools in optimization and control for proficient navigation. This makes the system easier to analyze and improve under nominal conditions. Nominal conditions imply that the internal model of the environment and the environment itself are sufficiently congruent. Deliberative systems can fail, sometimes catastrophically, when this assumption is violated.

Predictability also has direct influence on the lifespan of old measurements. For low predictability environments, keeping old data does not provide much advantage due to the diminishing correlation of these measurements with an environment model. For high predictability environments, using data backward in time for optimal estimation provides advantages analogous to searching forward in time for the solution to the optimal control problem.

Development

There are some advantages to behavior-based approaches to robotic navigation. Since individual behaviors generally connect sensory input to actuator controls, each behavior can be tested end-to-end without requiring specific performance from other behaviors. This allows easy parallel development that is difficult to achieve on a real vehicle in a hybrid approach unless many essential components (mapping, planning, following) are all working together. For deliberative systems, this issue can be mitigated through the use of simulated component operation in offline testing.

Behavior-based systems are also generally more robust to component failure, as operation can often continue even if some of the behaviors fail. This is a result of the generally parallel architecture representative of behavior-based systems as opposed to the generally serial architecture of hybrid systems.

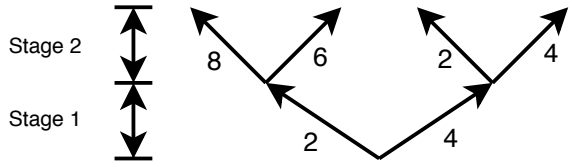

Figure 2.14. Illustrative example for comparison of architectures. System starts at bottom and chooses either left or right for each of two stages. Numbers illustrate cost of traversal.

Biological Inspiration

Another tangential but compelling advantage for behavior-based robotics in general lies in the belief that biological systems are fundamentally best understood as behavior-based and not primarily deliberative. Study and emulation of sufficiently complex behavior-based natural and man-made systems is likely to provide important insights into human cognition and animal behavior.

Optimality

A result in deliberative robotics and optimal control is that extending the planning horizon further in the optimization leads to more globally optimal solutions. The choice of horizon, then, is important for optimality and for ensuring that sequences of plans or actions are temporally consistent with each other and able to conform to the spatio-temporal dynamic constraints of the system. This is the motivation for the use of terminal cost in receding horizon control—to help ensure that the sequence of actions approximates the globally optimal solution.

Illustrative Example: Optimization Horizon

The simplified example of fig. 2.14 solidifies the planning horizon differences between deliberative and behavior-based architectures. This sample system has two basic choices (between left and right) to make before reaching a goal, and each decision has an associated cost as indicated. Two approaches are available to solve the problem of finding the best path to the goal:

1. A deliberative system would perform a search of all possible options and find the globally optimal solution – namely, to go right and then left. The horizon in this case is two stages, so the search space is completely covered in the deliberative approach.

2. A one-stage (reactive) behavior-based system with the two-behavior set of "go left" and "go right" ($\{L, R\}$) will choose the appropriate immediate behavior. Such a behavior-based

system will choose the suboptimal solution to go left and then right.

The second approach demonstrates that behavior-based control, in general, can be locally but not globally optimal. Note that a two-stage behavior-based system with the behavior set LL (go left twice), LR (left then right), RL (right then left) and RR (right twice) in this simplified example is equivalent to the deliberative system, because the behavior set spans the entire decision tree for this example problem. There are also standard ways to prune the decision tree to arrive at a globally optimal solution with less computational burden.

Other considerations guide a complete analysis of behavior-based or deliberative control. One is the consideration that the space of actions for a system can in general be high- or infinite-dimensional, and control directions can be continuous rather than discrete (e.g., $[-1, 1]$ instead of $\{L, R\}$). In this case, regardless of control strategy, it is necessary for computational tractability to reduce the dimensionality of the control space. Additionally, a standard strategy for behavior-based systems is to discretize continuous behaviors, while deliberative systems can perform optimizations on continuous-variable problems.

2.7 Methodology for Design and Evaluation of Robot Control Architectures

As discussed above, considerations of vehicle agility and environment predictability have a great influence on the choice of robot control architectures. Notionally, vehicle agility is a measure that refers to the degree to which a vehicle can avoid a pop-up obstacle that appears at some fixed distance away from the vehicle. Agility is therefore related to issues of vehicle speed and inertia, stability, and dynamic constraints.

Predictability, on the other hand, is a measure of both the environment and of the vehicle's sensing capabilities, and refers to the vehicle's ability to completely and accurately model the environment. Predictability, then, will depend on environment complexity—including whether dynamic elements exist in the environment and, if so, what their dynamics are—and on the sensing and perception capabilities of the vehicle.

Fig. 2.15 illustrates regimes for autonomous navigation based on predictability and dynamic feasibility (equivalently called agility). The inverse of these are indicated on the vertical and horizontal axes, respectively. While these are continuous measures, the figure is divided into four quadrants for illustrative purposes. Quadrant III represents the regime of much of "traditional" artificial intelligence research which dealt with slow-moving kinematic vehicles in highly

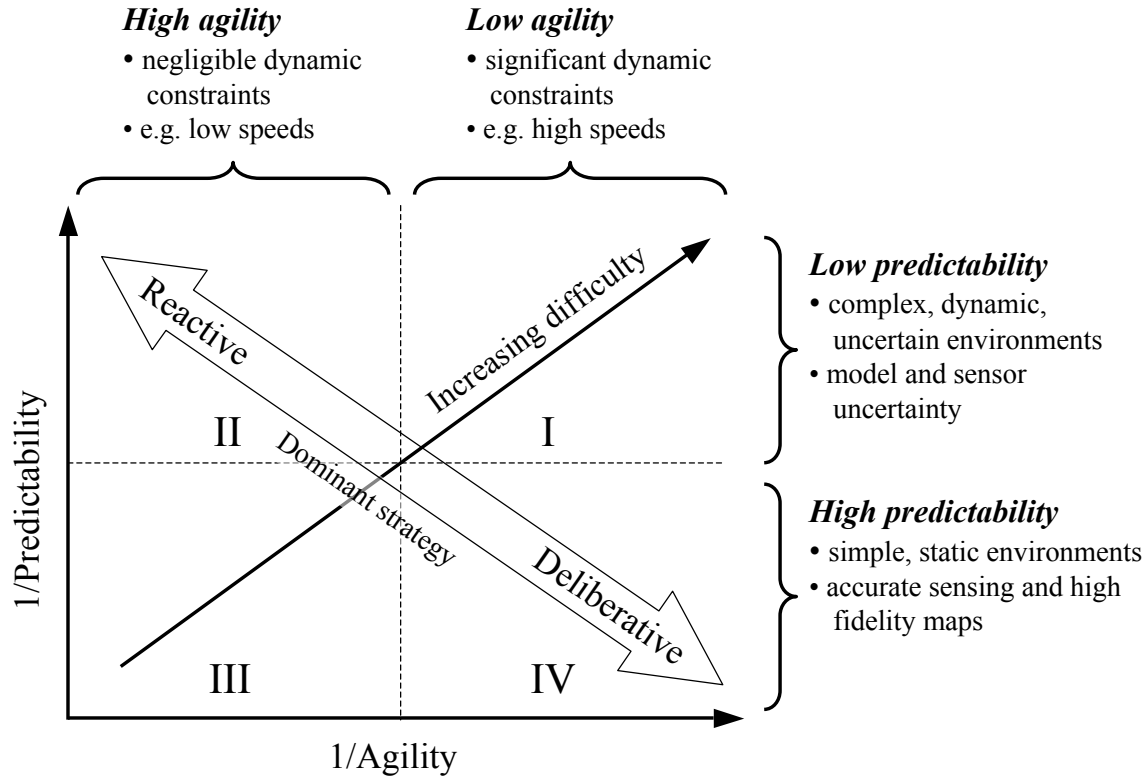

Figure 2.15. Autonomous navigation regimes based on agility and predictability. Hypothesized tendencies for dominance of more reactive and more deliberative strategies are shown.

predictable ("block world") environments. Quadrant II represents similar vehicles in less predictable environments, and quadrants IV and I represent the extension of operation in regimes where dynamic constraints play an important role in limiting the vehicle's actions (e.g., in high-speed operation where inertial effects are important). The combination of operating in environments with low predictability and in regimes of low agility represents the most difficult category of robot navigation.

The placement of the robot control spectrum on fig. 2.15 represents a *hypothesis* on the merits of different control architectures in the different operating conditions represented in the figure. Namely, the hypothesis is that more reactive control strategies will generally perform better for high agility vehicles in low predictability environments, and that more deliberative control strategies will perform better for low agility vehicles in highly predictable environments.

While, fig. 2.15 represents a qualitative guideline for design of robot architectures, it can be used to guide a methodology for designing, synthesizing and evaluating robot control architectures in order to develop concrete principles for robot control architecture design.

This chapter concludes with a description of a proposed methodology for robot control design. This methodology connects overall system performance with intermediate metrics such as agility, predictability, and position on the robot control spectrum. It also addresses the need to connect these intermediate metrics to specific design variables. The steps of this methodology are

1. Design a flexible robot control architecture that combines reactive and deliberative elements.

2. Identify parameters (such as T_s, T_p, and dependence on explicit models) that will explore the balance between these elements.

3. Evaluate intermediate metrics such as agility and predictability against particular vehicle tasks, using specific design parameters as variables (e.g. reaction time, vehicle speed, dynamic constraints, kinematic constraints, quality of state estimation, and sensor coverage parameters).

4. Through combination of simulation and experiments, measure overall mission performance in a variety of relevant operating conditions and mission tasks.

The process of applying the methodology above to many carefully designed experiments will provide data that can be used to develop formal, structured principles and methods for robot control design. Thorough investigation and development of these principles will prove valuable for current and future designers of robot control systems.

Chapter 3

Environment Modeling and Prediction for Autonomous Desert Navigation

Of central importance in most robotics applications is the ability to understand and effectively interact with the environment in which the robot is embedded. For a large class of applications, including all of deliberative robotics and some behavior-based approaches, this means constructing and maintaining a representation of the environment for use as the robot accomplishes its task(s). This is accomplished through the combination of sensor measurements from internal (proprioceptive) sensors, such as odometers, gyroscopes, accelerometers, and battery monitors, and from external (exteroceptive) sensors such as GPS, range sensors, machine vision, and tactile sensors. The manner in which these sensor measurements are combined is the subject of a large body of robotics research.

The choice of sensors and the choice of environment representation are important considerations in designing robotic systems that are able to efficiently and reliably accomplish a given task or set of tasks. The appropriate choice of sensor suite depends on the task. Following a white line on a table or floor may only require a light-sensitive sensor, while a factory assembly robot may require a tactile sensor, a camera, and position sensors on its arms. Similarly, different types of tasks will require different types of environment representations. The line-following robot may only need to know the lateral position of the white line with respect to the robot centerline, while the factory robot may need to know the six degree-of-freedom positions and orientations of each of the pieces it is assigned to assemble.

This chapter considers the task of building maps of the environment for the purposes of navigation through the environment. This broadly stated problem can be further clarified by limiting the class of environments considered for navigation. A large body of robotic navigation

literature is devoted to navigation in indoor environments, which are typically characterized by networks of straight, constant-width hallways separated by doorways. The comparison of such indoor environments with outdoor environments is often made in terms of environment complexity and structure.

The *degree of structure* of an environment, as it pertains to robotic navigation, refers to the proportion and complexity of distinctly recognizable features in the environment. For example, the simplest and most structured environment might consist of an infinite plane on which a robot navigates. The same environment with cylindrical obstacles could be characterized by the position $(x_i,\ y_i)$ and radius R_i of each of the obstacles. The ability to accurately represent such an environment with a low-dimensional state vector indicates a highly structured environment. An environment with more of such obstacles would be considered more cluttered but equally structured.

Indoor environments for robot navigation are often associated with the term *structured*; the straight walls and doorways form the structure of distinct recognizable features in the environment. Outdoor environments are often loosely called "unstructured," meaning that they have a low degree of structure. In fact, some outdoor environments are quite structured. City downtowns are composed of networks of roads lined on each side with tall buildings, analogous to the hallways and walls of indoor environments. Other outdoor environments will have varying degrees of structure. Areas affected by a major natural disaster are particularly unstructured, for example, as they may consist of collapsed buildings and unorganized piles of debris.

The type of map representation desired is highly dependent on the requirements of the specific application for which it is intended, including the degree of structure of the environment. There have been several major approaches to the modeling component for autonomous navigation in both indoor and outdoor applications. These can be differentiated from each other by the manner in which the environment is chosen to be represented, in other words, the type of map that is used.

A large body of robotic literature represents the environment as a collection of distinct objects or landmarks in a *landmark map*, largely owing to the efficacy of this representation in many solutions to the problem of Simultaneous Localization and Mapping (SLAM; see [51] for a thorough survey, especially as applied to indoor applications). SLAM is particularly useful in applications where accurate localization estimates are not independently available.

A second widely popular type of environment representation for mobile robotics is the *occupancy grid*, which is a representation of the probability that each cell in the grid is occupied.

Formal Bayesian filtering methods have been well developed for this type of map, and it is well suited for navigation through structured terrain such as indoor environments. Extensions of the occupancy grid are closely related to the idea of evidence grids pioneered by Martin and Moravec [33], and these ideas have been extended to three dimensions in the form of *voxel maps*, which can provide much more accurate representations of environments at the cost of greater memory usage.

Yet more sophisticated representations of the environment are also used for extremely accurate representations of complicated 3D geometry; these are popular in the computer graphics community for detailed modeling. An example of such sophisticated representations is the work done in terrain reconstruction via radial basis functions. These approaches, while able to deliver unprecedented accuracy, do so at a computational expense that makes them impractical for real-time applications.

For high-speed navigation, maintenance of complicated three-dimensional environment models is both unnecessary and undesirable, as one of the corequisite goals must be to maintain, process, and evaluate the map with the minimum latency possible. On the other hand, the effectiveness of strictly 2D maps is limited for navigation in unstructured terrain, since vital information about the surface geometry can be lost in these representations.

The modeling framework used here, the digital elevation map (DEM), represents a suitable middle ground between these approaches. In a DEM, the terrain is represented by a Cartesian grid, where each cell in the grid is assigned a height estimate for that cell. This is a compact representation that is amenable to computationally fast implementation in terms of storage, access, and evaluation. The digital elevation map is a so-called 2.5D model, since the terrain is represented as a (generally non-smooth) surface in 3D space. This representation is unable to accurately model features like tunnels and overhangs where the height of a given cell is multiply defined. Vertical surfaces present another challenge for terrain modeling with DEMs; these are commonly represented by either the average or maximum height within a cell.

This work focuses on environment modeling of outdoor, non-urban environments such as desert terrain. The primary interest is in developing applications generally applicable to unstructured terrain, where data association with distinct landmarks is not a requirement for mapping algorithms. However, even in the relatively unstructured context of desert navigation, detection and estimation of road features can provide a significant advantage in solving the navigation problem.

The remainder of this chapter describes work done in both digital elevation mapping and

road detection and tracking for the purposes of autonomous navigation. It is organized as follows: section 3.1 provides an overview of some stochastic tools available for application to this problem. section 3.2 presents results for digital elevation mapping while taking into account sensor uncertainties. section 3.3 describes a method for detection and tracking of desert roads as an aid for autonomous navigation.

3.1 Stochastic Methods for Robotic Mapping

Real-world sensor measurements are inherently imperfect; they are noisy and sometimes biased and sensitive to environmental conditions such as temperature, humidity, and atmospheric effects. Sensitivity to lighting conditions is one of the main reasons that real-world application of machine vision techniques is such a difficult research challenge. Because of this inherent limitation in measurement processes, it is impossible to provide a perfectly accurate estimate of the environment. Rather, our goal in environment estimation is to provide the best estimate possible given all of the measurements collected. Accuracy is typically balanced against computational complexity and efficiency in approaching this problem.

3.1.1 Bayesian Methods

Finding the best environment estimate has a direct mathematical interpretation with a foundation in probability theory. Much of stochastic analysis is founded on the exposition of Bayes' Rule, which relates the conditional and prior probabilities of stochastic events according to the relation

$$P(A|B) = \frac{P(B|A)\ P(A)}{P(B)}.$$

Here, $P(A|B)$ is *a posteriori* (posterior) probability of A conditional on B being true, $P(B|A)$ is the posterior probability of B given A, and $P(X)$ is the prior probability of X, given no other information.

Bayes' rule provides the foundation in estimation theory for determining the probability density of a state conditional on a history of indirect measurements of the state. Let $A = X$ represent the state of the environment as a random variable and let $B = Z$ represent the collection of measurements taken of the environment. The posterior probability $P(Z|X)$ of the set of measurements Z given an environment description X is also called the *likelihood function*. The process of estimating the environment can be cast as finding the X that maximizes the

observed set of measurements, i.e.,

$$X^* = \sup_{X \in \mathcal{X}} P(Z|X). \tag{3.1}$$

The process and methods for solving eqn. (3.1) are collectively called *maximum likelihood* (ML) estimation because they find the state X^* that maximizes the likelihood function. Many different such methods exist and are applicable to environment estimation and mapping, some of which are described in the following sections.

3.1.2 Kalman Filtering Methods

Consider the linear discrete-time system

$$\begin{aligned} x_k &= A_k x_{k-1} + B_k u_{k-1} + w_{k-1} \\ z_k &= C_k x_k + v_k \end{aligned} \tag{3.2}$$

where $x \in \mathbb{R}^n$ is the state vector, $z \in \mathbb{R}^m$ is the measurement vector, and $u \in \mathbb{R}^p$ is the system input. The process noise w_k and measurement noise v_k are random variables assumed to have normal distributions with zero mean and covariances Q and R, respectively.

$$p(w) \sim N(0, Q)$$
$$p(v) \sim N(0, R)$$

We seek a linear unbiased estimator of the state vector x_k, given the system dynamics and random-valued measurements of eqns. (3.2). Defining the estimation error at time k as $e_k = x_k - \hat{x}_k$, the unbiased condition implies that the expected value of this quantity is zero,

$$E[\hat{x}_k - x_k] = 0.$$

The Kalman filter updates the state estimate $\hat{x}_k$ as a combination of the *a priori* state estimate $\hat{x}_k^-$, before incorporation of measurement z_k, and a weighted difference between the actual measurement z_k and the predicted measurement $C_k \hat{x}_k^-$, i.e.,

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - C_k \hat{x}_k^-). \tag{3.3}$$

The Kalman filter is the optimal linear unbiased estimator in the sense that the choice of

K_k minimizes the mean-square estimate error given by the trace of the estimate covariance,

$$P_k = E[(\hat{x}_k - x_k)(\hat{x}_k - x_k)^T].$$

The estimator is typically implemented in two stages, a prediction stage in which the state estimate is propagated from $\hat{x}_{k-1}$ to $\hat{x}_k^-$ according to the system model, and a correction stage in which the estimate is transformed from $\hat{x}_k^-$ to $\hat{x}_k$ based on the measurement z_k according to eqn. (3.3). The prediction stage equations are

$$\begin{aligned} \hat{x}_k^- &= A_k \hat{x}_{k-1}^- + B_k u_{k-1} \\ P_k^- &= A P_{k-1} A^T + Q \end{aligned}$$

and the correction equations are

$$\begin{aligned} K_k &= P_k^- C^T (C P_k^- C^T + R)^{-1} \\ \hat{x}_k &= \hat{x}_k^- + K_k (z_k - C_k \hat{x}_k^-) \\ P_k &= (I - K_k C) P_k^-. \end{aligned}$$

3.1.3 Particle Filtering

Rather than representing the probability distribution of a random variable using a parameterized model (the mean and covariance in the case of Gaussian distributions), particle filters represent and maintain such distributions with a discrete set of samples called "particles." Their distribution implicitly represents the continuous probability distribution at time k, $P(X_k, Z_k)$, of the state being estimated, but no continuous representation is required for the particle filter implementation. The evolution of the particle filter is computed recursively in three steps: prediction, likelihood weighting, and resampling. Particle filters represent a promising way to deal with unpredictability by avoiding reliance on specific parametric models for system state and measurement probability distributions.

3.1.4 Moving Horizon Estimation

Kalman filtering methods, such as those presented in section 3.1.2, take a recursive formulation because of the assumptions of a linear model and Gaussian process and measurement noise. Their application can lead to failure when these assumptions are violated (in a case of low predictability

of the modeled process). As predictability decreases, the appropriate general response is greater reliance on sensing and perception as opposed to internal models. Moving horizon estimation is a means to do this by optimizing over a fixed history of sensor measurements to a sensor model.

For a discrete-time system of the form

$$x_{k+1} = f_k(x_k, w_k)$$
$$y_k = h_k(x_k) + v_k,$$

moving horizon estimation is formulated as the sequential implementation of the following optimization problem (written here for the measurements up to time T):

$$\Phi_T^* = \min_{x_0, \{w_k\}_{k=0}^{T-1}} \left(\sum_{k=T-N}^{T-1} L_k(w_k, v_k) + \sum_{k=0}^{T-N-1} L_k(w_k, v_k) + \Gamma(x_0) \right)$$

The optimization here has contributions from three terms. The first represents contributions from the most recent N measurements, the second from all previous measurements, and the third is a contribution from *a priori* information about the system initial condition. The second and third terms are typically approximated by an *arrival cost*, which can be updated using standard extended Kalman filter methods.

3.2 Stochastic Digital Elevation Mapping

This section provides a new computationally inexpensive approach to perception and modeling of the environment that allows fusion of sensory range data of various types and fidelities while explicitly taking into account a complete description of uncertainty in the range measurements. This approach makes use of known sensor uncertainty models to create a single 2.5D digital elevation map whose accuracy is robust to sensor noise and spurious data. This approach is particularly suitable for real-time application in high speed and highly unstructured outdoor environments for which reasonably accurate and timely vehicle state estimates are available. Experimental results are presented in which LADAR range measurements and state estimates are combined according to this approach. Additionally, qualitative comparison to other classes of environment modeling is provided.

Figure 3.1. Bob, Team Caltech's entry in the 2004 DARPA Grand Challenge, provides the test data for the results presented. Two stereovision camera pairs and a Sick LMS-2100 laser rangefinder are shown mounted above the cab; another Sick unit is mounted on the bumper. A Kalman filter achieves state estimates from differential GPS and IMU input data.

3.2.1 Introduction

Autonomous navigation for mobile robots has much potential in civilian, commercial, military, and space applications, but has not yet hit its stride in terms of demonstrating robust, real-time, high-speed operation in unstructured terrain for which there is no *a priori* terrain information available. Success in such an endeavor (and, indeed, in deliberative autonomous navigation in general), requires four fundamental contributions: *localization* within the environment, *perception and modeling* of the environment, *motion planning* through the environment, and *execution* of planned motion.

The combined picture of these contributions has been treated in several contexts. Kelly and Stentz [22] have provided a thorough system-level overview of the requirements for navigation in rough terrain, as well as a dynamical-systems oriented approach for vehicle control in such a situation [23]. Lacroix et al. [28] have developed and demonstrated a comprehensive approach for navigating long distances in unknown environments, suitable for autonomous planetary exploration. Bellutta et al. [6] and Stentz et al. [49] demonstrated approaches to the terrain perception problem in particular for the Demo III XUV and PerceptOR programs, respectively.

This section considers all four of these fundamental components (localization, perception, planning and execution), with a particular focus on the *perception and modeling* component, as applied in high-speed navigation in unstructured outdoor environments. Specifically, the goal addressed is the efficient and robust estimation of unstructured terrain given noisy state estimates and noisy range data. Since this terrain estimate is needed for real-time motion planning, issues of latency and throughput must be balanced with consideration for accuracy.

Note that a natural division can be and has been made within the modeling component between *terrain estimation* of the geometric properties of the environment and *terrain classification* of the surface and material properties of objects or regions in the environment. While the terrain classification problem has received various treatment in terms of both color and LADAR classification ([6, 32]), this section is strictly concerned with the terrain estimation problem.

For a DEM map representation as described above, our problem statement reduces to the following: *Given a set of noisy range data and vehicle state estimates, estimate the terrain surface elevation of the environment to provide an effective and efficient means for autonomous navigation.*

Several additional problem parameters will drive our approach. We will assume that reasonably accurate but noisy state estimates (3D location, pitch, roll, and yaw) are available to us, and that we can coregister these state estimates with our range measurements through the calibration parameters of our sensors. We further assume that our sensors are properly calibrated in terms of both the mounting parameters and intrinsic parameters of the sensor. The violation of this assumption would result in misregistered maps; while there are methods to correct out calibration error, we leave the integration of these methods as future work. Finally, by virtue of adopting a digital elevation map approach, we are making the 2.5D world assumption; that is, we assume that for each (x, y) cell location there exists a unique elevation, and that the actual elevation can be mathematically described as a function, albeit not necessarily in closed form. Violation of this assumption is tolerable; in such cases it may be impossible to show convergence, but formal proof of convergence is not a primary goal in this problem formulation.

Approaches to 2.5D terrain surface estimation in outdoor applications typically fall within one of two bipolar categories in terms of their treatment of uncertainty. The first approach neglects explicit account of uncertainty by averaging over the multiple measurements that fall within a given cell and/or discarding outliers that fit poorly the other data associated with the corresponding terrain. The second approach typically constitutes a Bayesian formulation that constructs an expression for the probability of a 3D surface conditional on the collection of range measurements [56, 57]. This approach, while most appealing because of its mathematical rigor, suffers a few drawbacks for our application. Its formulation requires either an *a priori* model to which the measurements are compared, or a parametric model whose parameters can be optimally estimated, neither of which are practical or desirable for navigation through unknown terrain. Finally, most methods that can accommodate this problem are not computationally efficient enough to process data as quickly and with as much throughput (on the order of

thousands of measurements per second) as is required.

This work, then, represents the "middle ground" between these two approaches. It provides a method to take range sensor uncertainty into explicit account in terrain estimation that is practical for real-time implementation at high speeds and data rates. Closely related approaches, in this respect, include those of Zhang [59], who estimates local parameters to fit a stereovision point cloud in a given region to a plane; Kelly and Stentz [23], who use the concept of a "scatter matrix" to represent the local geometric uncertainty in a grid; and Montemerlo and Thrun's recent approach [41] to accommodate sensor spatial resolution dependency on range.

This section presents an approach for 2.5D terrain estimation which is amenable to sensor fusion, at the map level, of any number and variety of range sensors for which sensor models can be estimated. This approach makes explicit use of a sensor model and provides an efficient method of updating the grid cells for each range measurement. Compared to deterministic techniques that do not consider sensor uncertainty, the main advantages of this approach are the generation of more complete, accurate, and robust terrain estimates, along with a spatial measure of the *uncertainty* in the terrain estimate, which may guide new strategies for path planning or directing sensor attention. We exercise this new method on real field data of LADAR range measurements that are coregistered with vehicle state estimates obtained from a moving vehicle, and we provide these raw data for further research in terrain estimation and autonomous navigation.

The research is presented as follows. Section 3.2.2 provides a formulation of and motivation for our approach. Section 3.2.3 develops the mathematical preliminaries and derivation of each of the components of our approach. A description of the experiment is given in section 3.2.4 and results are provided in 3.2.5. Section 3.2.7 provides a summary of this work and description of current and future work.

3.2.2 Formulation of Approach

Traditional simultaneous localization and mapping solutions use both state and range measurements to update the map of the environment and the location of the vehicle in the same update. State measurements (e.g., odometry) are used to update the locations of landmarks, and conversely, range measurements are associated with landmarks and used to update the estimate of the state of the vehicle. Our approach severs the latter connection, and propagates noisy but reliable state estimates through the usual geometric transformations to get a 3D description of the uncertainty of each measurement that encompasses the noise in the state as well as the

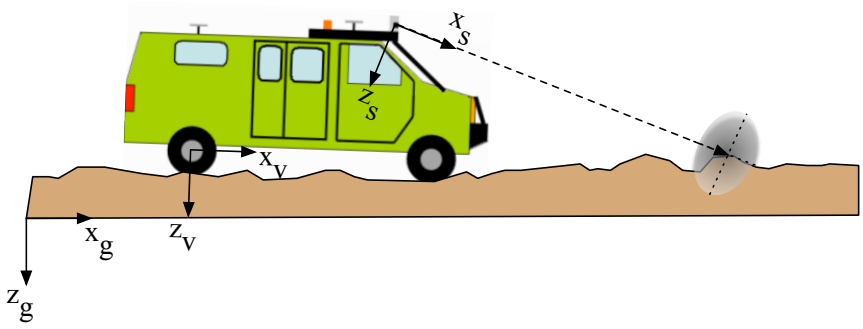

Figure 3.2. Two-dimensional schematic of a single uncertain range measurement taken from a sensor fixed to the vehicle. Coordinate axes corresponding to the global, vehicle, and sensor coordinate system are shown. Measurement uncertainty is depicted with a probability density function.

range measurement. In this way, the data association problem is avoided altogether, since range measurements are not used to update the state of the vehicle.

The approach to estimate the terrain profile is outlined below, and is depicted in fig. 3.2:

1. Construct a sensor uncertainty model from estimated raw variances in each of the state estimate variables (easting, northing, altitude, pitch, roll, yaw) and from variances associated with the range measurement (range, azimuth, elevation).

2. Take a range measurement for which the transformation between the sensor frame and the vehicle frame is known. This range measurement is transformed through the uncertainty model in step 1 to achieve a 3D description of a probability density function (pdf) for the given measurement.

3. Choose a region of cells around the mean of the pdf calculated in step 2, and calculate updates for each of these cells.

4. Update the cells chosen in step 3 according to an appropriate set of update equations.

3.2.3 Mathematical Preliminaries

This section provides detailed derivation of one such implementation of the approach outlined in section 3.2.2. Other implementations are possible and are likely to have their own advantages and disadvantages. Some remarks about the particular choice of implementation are provided in the subsections below. Subsection 3.2.3.1 describes an implementation of uncertainty model

and range measurement pdf computation (steps 1 and 2), and subsections 3.2.3.2 and 3.2.3.3 describes a cell update method implementation (steps 3 and 4). These subsections represent the method used to achieve the results presented in section 3.2.5.

3.2.3.1 Uncertainty Model

The following method is presented for computing the probability density function for a given range measurement. We refer to the pose of the vehicle at any given time as the collection of {easting, northing, altitude, pitch, roll, yaw}, denoted $\{x, y, z, \theta, \psi, \phi\}$, defined with respect to some inertial reference frame. We denote the variance in the estimate of each of these quantities as $\{\epsilon_x, \epsilon_y, \epsilon_z, \epsilon_\theta, \epsilon_\psi, \epsilon_\phi\}$. These raw state variances can be provided, for example, from the internal state of a Kalman filter state estimator or they can be estimated offline. Let the measurement in the sensor frame be defined by its range, azimuth, and elevation, denoted $\{\rho, \alpha, \beta\}$, and let the variances in these quantities be denoted by $\{\epsilon_\rho, \epsilon_\alpha, \epsilon_\beta\}$. We use in each of our coordinate frames the x-axis forward, y-axis right, and z-axis down convention, and define $\theta = \psi = \phi = 0$ when the vehicle is flat and pointed north. These coordinate systems are presented for clarity in fig. 3.2.

Our goal in this section is to derive a closed-form expression for the three-dimensional probability density function of a measurement, assuming each of the variance parameters defined above is given by Gaussian white noise. The location of a range measurement in sensor coordinates is given by

$$M_s = \begin{bmatrix} \rho \cos\alpha \cos\beta \\ \rho \cos\beta \sin\alpha \\ \rho \sin\beta \end{bmatrix}.$$

Applying the assumption of Gaussian noise (substituting $\rho \rightarrow \rho_0 + \epsilon_\rho$, $\alpha \rightarrow \alpha_0 + \epsilon_\alpha$, and $\beta \rightarrow \beta_0 + \epsilon_\beta$ and making a small angle approximation on the angle variances yields a description of the noisy measurement in the sensor frame. After some algebra, this noisy measurement can

be written as

$$M_s = M_{s_0} + M_s\epsilon = \begin{bmatrix} \rho_0 c_{\alpha_0} c_{\beta_0} \\ \rho_0 s_{\alpha_0} c_{\beta_0} \\ \rho_0 s_{\beta_0} \end{bmatrix} + \begin{bmatrix} c_{\alpha_0} c_{\beta_0} & -\rho_0 s_{\alpha_0} c_{\beta_0} & -\rho_0 c_{\alpha_0} s_{\beta_0} \\ \rho_0 c_{\alpha_0} c_{\beta_0} & s_{\alpha_0} c_{\beta_0} & -\rho_0 s_{\alpha_0} s_{\beta_0} \\ s_{\beta_0} & 0 & \rho_0 c_{\beta_0} \end{bmatrix} \begin{bmatrix} \epsilon_\rho \\ \epsilon_\alpha \\ \epsilon_\beta \end{bmatrix} \tag{3.4}$$

where the functions $\sin(\cdot)$ and $\cos(\cdot)$ are abbreviated as $s_{(\cdot)}$ and $c_{(\cdot)}$. Note that the expression above is the combination of a nominal term equal to the measurement value and an uncertainty term due to the variances in the sensor measurement. Based on (3.4), we can describe the probability density function in the sensor frame as centered around the nominal (measured) value with covariance given by $\Sigma = E(M_s \epsilon \epsilon^T M_s^T)$. Our goal, however, is to get the description of our measurement in the inertial frame in order to register it to our 2.5D map. This is done by transforming our measurement from the sensor frame to the vehicle frame, and then into the inertial frame. Noise in both of these transformations is accounted for by replacing the transformation variables with a nominal value plus additive noise, e.g., $x \rightarrow x_0 + \epsilon_x$ or $\phi \rightarrow \phi_0 + \epsilon_\phi$. Through this substitution, small angle linearization, and discarding of higher-order terms, first-order statistical expressions for the measurement in the vehicle and inertial frame can be obtained. Using the subscript syntax $(\cdot)_{vs}$ to indicate the transformation from the sensor to the vehicle, $(\cdot)_0$ to indicate the nominal measurement, and the *prefix* "$\epsilon\cdot$" to indicate the first-order statistic, the measurement in the vehicle frame can be written

$$M_v = R_{vs_0} M_{s_0} + R_{vs_0} \epsilon M_s + \epsilon R_{vs} M_{s_0} + T_{s_0} + \epsilon T_s, \tag{3.5}$$

where R and T represent the rotation and translation of the associated transformation, respectively. Following the same procedure for transformation of the measurement into the inertial (global) frame yields a similar expression for the measurement in terms of (3.5),

$$M_g = R_{gv_0} M_v + \epsilon R_{gv} (R_{vs_0} M_{s_0} + T_{s_0}) + T_{v_0} + \epsilon T_v. \tag{3.6}$$

This measurement description in the global frame is the sum of the nominal sensor measurement (transformed to the global frame) plus a first-order term that depends upon the combined variances from the sensor measurement itself as well as the variances that appear in both of

the coordinate transformations. The elaboration of the algebra takes several pages and was not performed by hand, but it does result in a closed-form expression consisting of additions and multiplications that lends itself to very fast computation (on the order of microseconds for a single measurement).

Through these noise propagation equations, each measurement can be described as a first-order probability density function with mean and covariance as derived from (3.6). The ellipsoid in fig. 3.2 represents a fixed-probability error surface as computed from one such measurement description in the global frame.

3.2.3.2 Measurement Discretization

The explicit, computable expression for the measurement uncertainty resulting from the analysis of section 3.2.3.1 will, in general, extend over multiple cells in a gridded map for a single measurement. It is therefore necessary to devise an appropriate way to determine which cells should be updated, and how they should be updated, for each measurement.

Subsection 3.2.3.3 describes the method we use to update a single cell given normally distributed cell input measurements z_k that are described by their mean z_m and variance σ_z. This section describes the way in which a single 3D measurement and associated uncertainty model is converted into a number of cell input measurements.

Simplifying assumptions were made about the equations presented in section 3.2.3.1 so that the uncertainty of a single measurement can be represented by first order statistics as a multivariate Gaussian. The shape and orientation of the Gaussian for each measurement depends on the direction of the sensor measurement as well as the variance in sensor range measurement and variances in vehicle position and orientation. The measurement is parameterized by a center $\mu \in \mathbb{R}^3$ and a covariance $\Sigma \in \mathbb{R}^{3\times 3}$ and is represented by the equation

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{3/2}\sqrt{\det \Sigma}} \exp\left[-\frac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1}(\mathbf{x}-\mu)\right]$$

where $p(\mathbf{x}) = p(x, y, z)$ is the probability that the measurement actually came from a surface at $\mathbf{x}$.

With an elevation grid representation of the environment, each cell C_{ij} in the grid can be reasonably assigned the height distribution

$$p_{ij}(z) = \iint_{C_{ij}} p(x, y, z)\ dy\ dx. \tag{3.7}$$

This function cannot be considered a probability density function, however, because the total integral of $p_{ij}(z)$ is not equal to 1. A normalization of this function is necessary to use it as a pdf for input to the Kalman filter update equations of section 3.2.3.3. One option for normalization would be a scaling of the function by α_{ij} so that

$$\alpha_{ij} \int_{-\infty}^{\infty} p_{ij}(z)\ dz = 1.$$

The implementation of this method, however, has the significant drawback that the cells far from the center of a measurement will have variances that are similar to those of the cells near the center of the measurement. For cells far away from the measurement, the fact that $p_{ij}(z)$ is much smaller should correspond to a high variance associated with the measurement. A new method is used to achieve this result; the mean μ_{ij} of $p_{ij}(z)$ is calculated and then $p_{ij}(z)$ is normalized by setting the standard deviation of the normalized Gaussian pdf to

$$\sigma_{ij} = \frac{1}{p_{ij}(\mu)\sqrt{2\pi}}.$$

Cells close to the center of the measurement, therefore, will have higher value of $p_{ij}(\mu)$ and a lower variance. Cells far from the center of the measurement will have lower $p_{ij}(\mu)$ and hence a higher variance.

In a practical implementation, it is necessary to decide which cells are to be updated for any given measurement. There are options for how to specify this. One method for doing so for a Gaussian $p(\mathbf{x})$ is to update any cell whose center lies within a χ-confidence ellipse. Another is to update those cells for which the peak probability is greater than some threshold. A third is to choose to update the cells whose centers lie within a specified geometric shape (rectangle, circle, ellipse). In the implementation described in section 3.2.5, this last method was used for the sake of ease of implementation. Also for practical considerations, the integral in eqn. (3.7) was approximated by

$$p_{ij}(z) \approx p(x_i, y_j, z)\Delta_x\Delta_y$$

where the center of cell C_{ij} is (x_i, y_j) and the resolution of the grid is specified by Δ_x and Δ_y.

3.2.3.3 Cell Update Equations

The update equation that governs each cell is a Kalman filter whose state is equal to the scalar height estimate for the cell. Since there are no independent dynamics associated with the height

of a cell, the state propagation equations are simply identity, and the height estimate is purely a result of updates from measurements whose (x, y) coordinates land near that cell. In this manner, cells in the map receive a flurry of updates when a series of sensor measurements pass over it, but are otherwise unchanged. This results in an efficient means of updating cells during high-speed navigation.

Note that this approach executes N^2 one-dimensional Kalman filters, which is tantamount to the assumption that the height of each cell is statistically independent of any other cell. This is a drawback of the algorithm in the case where one can make some *a priori* assumption about the properties of the terrain, such as how smooth it is. Rather than incorporating any such information, adjacent cells are coupled only in that a single range measurement generally corresponds to filter updates for a region of cells.

An abbreviated version of the Kalman filter measurement update equations (see for example [55]) is

$$\begin{aligned} K_k &= P_{k-1}H^T(HP_{k-1}H^T + R)^{-1} \\ \hat{x}_k &= \hat{x}_{k-1} + K_k(z_k - H\hat{x}_{k-1}) \\ P_k &= (I - K_kH)P_{k-1}. \end{aligned}$$

In this formulation, the state vector is actually the scalar height of a cell, a measurement that one can readily obtain from the geometry and uncertainty of the range measurement. Accordingly, one can set the H term equal to unity. The error covariance P is equal to the variance in the height estimate, σ_h^2, and R is equal to the variance of the elevation measurement, σ_m^2. It can be shown from these equations that the update equations reduce to

$$\begin{aligned} \hat{x}_k &= \frac{\sigma_h^2 z_m + \sigma_z^2 x}{\sigma_h^2 + \sigma_z^2} \\ P_k &= \frac{\sigma_h^2 \sigma_z^2}{\sigma_h^2 + \sigma_z^2}. \end{aligned}$$

These are the equations used to update the height and variance in height for the cells that are chosen to be updated according to the methods of section 3.2.3.2.

3.2.4 Description of Experiment

Experiments were run using Bob (fig. 3.1), an instrumented and actuated sport-utility vehicle that served as Team Caltech's entry in the 2004 DARPA Grand Challenge. Bob is equipped with

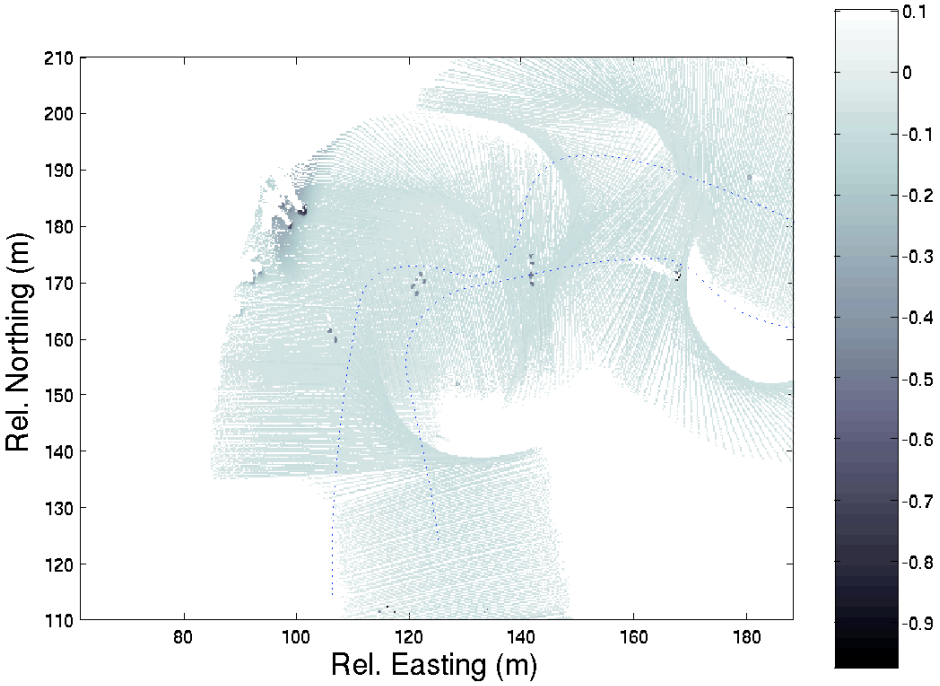

Figure 3.3. Elevation map resulting from the naïve approach of replacing cell data with the height of the new measurement if the new measurement is higher than the old.

two pair of stereovision cameras and two 2D scanning laser rangefinders mounted horizontally. For tests of these new algorithms, simultaneous state and range data (from the cab-mounted LADAR unit only) were taken at approximately 5 Hz, and a manually controlled path was driven through a handcrafted course.

The course consisted of a flat dry lakebed with hand-placed obstacles at measured locations on the course. The obstacles were cylinders approximately 1 m tall and 1 m radius, ice chests of approximate dimension (1 × 0.5 × 0.5) m, and a vertical plywood sheet of height and width 1 m × 1 m. Although the algorithms presented here are intended for high-speed operations, the experiments were run at moderate speeds in order to use the data to demonstrate the effectiveness of the algorithm under controlled conditions.

While demonstrated at moderate speeds, the algorithms developed based on sections 3.2.2 and 3.2.3 are able to run at very fast rates (processing of the LADAR range measurements – 201 per scan at 5 Hz scan rate – was demonstrated in real-time with a latency of a few tens of milliseconds on a 1.5 GHz Pentium 4 processor).

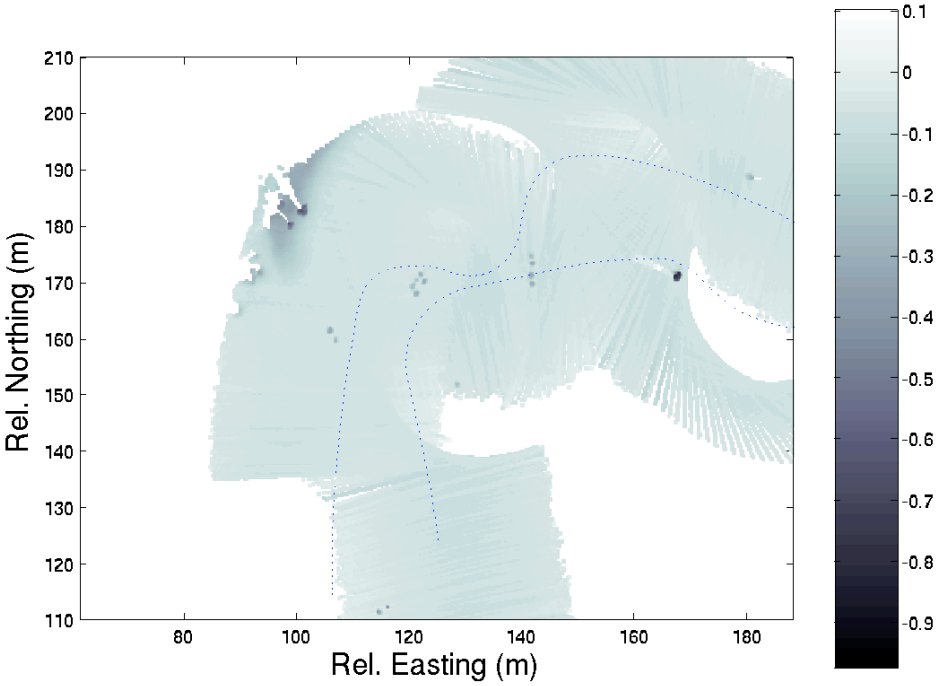

Figure 3.4. Elevation map resulting from the new approach, which updates a small region of cells for each measurement according to our update method (for comparison to fig. 3.3).

3.2.5 Experimental Results

The dotted lines in Figs. 3.3 and 3.4 represent the path that was taken by Bob during one data run. The maps created in each of these figures are both 0.25 m $\times$ 0.25 m resolution and represent the application of two different map estimation approaches. The first, naïve method processed the data while neglecting considerations of uncertainty in the measurement. It simply replaced the data in the map at the appropriate cell with the maximum of the measurement height and the current height in the cell, if any. The resultant map using this method is shown in fig. 3.3. The second approach (fig. 3.4) represents the result of the map update method presented in section 3.2.2, with the cell containing the measurement mean and the eight cells immediately surrounding it receiving updates for each measurement.

The resulting terrain maps in Figs. 3.3 and 3.4 are coded with intensity proportional to height. Cells that are coded white represent no data assignments. Several qualitative comparisons can be made from these results. First, most strikingly, the maps provided by this approach result in much more complete terrain maps than the simpler approach, meaning that many fewer cells remain assigned with no data. This is a direct result of updating a local region of cells for a given measurement. Second, this new approach tends to pick out obstacles more clearly than

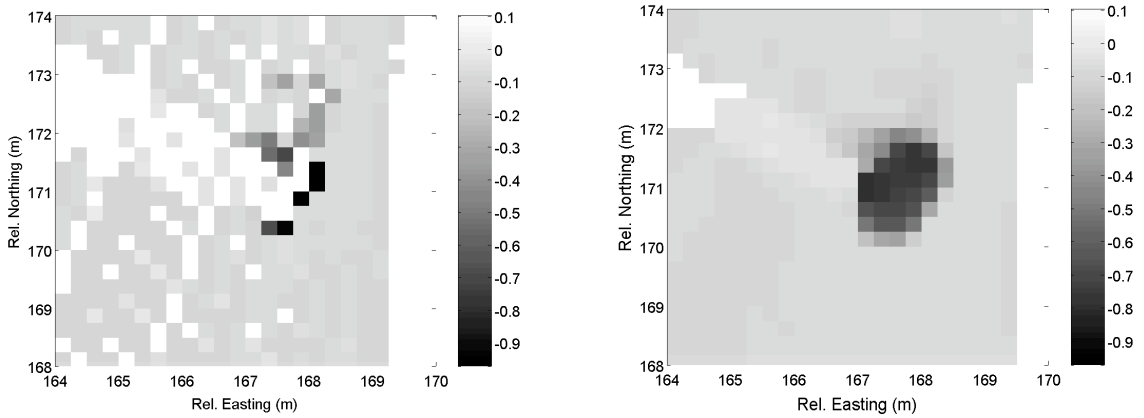

Figure 3.5. A zoomed portion of figs. 3.3 and 3.4 (on the left and right, respectively) near relative location (170, 170) m. Note that the sensor fusion method fills in the obstacle more fully and reduces the effect of spurious measurements.

the naïve approach, as seen for example in the zoomed image in fig. 3.5. Third, the fused data method of creating the terrain maps has a smoothing effect on the terrain estimate. This is an expected effect of the algorithm.

Also note that for both methods of terrain estimation, registration errors are apparent for those obstacles that are passed more than once in the course of the experiment (those in the figures with less than about 150 m relative easting). This effect is one disadvantage of not attempting to correct the state estimate with the range measurements, as is done in SLAM approaches. Application-specific considerations must be made when deciding whether to pursue this approach, which is a trade-off between potential accuracy and computation expense.

3.2.6 Multiple Sensor Considerations

The methodology above can be applied to multiple sensors, but the Gaussian noise assumption will break down if there is miscalibration between sensors. For example, Alice's long range sensor had a mounting pitch of -0.0539 radians ($z = -2.480$ m) and our longest medium range sensor had a pitch of -0.0803 radians ($z = -2.432$ m), relative to the flat ground. These correspond to horizontal range on flat ground of $\approx$46 m and $\approx$30 m, respectively.

Consider the sensor of fig. 3.2 in two dimensions (up and forward). Also, let

- z_s = the height of the sensor in global coordinates,
- $(x,\ z)$ = the actual downrange and elevation of the measurement,
- $(\hat{x},\ \hat{z})$ = the estimated downrange and elevation of the measurement,

- ρ = the range measurement from the sensor, and
- $(\phi,\ \hat{\phi})$ = the actual and estimated sensor pitch.

The measurement model for this system is given by

$$z(\phi) = z_s + \rho \sin\phi$$
$$x(\phi) = \rho \cos\phi$$

and by setting $\hat{\phi} = \phi + \Delta\phi$, the error model is

$$z_e = z - \hat{z} = \rho(\sin\phi - \sin(\phi + \Delta\phi))$$
$$x_e = x - \hat{x} = \rho(\cos\phi - \cos(\phi + \Delta\phi))$$

and is equivalent to

$$z_e = -\rho\ \cos\phi\ \Delta\phi \tag{3.8}$$
$$x_e = \rho\ \sin\phi\ \Delta\phi. \tag{3.9}$$

Eqn. (3.8) indicates that the sensitivity in the elevation to pitch calibration error is equal to the downrange position; i.e., the further out the sensor points on flat ground, the more sensitive it is to pitch miscalibration.

3.2.7 Summary and Extensions

The main contribution of this section is to provide an approach to real-time 2.5D terrain estimation that explicitly uses sensor models in its formulation and includes the effect of noisy but reliable state estimates. One such implementation of this approach was presented along with a qualitative analysis of its performance. It is expected that other implementations of this approach will be able to show marked improvement in terrain estimation for high-speed navigation through unknown and unstructured terrains.

This work is intended as a baseline for research into computationally inexpensive methods of real-time digital elevation mapping appropriate for high-speed navigation. It serves as a springboard to more mathematically formal approaches that are still amenable to processing at high speeds and high data rates. Future work includes development of such approaches and providing rigorous quantitative analysis and comparison of different techniques.

Specific potential areas for future work include the development of data-driven sensor uncertainty models, as they may show marked improvement over the Gaussian assumption of the measurement model. We also intend to extend this general method to sensor fusion at the map level of multiple types of sensors, including combined stereovision and LADAR.

3.3 Road Detection and Modeling via Kalman Filtering

This section applies some previously studied extended Kalman filter techniques for planar road geometry estimation to the domain of autonomous navigation of off-highway vehicles. In this work, a clothoid model of the road geometry is constructed and estimated recursively based on road features extracted from single-axis LADAR range measurements. We present a method for feature extraction of the road centerline in the image plane and describe its application to recursive estimation of the road geometry. We analyze the performance of our method against simulated motion of varied road geometries and against closed-loop detection, tracking, and following of desert roads. Our method accomodates full six DOF motion of the navigating vehicle, constructs consistent estimates of the road geometry with respect to a fixed global reference frame, and requires an estimate of the sensor pose for each range measurement.

3.3.1 Introduction

Estimation of road geometry is an important task for a variety of automotive applications in intelligent transportation systems because it enables prediction and evaluation of the future path of the vehicle. This information is particularly pertinent to driver assistance technologies that involve detection and response to other vehicles or hazards on its path, such as adaptive cruise control and collision warning systems.

Since road curvature parameters change as a function of distance along the road, they can be viewed as the state and output of a time-varying process as the vehicle moves along the road. Recursive estimation of these parameters using Kalman filtering techniques has become a standard for road (and many other types of) estimation, and this approach has been applied in recent years for navigation on well-structured paved roads with relatively clear boundary markings (see for example [13, 25, 58]). Common assumptions for these environments include planar road geometry, negligible sensor pitch and roll, and that the vehicle is traveling along the center of the road or lane. In rural or underdeveloped areas, however, many of these assumptions can break down, and novel feature extraction algorithms are needed. Reference [11] outlines a

feature extraction method that finds the straight line segments in the Cartesian ground plane which can be applied to such situations; this work presents a complementary method for feature filtering and extraction.

This work applies recursive estimation schemes for road estimation to "off-highway" environments, where roads are typically not painted with boundary markings and in many cases are unpaved. Off-highway navigation presents a special challenge for road estimation due to the rough motion of the sensor and the lack of visual structure like that found in highways and improved roads. We therefore make no assumption that the pitch and roll of the vehicle are negligible, but rather require a full six DOF estimate of the sensor pose. In this way, we are able to associate inertial and range information to do road feature extraction in a global coordinate system. The only assumption that we make about the vehicle pose relative to the road is that features of the road lie within the sensor field of view. As in other work, we do assume planar geometry for the road, but this assumption could be lifted with extension of the ideas presented here.

While off-highway scenes have been studied recently using image processing and computer vision techniques, as in [44], we have chosen to begin this work using a single axis laser detection and ranging (LADAR) measurement system. Advantages of using LADAR include operability in unfavorable lighting conditions and the ability to use direct range measurements to represent road features in an inertially fixed reference system. Extensions within the framework we provide here will be able to accomodate LADAR and image-based sensing together, but these are outside the scope of this work.

Our work is motivated by the problem of reliable fully autonomous navigation of ground vehicles in unstructured environments. Solutions to this problem will see application in places where human operation of vehicles is typically too costly and/or too dangerous. This will most ostensibly be seen in military transport operations within the next ten to twenty years; we anticipate economically and technologically feasible further application to construction, agriculture, manufacturing, and mining activities in twenty to fifty years. After several generations of advancement in the reliability of autonomous navigation, the level of autonomy could be sufficient to provide blind and disabled individuals personal transportation solutions (with assistance from the individual for high-level navigation instruction).

While road-feature identification in static camera images has been studied for on- and off-highway environments, and recursive road estimation has been studied using both camera and LADAR sensing techniques in urban environments, this work represents some of the first work of

the authors' knowledge that applies recursive estimation techniques to road estimation explicitly for navigation of off-highway roads. The main contributions of this work are the application of recursive road estimation techniques to off-highway environments, estimation of complete road geometry in the global coordinate system, development of techniques to accommodate pitching and rolling of the vehicle during navigation, and performance evaluation of these techniques.

These contributions are presented as follows. Basic assumptions and the road model are given in sections 3.3.2 and 3.3.3. The measurement model and recursive estimation framework are presented in section 3.3.4. We present road-feature extraction techniques in section 3.3.5 and the results from simulation and autonomous operation in sections 3.3.6 and 3.3.7. A summary and brief look toward future work are presented in section 3.3.8.

3.3.2 Assumptions

This work presents a solution for estimating the road geometry as the vehicle travels along the road. We maintain the following fundamental assumptions throughout this section. We assume

1. that a forward-looking sensor is mounted on the vehicle – we assign to this sensor both a Cartesian coordinate system $\mathcal{S}$ and an image coordinate system $\mathcal{I}$,

2. that we are able to extract estimates of road-features in either of the sensor-assigned coordinate systems,

3. that we have, at any given time, an estimate of the full six DOF pose of the sensor with respect to some fixed inertial (global) coordinate system, which we will call $\mathcal{G}$,

4. that the roughness of the road is small relative to the roughness outside the boundaries of the road, and

5. that there are small heading changes in the road at the scale of the sensing horizon.

Assumption 3 can be satisfied with some combination of GPS, inertial sensing, and odometry. We have done this through related work by using GPS and IMU data as inputs to a Kalman filter, following the principles of [19]. With assumptions 2 and 3 and a range sensor, the road-feature estimates can all be represented in the fixed coordinate system $\mathcal{G}$. The primary use of the inertial sensing is to provide a way to estimate the road geometry with respect to a fixed coordinate system.

We will further assume that the vehicle is driven such that the road geometry is maintained in the sensor's field of view, but no further assumption is made regarding the relative position

or orientation between the vehicle and the road. The road is assumed to be planar, and the width of the road is assumed to be fixed.

The estimation framework used here is an implementation of the extended Kalman filter (EKF). An EKF is used because the speed of the vehicle is not assumed to be constant and because the measurement model is linearized about the small heading changes assumed (assumption 5). The state variables of the Kalman filter are the local curvature κ_0 and arc length rate of change of curvature κ_1. Local here indicates the position along the road centerline that is closest to the vehicle coordinate system (assumed coincident with the sensor coordinate system $\mathcal{S}$).

We enlist another coordinate system, $\mathcal{R}$, that is attached to the road with its x-axis always pointing tangent to the road, but able to be repositioned as the road estimate progresses. One can think of the road estimate construction as analogous to laying down model railroad tracks one after the other, with $\mathcal{R}$ positioned at the end of the last laid track segment. Forward road geometry estimates extend from this coordinate system origin and represent the current curvature estimates.

3.3.3 Road Model

We have chosen a piecewise clothoid to model the road centerline geometry. The centerline is represented as a planar, twice differentiable curve as a function of arc length, $\mathbf{r}(s) \in \mathbb{R}^2$ and is parameterized by curvature $\kappa(s)$. In the clothoid model, the curvature profile is assumed to be piecewise linear, i.e., each segment of the road in this model corresponds to a constant arc length rate of change of curvature. For segment i which covers the arc length interval $s \in [s_i, s_{i+1})$, the curvature is given as $\kappa_i(s - s_i) = \kappa_{0,i} + (s - s_i)\kappa_{1,i}$. The twice differentiable assumption on the curve guarantees curvature continuity.

For normal Ackermann-steered automobiles, this parameterization corresponds to continuous nominal motion of the steering wheel as the road is traveled; this is a common method in the design of roads and highways [54]. Fig. 3.6 depicts an example simulated road geometry, determined completely by the curvature rate profile and the initial position, initial orientation, and initial curvature of the road.

The orientation and position of the road centerline in $SE(2)$ can be easily recovered from

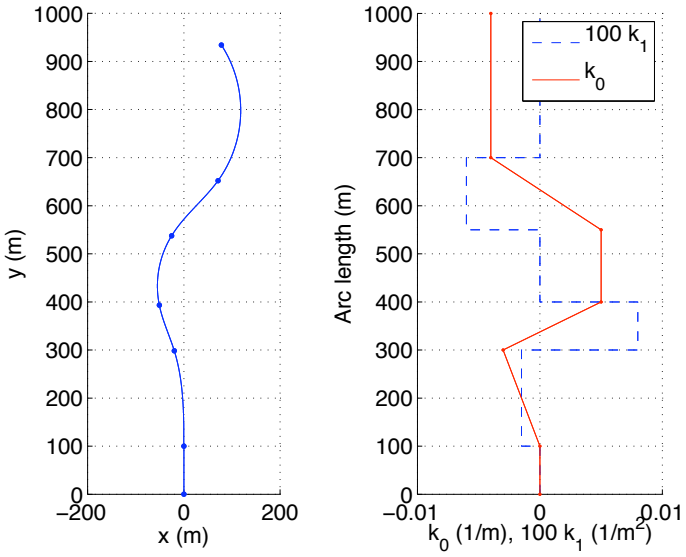

Figure 3.6. Simulated road geometry (left) and curvature as a function of arc length (right). In our coordinate conventions, negative curvature corresponds to a left-hand turn.

the curvature profile and are given as a function of arc length by

$$\begin{aligned} \theta(s) &= \theta_0 + \int_0^s \kappa(\tau)\, d\tau \\ x(s) &= x_0 + \int_0^s \cos\theta(\tau)\, d\tau \\ y(s) &= y_0 + \int_0^s \sin\theta(\tau)\, d\tau. \end{aligned} \tag{3.10}$$

Using the small angle approximation for the angle $\theta(\tau)$, the integrals in eqns. (3.10) can be evaluated in closed form to be

$$\begin{aligned} x(s) &\approx x_0 + s \\ y(s) &\approx y_0 + \theta_0 s + \tfrac{1}{2}\kappa_0 s^2 + \tfrac{1}{6}\kappa_1 s^3 \\ \theta(s) &= \theta_0 + \kappa_0 s + \tfrac{1}{2}\kappa_1 s^2 \\ \kappa(s) &= \kappa_0 + \kappa_1 s. \end{aligned} \tag{3.11}$$

By choosing the coordinate frame representation that is aligned with the road initial position and orientation, the initial angle θ_0 can be set to zero, and the validity of the small angle representation depends only on the distance along the curve and the curvature parameters.

Eqns. (3.11) are a spatial representation of the curve, and we can consider analytically the

derivatives of these variables with respect to arc length as one imagines a bead moving along the curve. These derivatives are

$$\begin{aligned} y'(s) &= \theta_0 + \kappa_0 s + \tfrac{1}{2}\kappa_1 s^2 \\ \theta'(s) &= \kappa_0 + \kappa_1 s \\ \kappa'(s) &= \kappa_1. \end{aligned} \tag{3.12}$$

The representation of eqns. (3.11) and eqns. (3.12) have powerful complementary spatial and temporal aspects that can be exploited in a novel application of Kalman filtering techniques. First, the spatial eqns. (3.11) can be used to determine the predicted location of features on the curve at some look-ahead distance $s = x_m$ where a measurement is taken. Second, eqns. (3.12) can be used as a dynamical basis for temporally evolving the representation of the road as a sensor moves along it and picks up new measurements. While these measurements can be unreliable due to noise or bad feature extraction, the model-based filtering techniques provide robustness to these issues, especially as a large number of measurements are taken and noise is effectively averaged out.

Using the small angle approximation for the angle $\theta(\tau)$ (justified by choosing the road coordinate aligned with the initial θ_0), eqns. (3.10) can be transformed into a spatio-temporal representation using the chain rule and assuming some time-dependent speed $v(t)$ along the curve. The resulting differential equations are

$$\dot{y}(s) = v(t) \left(\theta_0 + \kappa_0 s + \frac{1}{2}\kappa_1 s^2\right) \tag{3.13}$$

$$\dot{\theta}(s) = v(t) \ (\kappa_0 + \kappa_1 s) \tag{3.14}$$

$$\dot{\kappa}(s) = v(t) \ \kappa_1. \tag{3.15}$$

We use the representation of eqns. (3.13–3.15) to provide both the propagation and the measurement model for our Kalman filter design. A time integral of eqn. (3.13) at a fixed look-ahead distance s provides the measurement model with y (the lateral offset in the road coordinates) as the measurement variable, and propagating the road coordinate system forward with the vehicle provides the dynamic equations for tracking of road parameters.

The evolution of the curvature variables as the vehicle moves forward at speed $v(t)$ is modeled by

$$\begin{bmatrix} \dot{\theta}_0 \\ \dot{\kappa}_0 \end{bmatrix} = \begin{bmatrix} 0 & v(t) \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \theta_0 \\ \kappa_0 \end{bmatrix} + \begin{bmatrix} 0 \\ w(t) \end{bmatrix}, \tag{3.16}$$

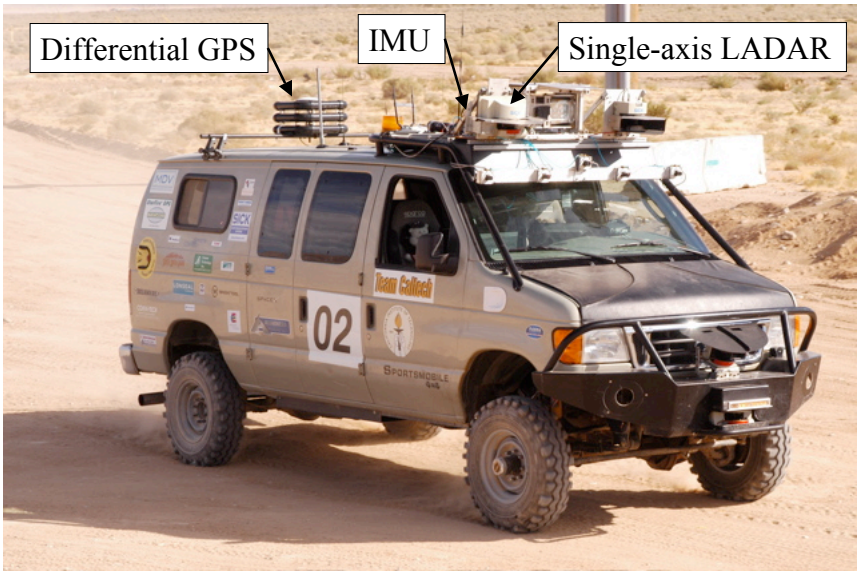

Figure 3.7. The test platform is a 2005 Ford E-350 Econoline van modified by Sportsmobile of Fresno, California. A roof-mounted LADAR and custom inertial navigation system (INS) provide the raw data used for navigation of desert roads.

where $\dot{(\cdot)} \triangleq \frac{d}{dt}(\cdot)$ and $w(t)$ represents a noise term that drives the evolution of κ_0. The noise signal $w(t)$ is assumed to be Gaussian and white with $p(w) = \mathcal{N}(0, Q)$. Eqn. (3.16) forms the process dynamics for our estimator, to be used in the propagation step.

Note that there is no input in this process model, but any *a priori* information about the road could be incorporated into the model here. Note also that we chose to use a state formulation with heading and curvature, rather than curvature and curvature rate, as this resulted in better performance in closed-loop testing. A comparative analysis of the different approaches is not within the scope of this work, but warrants attention.

In compact form, we re-express eqn. (3.16) as

$$\dot{\mathbf{x}} = A(t)\mathbf{x} + \mathbf{w}(t). \tag{3.17}$$

We approximate eqn. (3.17) with the first order difference equation

$$\mathbf{x}_k = A_k \mathbf{x}_{k-1} + \mathbf{w}_k \tag{3.18}$$

where $A_k \triangleq (A(k\Delta t)\Delta t + I)$.

3.3.4 Measurement Model

The underlying measurement model used follows the approach in [13], and relies on small heading changes between the road coordinate x-axis and the heading of the road at the so-called look-ahead location at which the road measurement is taken. With this small angle approximation, $\cos\theta_r \approx 1$ and $\sin\theta_r \approx \theta_r$, and the lateral location of the centerline in the road coordinate at look-ahead distance x_m is recovered from eqns. (3.10) as

$$y \approx \frac{1}{2}\theta_0 x_m + \kappa_0 x_m^2 + \frac{1}{6}\kappa_1 x_m^3.$$

By formulating with heading and curvature only, we can eliminate the κ_1 term, and therefore find the following as the measurement equation associated with our process model:

$$y(t) = \begin{bmatrix} x_m & \frac{1}{2}x_m^2 \end{bmatrix} \begin{bmatrix} \theta_0 \\ \kappa_0 \end{bmatrix} + \nu(t). \tag{3.19}$$

We use a discrete version of eqn. (3.19) for use with the discrete EKF,

$$y_k = C_k \mathbf{x}_k + \nu_k. \tag{3.20}$$

The signal ν_k represents our measurement noise, which we assume to be white and Gaussian with $p(\nu) = \mathcal{N}(0, R)$, where R is the measurement noise (co)variance. This measurement noise can be estimated from a statistical analysis of a sequence of recorded measurements.

The discrete process and measurement eqns. (3.18) and (3.20) were used in the design of the extended Kalman filter. Letting $\hat{x}_k$ denote the state estimate, and given an initial estimate of the process covariance P_0, the propagation equations are given by

$$\begin{aligned} \hat{x}_k^- &= A_k \hat{x}_{k-1} \\ P_k^- &= A_k P_{k-1} A_k^T + Q \end{aligned}$$

and the update equations are

$$\begin{aligned} K_k &= P_k^- C_k^T (C_k P_k^- C_k^T + R)^{-1} \\ \hat{x}_k &= \hat{x}_k^- + K_k (y_k - C_k \hat{x}_k^-) \\ P_k &= (I - K_k C_k) P_k^-. \end{aligned}$$

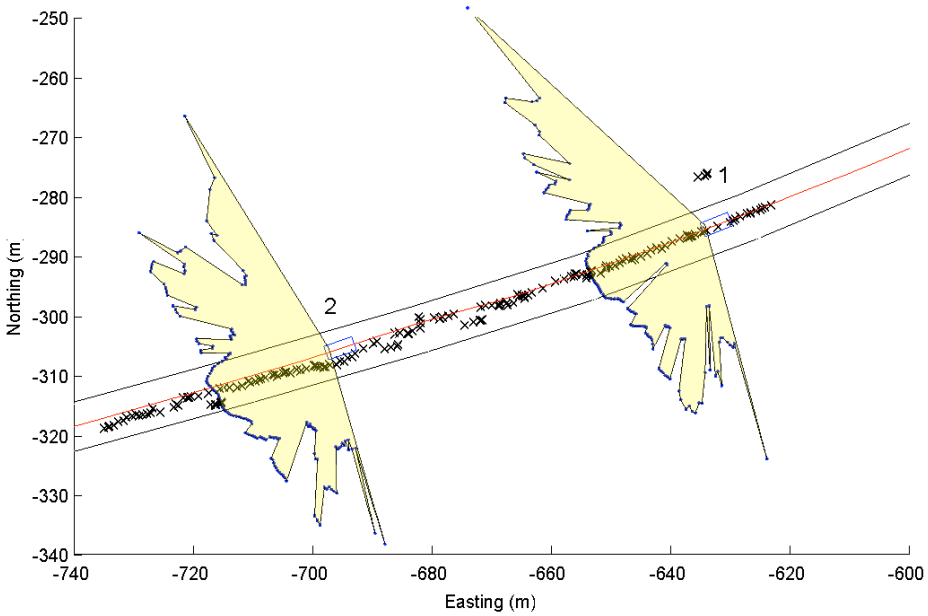

Figure 3.8. A pair of scans taken by our test vehicle several seconds apart during autonomous operations on a slight right-hand curve. The sensor is mounted over the cab of the vehicle with approximately -7 degrees pitch. Pitching and rolling of the vehicle and sensor cause the intersection of the scan with flat ground to move relative to the vehicle. Estimated vehicle (roll, pitch) in position 1 are (1.21, -1.74) degrees and for position 2 are (0.94, -1.58) degrees. Approximate road boundaries are shown as parallel lines. The $\times$s are the result of the road-feature extraction techniques of section 3.3.5. The thin line is the trace of the vehicle position as it followed the road estimate.

The matrix K_k represents the Kalman gain here; see [55] for a good background reference. Note that since the matrices A and C depend on the current speed and current look-ahead distance, they are not constant but rather are dependent on the timestep k.

The inputs to this extended Kalman filter implementation are the sequence of estimates y_k for the road centerline lateral coordinate (in the road coordinate system $\mathcal{R}$) at the look-ahead corresponding to a given scan. The calculation of these estimates, given the range images and sensor pose estimate, is the subject of the next section.

3.3.5 Road Feature Extraction

A LADAR image array can be considered a 2D Cartesian pixel map where each pixel (u, v) represents an (azimuth, elevation) pair (θ, ϕ). The values in this pixel map for a given scan represent the range measured in that direction. Let the sensor x-axis be aligned with the measurement at $(\theta, \phi) = (0, 0)$. If we consider a Cartesian coordinate system with its x-axis pointed along the range measurement vector, a ZYX Euler rotation with (roll, pitch, yaw) =

Figure 3.9. Visual forward image corresponding to position 2 of fig. 3.8. LADAR-based road-feature estimation has the advantage of insensitivity to lighting conditions such as long shadows, but is limited to a single look-ahead range. Computation times are significantly faster with LADAR feature extraction.

$(0, -\phi, -\theta)$ will transform the point $p = (\rho, 0, 0)$ into the sensor frame. For our single axis LADAR, the elevation is zero and the azimuth lies in the interval $[-90,\ 90]$ degrees. The range image becomes one dimensional, and the Cartesian coordinates in the sensor frame are reduced to

$$\begin{bmatrix} x_s \\ y_s \\ z_s \end{bmatrix} = \rho \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix}.$$

Since we are using a range sensor for feature extraction, we are able to transform any feature between any of the coordinate systems we have defined so far – namely the image coordinates $\mathcal{I}$, sensor coordinates $\mathcal{S}$, road coordinates $\mathcal{R}$, or global coordinates $\mathcal{G}$ – with ease. We have a choice, therefore, of performing feature extraction in any of these coordinates.

Fig. 3.8 depicts two full range scans from a bird's-eye perspective, where the LADAR sensor is rigidly mounted on the front of the vehicle at a height of 2.4 meters and pitched downward by approximately 6.9 degrees with respect to flat ground. On flat ground in this configuration, the scan plane intersects the ground on a line perpendicular to the vehicle orientation at approximately 20 meters from the sensor position. Positive pitching of the vehicle causes this line to move further away, and positive roll causes the line to rotate clockwise. Feature extraction in this space can be based on finding and filtering the straight segments that correspond to the road surface and road boundaries. This has been achieved with considerable success as shown in [58] for navigation at low speeds (up to 5 m/s) in urban environments.

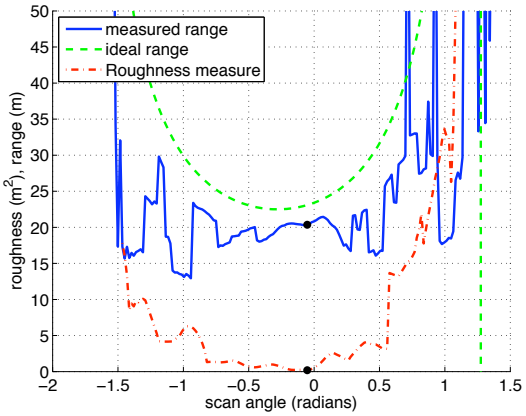

Figure 3.10. The range image corresponding to position 2 in fig. 3.8, and the image that would result in a scan of flat ground. The difference between these two is used to extract features that correspond to the road centerline.

Fig. 3.10 shows the range image corresponding to the scan shown in position 2 of fig. 3.8. Features in the scan associated with vegetation, road berms, and flat road are all apparent to the trained human eye, if correlated with the camera image of fig. 3.9.

The method used in this work to extract road features is to perform a search in the image plane for a best candidate section for flat ground. With the assumption of flat ground, the ideal range image is a function of the height of the sensor above the ground z_s, the pitch of the sensor ϕ_s, the roll of the sensor ψ_s, and the sensor scan angle θ, and is given by

$$\rho_{\text{flat}}(\theta) = \frac{z_s}{\cos\theta \sin\phi_s - \cos\phi_s \sin\psi_s \sin\theta}.$$

The feature extraction of a point on the road centerline is done by considering the difference $\rho_e(\theta)$ between the LADAR range image $\rho(\theta)$ and the flat ground range image $\rho_{\text{flat}}(\theta)$ as calculated above. An optimization problem is posed to find the most likely road center feature in the scan, and can be expressed as

$$\min_i \sigma(\rho_e(W_i))$$

where $\sigma(\cdot)$ represents a variance and W_i is the scan angle interval that corresponds to the road width (which is here assumed to be fixed). Fig. 3.10 also shows the solution of this optimization (performed by brute force but still more than fast enough for real-time implementation) for the scan shown.

The search is performed through the image plane by considering several overlapping discrete window positions in the scan as candidates for a road cross section. The smoothness of each

road-window candidate is calculated by taking the variance of $\rho_e(\theta)$ for the range of θ that corresponds to the window. The range measurement at the center of the minimum variance window is used to compute the measurement for each update step of the extended Kalman filter.

Note that this algorithm will work only in the situation where the roughness of the road is small relative to the roughness outside the boundaries of the road. If the terrain on either side of the road is also smooth, false outlier road features are likely to occur. If road berms are geometrically significant, restricting the search to a region around the current estimate of the road can improve results in this situation.

3.3.6 Simulation Results

The extended Kalman filter estimation scheme as presented above, but with the curvature and curvature rate formulation, was simulated over the road geometry depicted in fig. 3.6. The vehicle was simulated to move along the road at 5 m/s and scans were simulated to provide noisy measurements of the road centerline. The process noise covariance was set to $Q = \text{diag}(5.0\times10^{-5}, 5.0\times10^{-2})$ and the measurement noise variance was set to $R = 3.0\times10^{-4}$. The value of the process noise covariance was chosen to be consistent with results from previously published literature [24]. The initial covariance estimate was set as $P_0 = \text{diag}(5.0\times10^{-5}, 5.0\times10^{1})$. The resulting estimated versus actual curvature parameters for one of these simulated runs is presented in fig. 3.11, along with a trace of lateral error as a function of arc length from the beginning of the simulated road.

The noise provided in this simulation is significant and is designed to be comparable with the noise observed from analysis of collected LADAR scans using the feature extraction methods of section 3.3.4. Even with this amount of noise, these simulation results are comparable to previous results; see for example [24] for results from a two-clothoid model.

A few practical comments on the estimator design are warranted. For the look-ahead distances used here (approx. 20 m), no estimator parameters were found that gave good estimation performance of $\kappa_1(s)$. Poor κ_1 estimates sometimes caused oscillations in the κ_0 estimates, which provided a large part of the incentive to investigate the heading and curvature formulation in real-world runs.

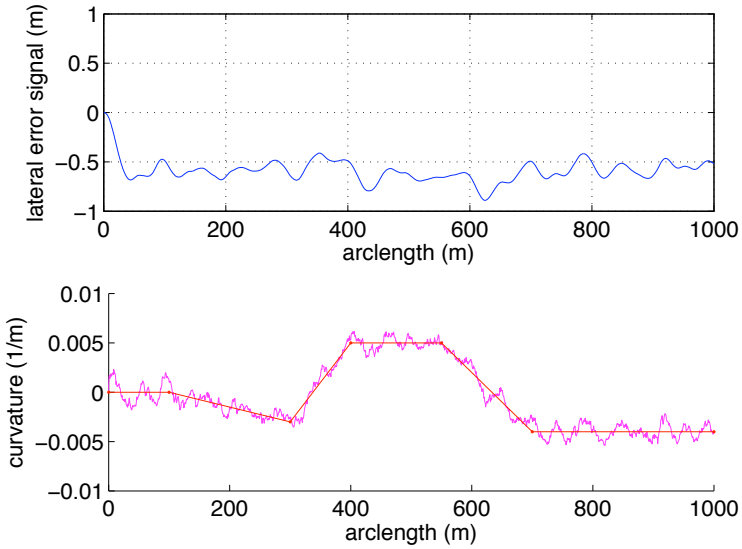

Figure 3.11. Typical simulation performance for the simulated road geometry given in fig. 3.6. Noisy data were simulated to generate lateral offset estimates from the current road geometry estimate. Simulation results show good tracking of the curvature of the road and small lateral errors even at curvature transition segments. Curvature rate estimates are not shown.

3.3.7 Autonomous Operation Results

The EKF as designed above was also implemented to determine the performance of the road estimation scheme with real data, processed using the methods in section 3.3.5 to provide road-feature estimates in the road coordinate system to the extended Kalman filter presented in section 3.3.4. The process and measurement noise covariance were set to be $Q = \text{diag}(5.0\times10^{-4}, 5.0\times10^{-5})$, $R = 5.0\times10^{2}$. These were tuned heuristically to find a good balance between matching the local curvature and reducing the lateral offset error in the estimate.

The data presented were collected while running the road-feature detection and road geometry estimation as described above, in closed loop, while our test vehicle controlled the throttle, brake, and steering in order to track the road estimate. Successful tracking of the road was consistently achieved while driving a section of desert road at speeds between 4 and 6 m/s.

LADAR scans and synchronized state data were collected at a rate of approximately 75 Hz. Fig. 3.12 shows some sample statistics from the collected data, including traveling speed, roll, pitch, and relative yaw to the estimated road centerline features. These are indications of the degree to which the flat ground and the constant speed assumptions are violated, which require us to use the extended Kalman filter implementation to account for changing look-ahead distances and non-constant process matrix A_k.

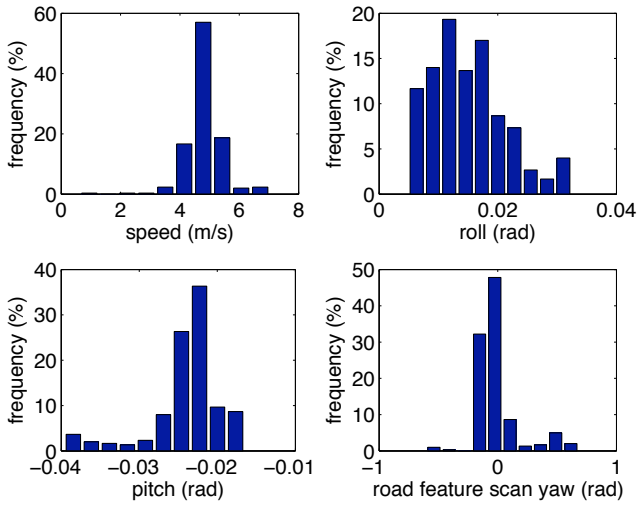

Figure 3.12. Speed, pitch, and roll histograms for a sample segment of the data on which the techniques presented in this work were tested. These data represent a sampling of 1,000 full LADAR scans with 181 scanpoints each, which corresponds to about 43 seconds of data at 75 Hz.

Example performance of the algorithms on collected data is depicted in fig. 3.8. Approximate road boundaries are shown, as well as extracted road centerline feature estimates (×s) and the resultant vehicle trace. These results indicate the performance of the road-feature extraction algorithms of section 3.3.4, although outliers are present. The results also indicate the ability of the filtering algorithm to handle outlier as well as noisy data. In particular, the road estimate conforms well to the series of measurements, while not being affected terribly by the outliers.

3.3.8 Summary and Extensions

We have developed an extended Kalman filter framework for estimation of road geometry that has been applied in simulation and to closed-loop autonomous operation in off-highway environments. Results from simulation and from real data indicate that reliable estimation and tracking of off-highway road geometry is possible for use in autonomous systems and future intelligent vehicles. We have presented feature extraction methods for the road centerline that provide good tracking in moderate off-road environments.

Several improvements and extensions are the subject of future research. In general, careful testing in a wider variety of road types will provide insight for the improvement of these algorithms and likely enable us to eliminate several of the assumptions presented in the beginning of this section.

The extent to which the filter is able to handle outlier measurements is limited. Exploration of other solutions to the optimization problem posed in section 3.3.4 for extraction of road-feature estimates will be necessary to improve the conditions under which the application of these techniques will be successful. Initial investigation suggests that restricting the search for the road features in a scan to a local region around the current estimate may prove to be an intelligent way to reduce the number of outlier measurements.

In addition, there is clear benefit to extending the estimation framework to include online estimation of road width, so that the vehicle has sufficient information to determine whether it can navigate potential obstructions while still remaining on the road. Supervised learning, matched filter, and other techniques could enable one to extract estimates of left and right road boundaries as well as road center.

Finally, there are indications that significant performance increases might be achieved by combining road-feature extraction techniques from LADAR sensing and monocular vision to generate a unified road model for hetereogeneous sensor suites. These would take advantage of the long range sensing properties of the camera (enabling vanishing point detection) with short range geometric information from the LADARs to better estimate the global road geometry.

3.4 Alternative Methods

The road estimation work presented above might be improved through use of the alternative methods of moving horizon estimation or particle filtering. The gains of taking such an approach remain to be seen, but it is expected that such methods will provide improved robustness to non-Gaussian measurements.

Chapter 4

Alice: An Autonomous Vehicle for High-Speed Desert Navigation

This chapter provides a detailed description of the design strategy, system architecture, and components of Alice, a full-size high speed autonomous vehicle finalist in the 2005 DARPA Grand Challenge. Alice's design and implementation took advantage of several important achievements in robotics and control over the past twenty years. The path planning strategy employed is a real-time implementation of receding horizon control [38], where the planned path is a result of a constrained optimization that takes into account vehicle dynamics and spatially encoded speed constraints. This method is carried over from previous application to flight control systems [15]. The system architecture was designed and implemented based on the experience of Team Caltech's entry in the 2004 Grand Challenge (Bob) and on the experience of the other finalists in that competition, notably that of the Red Team [53]. A supervisory control element was implemented that made use of new ideas employed at the Jet Propulsion Laboratory for fault protection of complex systems in the form of the Mission Data System [46].

While an attempt is made to make this chapter comprehensive, special attention is paid to the systems engineering, algorithmic, and software aspects of the design of Alice. Deeper analysis of the hardware and some of the software components of Alice can be found in [12] and other references cited in the appropriate subsections.

Alice represents the culmination of three years of effort and experience for more than one hundred undergraduate students, four graduate student coordinators, four faculty advisors, and many external reviewers, totaling several tens of thousands of person-hours. The resulting system is proficient at navigating autonomously at speeds up to 15 m/s (35 mph) in clear conditions, deciding online appropriate navigation speeds depending on terrain roughness and difficulty, detecting and incorporating road information into its plans, and satisfying kinematic and dynamic constraints in its path planning and execution. A key quality of Alice's design

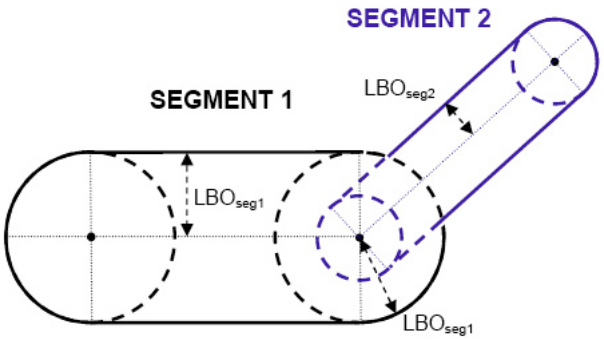

Figure 4.1. Specification of route boundaries for autonomous navigation was done with a list of GPS waypoint locations and lateral boundary offsets (LBOs) as indicated. Each waypoint segment was also assigned a speed limit. Source: DARPA Route Data Definition File document.

is that it does not rely on human preplanning of paths or any *a priori* data other than the Route Data Definition File (RDDF). It uses the RDDF as a mask on the possible regions for navigation, but does not in particular favor the centerline of the corridor, instead finding the most desirable navigation path within the limits of the corridor.

This chapter will serve as an exposition of the essential features and details of Alice's implementation and can be considered a guide for development of an autonomous vehicle for the purpose of navigation in outdoor unstructured environments. It is presented in the context of the DARPA Grand Challenge, but it serves as a general strategy for successful development of a research testbed for autonomous navigation.

4.1 DARPA Grand Challenge: Background

The DARPA[1] Grand Challenge was a race for autonomous ground vehicles that was announced in July 2002.[2] Two races were run in association with the Challenge, the first on March 13th, 2004 (142 miles across the Mojave desert of California) and the second on October 8th, 2005 (132 miles through the desert around Primm, Nevada), with fifteen finalists in the first race and twenty-three finalists in the second. For each race, the field of finalists was narrowed down from a larger number in advance through a series of applications, qualifying events, and site visits.

Two hours in advance of each race, teams were given a Route Definition Data File (RDDF) consisting of a set of waypoints, lateral offsets, and speed limits. As detailed in the event rules

[1]Defense Advanced Research Projects Agency
[2]Substantial details were released at a DARPA-sponsored conference in February 2003.

Table 4.1. Basic statistics from route definitions for two Grand Challenge races.

Course	Number of Waypoints	Distance (mi)	LBO (ft)			speed limit (mph)		
			Min	Max	Median	Min	Max	Median
2004 Race	2,586	142	6	800	8	2	60	20
2005 Race	2,935	132	5	50	6	5	45	20

[1], this defined a corridor segment from each waypoint to the next with a boundary defined at the lateral offset distance from the line segment between adjacent waypoints (see fig. 4.1). A speed limit was imposed for each corridor segment. The course boundaries consisted of the union of the corridor segments, and each vehicle was expected to traverse the corridor segments in order while following roads along the course and avoiding obstacles along the way. The first team to finish the course in less than ten hours was to be awarded a grand prize of one million dollars ($1,000,000). In the first race, no vehicle completed more than 7.4 miles (about 5%) of the course, leading to the announcement of a second race and an increase of the prize to two million dollars ($2,000,000).

The route definition data file (RDDF) represented free *a priori* knowledge about the path from the first to last waypoint. The required degree of autonomy of the competing vehicles was specified by the nature of this RDDF, including what percentage of its interior is traversable and whether the speed limits reflect actual safe autonomous driving speeds. Statistics for the RDDF for the 2004 and 2005 races are displayed in table 4.1. By several measures, the 2005 race course was an easier course for vehicles to complete: there were more waypoints, the total distance was shorter, speed limits were lower, and boundaries were tighter, all of which required less autonomy in choosing the vehicle path. In addition, no manmade obstacles appeared on the interior of the course and all desert roads were nicely graded before the race. The fact that only five of 23 teams completed the course is a testament to both practical and research challenges involved in doing so.

4.2 Team Caltech

Team Caltech was formed in February 2003 to design, build, and test a vehicle to compete in the 2004 DARPA Grand Challenge. It consisted of primarily undergraduate students, with a few graduate student technical coordinators and faculty advisors. Our first vehicle, Bob (fig. 3.1), executed a behavior-based architecture and was one of only six to traverse more than one mile (1.3 miles) of the 2004 race course.

Team Caltech's mission was twofold. First, we sought to provide a high quality educational experience in multidisciplinary systems engineering. The team's activities were organized around a year-long course that spanned Computer Science, Mechanical Engineering, and Electrical Engineering disciplines, and summer activities were supported through Caltech's Summer Undergraduate Research Fellowship (SURF) program. Second, in a manner consistent with this first goal, Team Caltech sought to design, build and test a vehicle to win the 2005 DARPA Grand Challenge.

Based on these high-level goals, a top-level system specification was defined that drove development strategy. The team adhered to a total equipment budget of $120,000, excluding donations. According to the rules of the DARPA Grand Challenge, no sources of government funding were used. Additional specifications were derived based on the rules, other guidance from DARPA, and best guesses as to the nature and difficulty of the course; these specifications included the following:

S1) 175 mile (282 km) range, 10 hours driving, 36 hours elapsed (with possible overnight shutdown and restart).

S2) Traverse 15 cm high (or deep) obstacles at 7 m/s, 30 cm obstacles at 1 m/s, 50 cm deep water (slowly), 30 cm deep sand and up to 15 deg slopes. Detect and avoid situations that are worse than this.

S3) Operate in dusty conditions, dawn to dusk, with up to 2 sensor failures.

S4) Safe operation that avoids irreparable damage, with variable safety factor. Safety driver with ability to immediately regain control of vehicle; ability to withstand 20 mph crash without injury to occupants.

S5) Rapid development and testing: street capable, 15 minute/2 person setup.

S6) Average speed versus terrain type according to table below. (Total mileage was specified as 175 miles maximum, here total mileage is set at 132 miles for comparison with 2005 race course).

Terrain type	Speed (mph)			Expected Distance		Time (hr)
	Min	Max	Mean	mi	%	
(P) Paved road	20	30	25	12	9.1%	0.48
(D1) Easy dirt road	20	30	25	44	33.3%	1.76
(D2) Moderate dirt road	15	25	20	43	32.6%	2.15
(W) Dirt road, winding	15	25	20	4	3%	0.20
(R) Dirt road, rolling	15	25	20	9	6.8%	0.45
(M1) Mountain road, moderate	10	20	15	11	8.3%	0.73
(M2) Mountain road, extreme	5	10	7.5	8	6.1%	1.07
(S) Special (off-road, tunnels)	3	8	5.5	0.8	1%	0.19
Total	3	30		132	100%	7.03

The speed versus terrain table was developed by analyzing the 2004 DARPA Grand Challenge race course and choosing a representative number of distinct terrain types, as determined by factors of roughness, degree of rolling, degree of winding, and road width. Since a vast majority of the course consisted of dirt roads of various forms, all but one of the terrain categories are breakdowns of different representative types of roads. The average speed specifications for each terrain type were chosen to provide a reasonable margin for finishing 175 miles (the maximum prescribed 2005 course distance) in ten hours or less.

All of the specifications were reviewed and revised as necessary throughout development. Some ambitious early specifications, such as being able to withstand a computer failure, were deemed unnecessary; it was deemed more effective to eliminate this specification and take steps to reduce the chance of computer failure.

In order to satisfy these specifications, Team Caltech organized itself around three main race teams: terrain, planning, and vehicle/embedded. The scope and responsibilities of these teams were divided as follows:

- **Terrain**: Sensor selection, sensor configuration, sensor processing, vehicle pose estimation, environment mapping, sensor fusion of real-time terrain and road data with RDDF information

- **Planning**: Path planning and evaluation, trajectory tracking, actuator characterization, supervisory control, graphical user interface, data logging, playback and visualization

- **Vehicle/Embedded**: Vehicle chassis, power delivery, actuation, sensor and computer mounting, E-stop integration

Team Caltech maintained a focus on field testing in race conditions throughout development. An informal qualification procedure was followed for new and updated software components, from planar simulations to live data playback and validation of component behavior, to live testing on vehicle under increasing levels of autonomy, increasing terrain difficulty, and increasing speeds.

An initial additional focus on simulation capabilities included developing the ability to simulate the six degree-of freedom vehicle, sensors, and collisions in the Gazebo 3D simulation environment [17]. We also developed a custom in-house planar kinodynamic simulator, which was used primarily for trajectory tracker development. For each simulator the achieved goal was to test the same software that runs on the vehicle with a minimum of reconfiguration.

While end-to-end system simulation capabilities from sensors to actuator drivers seemed feasible in the Gazebo environment, this endeavor was deliberately abandoned during development for several reasons. Although we were able to demonstrate such end-to-end capabilities with a small number of simulated LADAR sensors in simple environments, there remained a significant gap between such demonstrations and real-world performance. As closing this gap by adding more sensors would result in slower than real-time simulation performance, and still leave significant uncertainty in vehicle, sensor, and environment models, we chose to abandon such ambitions. Instead, focus was placed on capabilities for testing system components using playback of logged data and on real-world system testing of the vehicle.

4.3 Alice: A Platform for Autonomous Desert Navigation

A top-down flow of the system specifications drove the design for the race teams, software, and hardware components of Alice. These are described in the following sections.

4.3.1 Hardware Description

The vehicle design choices for Alice were made so that it would be robust to driving in particularly harsh desert terrain, with a focus on durability over ride quality. Alice was built on a 2005 Ford E-350 chassis with significant modifications for off-road navigation made by Sportsmobile West of Fresno, California.

Power delivery for computing, sensing, and actuation components was supplied from a 3 kilowatt generator hinge mounted to the rear door. This generator fed two 1500 watt inverters/chargers which provided direct power and charging of four deep cycle marine batteries. The

Figure 4.2. Alice in the 2005 Grand Challenge.

batteries delivered power when the generator was not operating, such as during transition to externally supplied 120 VAC power.

The steering, throttle, brake, transmission, and ignition were all outfitted with actuators for computer-controlled operation. Steering was actuated by a chain drive to the steering column with a DC servomotor, brake actuation was achieved with a series of pneumatic pistons, transmission was controlled with an electric linear actuator, and throttle and ignition were electronically controlled. Interfaces for on-board diagnostics (OBD II) and an emergency-stop (E-stop) system were also installed, along with a series of dashboard switches that allowed the safety driver to immediately regain control of any or all of the actuated components if necessary.

A multi-threaded program called Adrive provided the software abstraction to all of the vehicle hardware and actuation. This program handled the receiving of software actuator commands, translation of these into hardware signals via the RS-232 protocol, reading and reporting position and status for each actuator, handling the logic associated with the E-stop interface, and managing actuator fault protection and recovery.

Focus on a rapid development cycle and ease of testing led to a choice of a vehicle that was able to carry a safety driver and three developers able to monitor and improve Alice's performance throughout testing. The vehicle was configured with two front and two rear racing seats, each outfitted with five point harnesses. A shock-isolated computer box was centrally located and served as a mount point for monitors for the rear workstations. An eight-input, four-output keyboard-video-mouse switch was installed to provide independent workstations

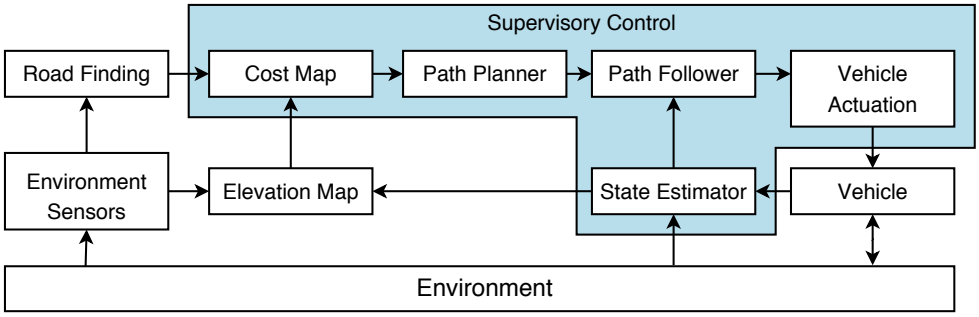

Figure 4.3. System architecture for Alice.

for onboard monitoring and development, and a laptop onboard served as a field server for collaborative code management, bug reporting, and documentation. Gigabit Ethernet provided communication between the seven onboard computers, and it was augmented with wireless hub and wireless bridge communication for low bandwidth operations from outside the vehicle.

The rugged physical capabilities of Alice were intended to handle challenging terrain similar to the 2004 Grand Challenge course, and were more than sufficient to handle the less physically and technically demanding 2005 course. In hindsight, the rapid development specification was essential, and the onboard development capabilities of Alice proved (and continue to prove) to be indispensable. The safety driver function allowed more aggressive testing strategies and faster recovery from mistakes during development, and onboard development enabled a more accelerated testing schedule.

4.3.2 Deliberative System Architecture

Alice's navigation system was built on a deliberative control framework as a result of a principled evolution of the system architecture of Bob, Team Caltech's finalist in the 2004 DARPA Grand Challenge (DGC). Typical sensor suites of DGC finalists were not omnidirectional; that is, at any given time they did not have a combined field of view that covered all of the terrain of interest around the vehicle. For this reason, it was necessary for the autonomous vehicle to build an understanding of its environment over time. This rules out the effective use of purely reactive systems, which by definition do not maintain internal representation of the environment when translating sensory inputs to actuation.

The architecture chosen for Alice embraced the deliberative control architecture approach discussed at length in chapter 2, with distinct and serially organized mapping, path-planning, and path-following components. The first defining feature of this architecture as opposed to

behavior-based ones is the separation of the model of the environment from the actual environment itself. The second defining feature is the existence of a path planner, which serves to choose between potential alternate future states.

The system architecture diagram of fig. 4.3 depicts the essential components and their interaction. The architecture possessed a nested feedback control loop structure, with a path-following inner loop that managed the uncertainty of the vehicle dynamics and interaction with the environment, and a path-planning outer loop that provided new paths in response to a continuously updated map model. State estimation was performed in a global, inertial coordinate frame with the aid of commercially available differential GPS, and knowledge of six degree of freedom sensor pose at any given time allowed for the map model to be represented in this global coordinate frame.

Vehicle state estimation for Alice is discussed in more detail in section 4.3.4, and the sensor fusion and mapping are elaborated in section 4.3.6. The environment map combined the information from the RDDF, detected road geometry, and terrain elevation into a single layer of the environment that encoded, in each cell in the map, the estimated maximum speed at which any part of the vehicle could traverse that cell. These speeds served as both constraints and cost for path planning based on gradient-based optimization of the traversal time through the RDDF. This optimization was performed to a speed-dependent horizon of no more than 100 meters. Alice, therefore, had the capability to navigate using only local online path planning and without relying on any offline route planning based on static data and human input. The details of the path planning implementation are presented in section 4.3.7.

The output of the planning process was a spatially consistent sequence of trajectories that satisfied vehicle dynamic and kinematic constraints, as well as constraints imposed by the mapping process. A trajectory follower module calculated the actuation commands that would best keep the vehicle on this spatial path and driving at the specified speed. This module is described in detail in section 4.3.8.

4.3.3 Network-Centric Approach

The system architecture was implemented in a networked computing environment of six Dell PowerEdge servers and one IBM dual-core dual-chip AMD server. Modules were run on different servers to balance load, and communication between modules took place using the Spread Toolkit for network communications [2, 3], which sits on top of the standard TCP and UDP protocols.

The message interfaces between modules utilized a set of commonly used message types,

Table 4.2. Message categories, rates, and cumulative throughput for Alice's software during the 2005 Grand Challenge.

Message Type	Approximate Cumulative Send Rate (Hz)	Throughput (kbytes/sec)
Fused map	40	614
Trajectories	4	529
LADAR elevation	50	474
State estimates	150	28.6
Status + fault protection	280	18.7
Actuator commands	90	7.7
Road data	14	7.6
Stereo elevation	1	1.6
Total	714	1681

listed in table 4.2. These standard message types (vehicle state, range measurement, map update, trajectory and actuation command) are described in the following paragraphs.

The vehicle *state estimate* message consists of information needed to establish vehicle pose information for the purposes of mapping, trajectory generation, and trajectory following. The vehicle state message consisted of northing, easting, altitude, roll, pitch and yaw estimates for a coordinate system fixed to the vehicle in a forward-right-down configuration, as well as the first and second derivatives of each of these values, a timestamp, and information about the confidence in these estimates.

Range or *elevation measurement* messages consisted of three dimensional points in space in the global coordinate system (northing, easting, altitude) that represent the location of a range return from onboard laser detection and ranging (LADAR) sensors or stereo vision.

A *map update* consisted of a set of (northing, easting, value) triplets that provided either replacement values or input measurements for communication of new map data. Maps consisted of several layers, and the value of a map update could represent elevation, speed, confidence, or another user-defined value.

A *trajectory* was a dense discrete sampling of a twice differentiable planar curve for the vehicle to follow, where each point in the trajectory encoded the following values:

- northing (N) and easting (E), the Universal Transverse Mercator (UTM) Cartesian projection of latitude and longitude,
- the first derivatives of northing and easting, $\dot{N}$ and $\dot{E}$, and
- the second derivatives of northing and easting, $\ddot{N}$ and $\ddot{E}$.

This encoding allowed the trajectory follower to calculate the nominal speed and lateral and

Table 4.3. State estimation sensors used on Alice.

Sensor Type	Mounting Location	Specifications
IMU (Northrop Grumman LN-200)	Roof	1°–10° gyro bias, 0.3–3 mg acceleration bias, 400 Hz update rate
GPS (Navcom SF-2050)	Roof	0.5 m Circular Error Probable (CEP), 2 Hz update rate
GPS (Novatel DL-4plus)	Roof	0.4 m CEP, 10 Hz update rate

longitudinal acceleration for any point on the trajectory, assuming conditions of negligible tire sideslip. We limited lateral accelerations sufficiently when computing the paths to justify this assumption.

An actuation command message consisted of throttle, brake, and steering commands for the driving software to follow. In return, measured throttle, brake, and steering commands as well as the status of these actuators were returned by the driving software.

A summary of the message types described along with statistics on their throughput and rates are shown in table 4.2. These data were extracted by processing the message log from the 2005 race. By comparison, raw data rates from sensor hardware are approximately 350 Mb/s (44 MB/s).

4.3.4 State Estimation

As the vehicle moves throughout its navigation, it is essential for map building to register the pose of each of the sensor measurements in order to represent them in a single coordinate system. Because the sensors used on Alice were all fixed to the vehicle, this amounted to measuring the mounting parameters of each of the sensors and providing an estimate of the vehicle position and orientation in three dimensions at a sufficiently high rate to allow correlation with environment sensor measurements.

State estimation was achieved using a combination measurements from an inertial measurement unit (IMU), a differential global positioning system (GPS) receiver, and onboard diagnostics from the automotive industry standard OBD II protocol (for speed correction). A Kalman filter performed this sensor fusion and tracked and updated the state estimate according to standard inertial navigation equations (see, e.g., [19] and [12] for specific implementation details). The point mass model was used for the propagation step of these equations.

Outages of the GPS signal are generally a standard occurrence from occlusion of satellites by buildings and natural terrain features, and a specific requirement for the DGC was to maintain

Table 4.4. Road detection and mapping sensors used on Alice.

Sensor Type	Mounting Location	Specifications
Road-Finding Camera (Point Grey Dragonfly)	Roof	640×480 resolution, 2.8 mm focal length
LADAR (SICK LMS 221-30206)	Bumper	180° field of view (FOV), 1° resolution, 80 m max range, pointed horizontally

vehicle navigation integrity in response to such events. When GPS outages occured, integrating the inertial accelerometer and gyroscope measurements from the IMU continued to provide an adequate state estimate for a short period of time, but this estimate drifted away from the actual position and orientation after an extended duration. Upon reacquisition of the differential GPS signal, the drifted state estimate and the GPS location could be an appreciable distance apart from each other. In such cases, the state estimate was corrected back to the reported GPS position in the standard way: using the measurement confidence delivered by the GPS unit to assign a covariance to the GPS measurement input to the Kalman filter. An additional mechanism was used for reporting and responding to situations in which the discrepancy between the state estimate and reported GPS location was too large; this is discussed in more detail in section 4.3.9.

4.3.5 Road Finding

Alice employed a monocular camera and a horizontally oriented LADAR unit to detect road geometry for race and development, the details of which are presented in [45]. This method performed remarkably well in detecting road regions for "utility fusion" into the map in the form of speed bonuses. The comparative advantage of the work in section 3.3 is to model curvature of the road centerline, while the method implemented for the race assumed straight road centerlines and had the tendency to cut corners on curves.

4.3.6 Mapping

The maintenance and use of a map of the environment is central to the deliberative architecture employed by Alice. Alice's map is updated continuously and asynchronously by processing data from ten onboard environment sensors consisting of LADAR units, monocular cameras, and stereovision cameras. These sensors and their specifications are listed in table 4.5 for elevation mapping and 4.4 for road mapping. The map format was a 200 m by 200 m Cartesian grid centered on the vehicle location, with a cell size of 40 cm by 40 cm. This map was oriented

Table 4.5. Elevation mapping sensors used on Alice.

Sensor Type	Mounting Location	Specifications
LADAR (SICK LMS 221-30206)	Roof	180° FOV, 1° resolution, 75 Hz, 80 m max range, pointed 20 m away
LADAR (SICK LMS 291-S14)	Roof	90° FOV, 0.5° resolution, 75 Hz, 80 m max range, pointed 35 m away
LADAR (Riegl LMS Q120i)	Roof	80° FOV, 0.4° resolution, 50 Hz, 120 m max range, pointed 50 m away
LADAR (SICK LMS 291-S05)	Bumper	180° FOV, 1° resolution, 80 m max range, pointed 3 m away
Stereovision Pair (Point Grey Dragonfly)	Roof	1 m baseline, 640×480 resolution, 2.8 mm focal length, 128 disparities
Stereovision Pair (Point Grey Dragonfly)	Roof	1.5 m baseline, 640×480 resolution, 8 mm focal length, 128 disparities

along the cardinal directions and shifted along them by multiples of the cell width as the vehicle moved through the environment, keeping the vehicle in the center of the map.

The map was implemented in an extensible manner to have multiple layers for debugging and development purposes. An elevation layer was maintained for each of four single-axis LADAR units and two stereovision camera pairs. Each layer was formed by averaging of 3D point clouds that were transformed from the sensor frame into the global coordinate frame.

The elevation layer was translated to speed according to the local gradient of the terrain. Perfectly flat terrain was marked with a maximum speed according to sensor range, and steep local features above a certain threshhold were marked with a zero maximum speed. Details of the implementation of this conversion, including related tasks of cost fusion, obstacle growing, and assignment of speeds in the absence of terrain data, are provided in [12].

Rather than combining the elevation layers from the different sensors into a single estimate of terrain elevation, the elevation layer for each sensor was converted into a cost/speed map to assign speeds according to the local gradient of elevation. The cost layers from each sensor were then combined using a weighted average, with weights for each sensor determined heuristically. The primary and compelling reason for doing speed-based fusion rather than elevation-based fusion was to avoid sensitivity to calibration errors between sensors. A basic analysis of such sensitivity was given in section 3.2.6.

The decision between elevation fusion and cost fusion for individual range sensors has a direct

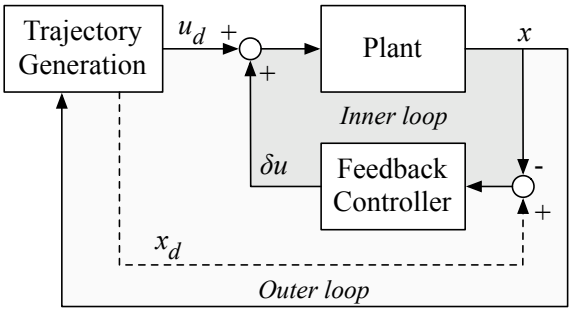

Figure 4.4. The two degree of freedom design for real-time trajectory generation and control. The input to the plant is composed of a feedforward term u_d from the properties of the trajectory and a feedback term δu based on the error between desired and actual state.

interpretation in terms of predictability. If the transformation between two sensors was known to be extremely well calibrated, then the perception of terrain elevation is highly predictable using both sensors, and it makes sense to estimate a single elevation, and to perform deliberative control. Without careful calibration of sensors, terrain elevation is not as predictable, and the cost fusion method represents a shift from deliberative sensor fusion with a unified map toward behavior-based utility fusion with distributed representations (cf. section 2.1.5).

4.3.7 Path Planning

Receding horizon control (RHC) [16, 37, 43] is an method for optimal control of dynamical systems. It relies on a mathematical model of the system to be controlled in order to generate trajectories that satisfy the dynamics of the controlled system, and can include constraints on the system's state and input. Receding horizon control is also called *model predictive control* (MPC), and has early origins in the 1980s in chemical engineering, in applications for which process time constants were large enough to make online computation feasible given the contemporary computing constraints. Since then, increased computing speeds and tools for numerical optimization have made fast online computation possible even for systems with fast dynamics. An example of such a system is the Caltech ducted fan, on which Mark Milam was able to demonstrate real-time nonlinear RHC while satisfying preprogrammed state and input constraints [39].

Receding horizon control is formulated so that the path is computed using the updated vehicle state as an initial condition of the path. Pure receding horizon control can be implemented with no low-level path tracking if the model used to generate the path is a precise representation of the vehicle and actuation dynamics. In many practical implementations, the model is not precise (and disturbances are present), and applied actuator control is a combination of a feedforward

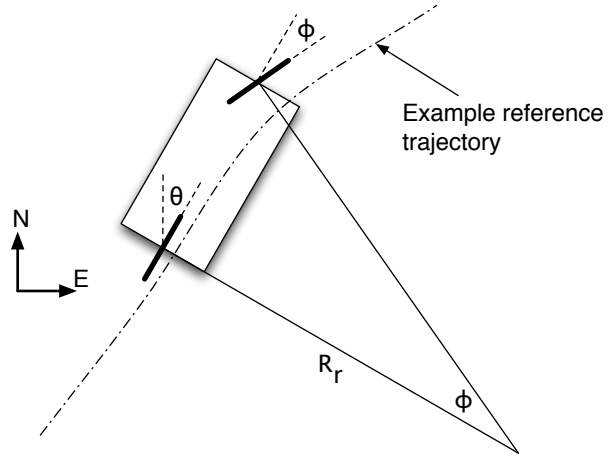

Figure 4.5. Schematic diagram to illustrate derivation of the bicycle model. Slightly different equations can be derived for a front-axle centered coordinate frame.

component computed based on the properties of the planned trajectory and separate feedback control to compensate for deviation from the planned trajectory. This "two degree of freedom" [40] control design is a standard configuration for real-time trajectory generation. The diagram of fig. 4.4 illustrates the inner-outer loop configuration, with the path planning forming the outer feedback loop and the error compensation forming the inner feedback loop.

Errors between the planned trajectory and vehicle state are inevitable whenever disturbances are present or when there is a mismatch between actual vehicle dynamics and the modeled dynamics of the vehicle. This is the reason for using the inner-outer loop design shown. Care must be taken when implementing inner-outer loop control for real-world systems, because the two feedback loops can interact with each other in undesirable ways.

In Alice's software design, path planning and control through the map of the environment were performed as a variation of receding horizon control (see [27] for details). Gradient-based optimization tools were used to find the best path according to a specified cost function, given initial, trajectory, and final constraints on the state of the vehicle. The kinematic bicycle model (fig. 4.5) for the vehicle was used for path planning (as well as path following). With the coordinate system at the center of the rear axle of a vehicle this model is given by the equations

$$\dot{N} = v\cos\theta \tag{4.1}$$

$$\dot{E} = v\sin\theta \tag{4.2}$$

$$\dot{\theta} = \frac{v}{L}\tan\phi \tag{4.3}$$

$$\dot{v} = a \tag{4.4}$$

where (N, E) are the (northing, easting) coordinates of the center of the rear axle that represent (x, y) in the north-east-down coordinate system, v is the speed of the center of the rear axle, θ is the vehicle yaw measured clockwise from north, and ϕ is the steering angle of the front wheels measured positive clockwise. The distance between the front and rear axles is L. These equations are derived based on the no sideslip condition for the rear wheels, i.e., that the direction of travel of the rear wheels (and therefore of the center of the center of the rear axle) is only along the orientation of the vehicle. The $\dot{\theta}$ equation is derived from the geometric relations $\tan\phi = L/R_r = \kappa_r L$ and the definition of curvature, $\kappa \equiv d\theta/ds$. At sufficiently slow speeds, an Ackermann-steered vehicle's tires satisfy the no sideslip condition sufficiently such that this kinematic bicycle model is a reasonable predictor of vehicle planar motion.

In addition, the following vehicle linear and nonlinear constraints are satisfied by the path planner to ensure feasible trajectories:

$$\begin{aligned} v &\in (0,\ v_{max}] \\ a &\in [a_{min},\ a_{max}] \\ \phi &\in [\phi_{min},\ \phi_{max}] \\ \dot{\phi} &\in [\dot{\phi}_{min},\ \dot{\phi}_{max}] \\ v^2\frac{\tan\phi}{L} &\in [-\alpha_{max}, \alpha_{max}]. \end{aligned} \tag{4.5}$$

The constraint that the vehicle speed be always positive is required to avoid divide-by-zero issues when using the differentially flat representation shown below and when computing the feedforward steering angle from the trajectory representation. In Alice's architecture, the ability to stop and reverse was handled in a supervisory manner above the trajectory planning. The last, nonlinear constraint is a limitation on the vehicle's lateral acceleration in order to ensure sufficiently slow navigation around turns in order to avoid excessive sideslip or rollover.

The solution space for the dynamic equations above lives in the class of twice differentiable functions, and is therefore infinite dimensional. As such, numerical optimization problems in this broad class are intractable, and it is necessary to approximate this trajectory space with some finite- and low-dimensional representation. This is done by approximating the solution space by the coefficients of a set of basis functions. B-splines [42] and polynomial splines are convenient and popular choices for this set of basis functions.

The optimization problem for dynamical systems can often be further simplified by exploiting

differential flatness. A system of the form

$$\dot{x} = f(x, u)$$

is differentially flat with output vector z if there is a smooth function of the form

$$z = h(x, u, \dot{u}, \ddot{u}, \ldots, u^{(p)})$$

such that the state and inputs of the system can be written in terms of z and a finite number of its derivatives, i.e., for some smooth functions a and b and integer q,

$$\begin{aligned} x &= a(z, \dot{z}, \ddot{z}, \ldots, z^{(q)}) \\ u &= b(z, \dot{z}, \ddot{z}, \ldots, z^{(q)}). \end{aligned}$$

For the system described by equations (4.1)–(4.4), we can rewrite the system in terms of the variables $\theta(s)$, $v(s)$, and the total distance of the trajectory S_f. Defining l as the arc length along the trajectory, and $s \in [0, 1]$ as arc length normalized by S_f, we can use the change of variables

$$\begin{aligned} (\cdot)' &\triangleq \frac{d(\cdot)}{dl} = \frac{d(\cdot)}{dt}\frac{dl}{dt} \\ \dot{(\cdot)} &\triangleq \frac{d(\cdot)}{dt} = \frac{d(\cdot)}{ds}\frac{ds}{dt} \end{aligned}$$

to derive the equations for the state and output variables in terms of S_f and dependent variables $\theta(s)$ and $v(s)$:

$$\begin{aligned} \dot{N}(s) &= v(s)\cos\theta(s) \\ \dot{E}(s) &= v(s)\sin\theta(s) \\ N(s) &= N_0 + S_f \int_0^s \cos\theta(\sigma)d\sigma \\ E(s) &= E_0 + S_f \int_0^s \sin\theta(\sigma)d\sigma \\ \tan\phi(s) &= \frac{L}{S_f}\frac{d\theta(s)}{ds} \quad \Longrightarrow \quad \dot{\theta} = \frac{v(s)}{S_f}\frac{d\theta(s)}{ds} \\ \dot{\phi}(s) &= \frac{v(s)L\frac{d^2\theta(s)}{ds^2}}{S_f^2 + \left(L\frac{d\theta(s)}{ds}\right)^2}. \end{aligned}$$

The advantage of this representation is that the optimization problem, cost function, and constraints can all be represented in terms of the output variables $\theta(s)$, $v(s)$, and S_f. The function $\theta(s)$ is approximated by the coefficients of a quadratic spline, and $v(s)$ is approximated by a linear spline to ensure continuous speed profiles. With this choice, the acceleration profile can be discontinuous, ignoring engine dynamics, but inner loop feedback control compensates for this intentional model mismatch, allowing optimization in a lower-dimensional space.

The planning optimization problem that was solved is to minimize the following objective function subject to the constraints of eqns. (4.5):

$$J = S_f \int_0^1 \frac{1}{v(s)} ds + k_1 \left\| \dot{\phi}(s) \right\|_2^2 + k_2 \left\| a(s) \right\|_2^2$$

This allowed tuning of the system to achieve more or less aggressive behavior depending on how the time of traversal term balanced against the terms for steering rate and longitudinal acceleration. The planning was implemented in software on top of the SNOPT numerical solver; implementation details are available at [12] and [27].

4.3.8 Trajectory Following

The design specification for the trajectory tracking algorithm was to receive a trajectory from a planning module and output appropriate actuator commands to keep Alice on this trajectory. The inputs to the algorithm were the current state of the vehicle (position and orientation, along with first and second derivatives) and the desired trajectory (specified in northing and easting coordinates, with their first and second derivatives). From these inputs, the algorithm output steering and brake/throttle commands to an actuation module. Goals for accuracy were +0%/-10% for velocity tracking, and ±20 cm perpendicular y-error at 5 m/s, with larger errors allowable at higher speeds. These performance criteria needed to be met on any terrain type found in the system specifications at speeds up to 15 m/s.

System Characterization Before designing the controller, it was necessary to characterize the open-loop dynamics of the system. With this characterization, a mapping from actuator positions to lateral and longitudinal accelerations was obtained. It showed that Alice understeered, and it allowed the determination of safe steering commands at various speeds, such that the vehicle would remain in the linear response region. In this region, the feedforward term was reasonable, and possibly dangerous roll angles/sliding were avoided. Additionally, system delays were determined by examination of the time between commands leaving this module and the

resulting vehicular accelerations.

Control Law Design Although not entirely independent, the lateral and longitudinal controllers were treated separately in the system design. Longitudinal (throttle and brake) control was executed by a feedback PID loop around error in speed plus a feedforward term based on a time-averaged vehicle pitch to reduce steady-state error when traveling up or down hills.

The trajectories received as input to the trajectory follower encoded first and second derivative data as well as geometry of the path, so that desired velocity and acceleration could be calculated at any point on the trajectory. For the longitudinal controller, we decided not to use a feedforward term associated with acceleration based on the input trajectory. This was determined by experience; there were occasions when the feedforward term would overpower the feedback and simultaneous tracking of speed and acceleration was not achievable. For example, the vehicle could not correct error associated with going slower than the trajectory speed if the trajectory was slowing down.

The lateral control loop included a feedforward term calculated from curvature of the path along with a PID loop around a combined error term. The error for the lateral PID is a combination of heading and lateral position errors:

$$C_{\text{err}} = \alpha \tilde{Y}_{\text{err}} + (1-\alpha)\beta\theta_{\text{err}},$$

where $\tilde{Y}_{\text{err}}$ is the lateral position error (saturated at some maximum value $Y_{\max}$), θ_{err} is the heading error, and β is a scale factor. This form was motivated by a desire to obtain stability at any distance from the path. Using this error term, the (continuous) vector field in the neighborhood of a desired path will follow tangent to the path as $Y_{\text{err}} \to 0$ and will head directly toward the path at distances greater than $Y_{\max}$ away from the path.

Note that the use of this error term requires an accurate estimate of the vehicle heading. Systematic biases in this estimate will result in steady-state error from the desired path.

The lateral feedforward term was generated by determining the curvature required by the input trajectory and applying the mapping for steering position to curvature, which yields

$$\phi_{\text{FF}} = \arctan\left(L\frac{\dot{N}\ddot{E} - \dot{E}\ddot{N}}{(\dot{N}+\dot{E})^{\frac{3}{2}}}\right),$$

where N is the northing position, E is the easting position and L is the distance between front and rear wheels. For this calculation, it was assumed that Alice behaves as described by the

bicycle model [14].

To avoid oscillations, an integral reset was incorporated in both the lateral and longitudinal controllers when the relevant error was below some acceptable threshold. In testing, the lateral integral term rarely built up to any significant amount, since performance of the system maintained modest errors comparable to the threshold. For the longitudinal controller, resetting helped to alleviate the overshoot associated with transferring from hilly to flat ground.

Finally, to compensate for system delays, a look-ahead term was added. This term defined the point on the trajectory from which lateral feedforward and longitudinal feedback would be computed.

4.3.9 Contingency Management

The deliberative approach presented above relies on assumptions about and by its constituent components. This is a general feature of deliberative control; since sensor information is abstracted away into a representation of the environment, it is essential that this representation accurately reflect the world. When assumptions made in maintaining this representation are violated, system failure may result.

The supervisory control module served to detect and manage higher-level system faults that other individual modules could not. This included scenarios such as losing and reacquiring GPS, and being stuck on an undetected obstacle. The supervisory control module was also responsible for maintaining forward progress in unusual conditions,

Alice employed a supervisory control module to periodically cross-check important assumptions made by each of the system components and provide the appropriate response when any of these were violated. In the context of the architecture discussion of chapter 2, this supervisory control acted as a high-level *behavior-based* layer for management of uncertainty in the deliberative process.

Examples of conditions where system assumptions were violated include excessive drift of the state estimate, travel outside of the DARPA-specified route corridor, a stuck condition on an unseen obstacle, and perception of no possible route forward in the corridor. In each of these conditions, supervisory control brought the vehicle into a state or sequence of states intended to keep the vehicle safe and to maintain forward progress along the course.

The supervisory control module is described in detail in [12], but three of its strategies are described here for later reference; these are *Slow Advance*, *Lone Ranger*, and *GPS reacquisition*.

- Slow Advance mode was employed when the vehicle's planned trajectory passed through

an obstacle, but had not done so previously. In this case a low maximum speed cap is applied to slow the vehicle. If the perceived obstacle remains, the planner will either 1) report trajectories that bring the vehicle to a stop before the obstacle, 2) create a new plan around the obstacle, or 3) report planner failure, which will result in more aggressive braking.

- Lone Ranger mode was used in situations where Alice's current plan passes through a perceived obstacle and has stopped. In this mode, Alice will attempt to push through this perceived obstacle (and reverse if unable).

- GPS Reacquisition is triggered by GPS measurements that are greater than some threshhold away from the current state estimate. This condition happens if significant state drift occurs during GPS outages. In this case, Alice is brought to a stop while zero-speed corrections fix the discrepancy.

4.4 Experimental Results

The previous sections describe in detail Team Caltech's approach to designing its unmanned ground vehicle, Alice. While Alice shares many subsystems with previous autonomous vehicles and other Grand Challenge entrants, four unique characteristics set Alice apart.

The first such characteristic is its sensor coverage: Alice relied heavily on its disparate combination of sensors to generate information about its environment, making use of all available data when determining the locations of obstacles. In particular, Alice only "forgot" prior map data when a sufficient amount of new map data arrived, regardless of the accuracy of existing measurements, allowing it to "drive blind" for as long as necessary.

A second unique characteristic was the use of speed limit maps to encode information about both the traversability and difficulty of the terrain. This approach allowed Alice to precisely navigate through areas that were complex both in terms of how the vehicle should navigate spatially (e.g., a field of obstacles, through which a precise course must be chosen to prevent collisions) as well as how the vehicle should navigate temporally (e.g., rocky or bumpy areas through which precise speed limits should be followed to prevent damage to the vehicle or instability).

The third characteristic that sets Alice apart from most other Grand Challenge entrants is the fact that its software architecture makes absolutely no assumptions about the structure of the race course as defined by the RDDF. In particular, Alice makes no use of *a priori* map

data, thereby enhancing its flexibility; it would be just as able to drive a desert road in a foreign country as it would in the Mojave desert. Additionally it does not assume that the RDDF will "hold its hand," so to speak, by constraining it to a road, or forcing it to decelerate around turns or in rough terrain. In fact, a great deal of testing was done in which the corridor's width was set very high (on the order of dozens of meters) and the speed limit was chosen to be essentially infinity. Despite the open-endedness of this problem, Alice's architecture allowed it to consistently choose an optimal course and speed.

The fourth and final feature that set Alice apart from many other GCE entrants was its physical robustness. While there were several other entrants who would likely have been at least as physically capable as Alice in a true off-road environment, Alice's tough physical exterior enabled it to take risks that may have damaged some vehicles catastrophically.

4.4.1 Desert Testing

Team Caltech documented over 300 miles of fully autonomous desert driving in the Mojave desert with Alice from June 2005 to the National Qualifying Event in Fontana, California in September 2005.

Approximately 33 of these miles were driven on the 2004 Grand Challenge race route during the week of June 13th, 2005. Alice traversed these miles with a testing team of four people inside, scrutinizing its performance and making software improvements and corrections. Over the course of three days, Alice suffered three flat tires including a debilitating crash into a short rocky wall that blew out the inside of its front left tire and split its rim into two pieces. This crash was determined to be caused primarily by a lack of accurate altitude estimates when cresting large hills. Along with several related bug fixes, an improved capability to estimate elevation was added to the state estimator.

The majority (approximately 169 miles) of autonomous operation for Alice took place in the two weeks leading into the National Qualifying Event. This operation included a full traverse of Daggett Ridge (the most difficult section of the 2004 race course) at 4 m/s average speed and significant operation in hilly and mountainous terrain. The top speed attained over all autonomous operations was 35 mph. The longest uninterrupted autonomous run was approximately 25 miles.

4.4.2 National Qualifying Event

As one of 43 teams at the California Speedway in Fontana, Alice successfully completed three of its four qualifying runs. Several of its runs were characterized by frequent stopping as a result

Run	Gates	Obst	Time	Issues
1	21/50	0/4	DNF	State estimate problems after tunnel
2	44/50	4/4	12:45	
3	49/50	5/5	16:21	Multiple pauses due to short in E-Stop wiring
4	44/50	5/5	16:59	Frequent stops due to state drift

Figure 4.6. Alice on the National Qualifying Event course (left). Table of results from Alice's four runs at the event (right).

of the performance of the state estimator under conditions of intermittent GPS. Specifically, under degraded GPS conditions its state estimate would drift considerably, partially due to miscalibration of IMU angular (especially yaw) biases and partially due to lack of odometry inputs to the state estimator. However, zero-speed corrections were applied to the Kalman filter when Alice was stopped, which served to correct errors in its state estimate due to drift.

During Alice's first national qualifying event (NQE) run, zero-speed corrections were not applied in the state estimator. Accordingly, drift accumulating in the state estimator was not corrected adequately when Alice stopped. Travel through the man-made tunnel produced drift substantial enough for Alice's estimate to be outside the RDDF, which at the time resulted in a reverse action. Alice performed a series of reverse actions in this state, going outside the actual corridor, causing DARPA to pause it and end its first run.

Zero-speed corrections were added to the state estimator after Alice's first run, enabling it to successfully complete all subsequent runs, clearing all obstacles and 137 out of a total 150 gates. Completion times were slow for runs 2, 3, and 4 partially due to frequent stopping as a result of state estimator corrections. Fig. 4.6 provides a summary of Alice's NQE run performances.

4.4.3 Grand Challenge Event

When Alice left the starting line on October 8th, 2005, all of its systems were functioning properly. However, a series of failures caused it to drive off course and topple a concrete barrier, resulting in a DARPA E-stop PAUSE and subsequent DISABLE. Although, as mentioned above, the system we have described performed well over the course of hundreds of miles of testing in the desert prior to the Grand Challenge, we believe the pathological nature of this particular failure scenario demonstrates a few of the more important weaknesses of the system and exemplifies the need for further ongoing research. We will begin by providing a brief chronological timeline of the events of the race leading up to Alice's failure, followed by an analysis of what weaknesses

contributed to the failure.

Alice's timeline of events in the Grand Challenge was as follows:

- Zero minutes into the race (9:03 AM), Alice left the starting line with all systems functioning nominally.
- Approximately four minutes into the race, two of its midrange LADARs entered an error mode from which they could not recover, despite repeated attempts by the software to reset. Alice continued driving using its long and short-range sensors.
- Approximately 30 minutes into the race, Alice passed under a set of high-voltage power lines. Signals from the power lines (and, possibly, electronic equipment in the vehicle pit area) interfered with its ability to receive GPS signals, and its state estimate began to rely heavily on data from its Inertial Measurement Unit (IMU).
- Approximately 31 minutes into the race, Alice approached a section of the course lined by concrete barriers. Because new GPS measurements were far from its current state estimate, the state estimator requested and was granted a stop from supervisory control to correct approximately 4 meters of state drift. This was done and the map cleared to prevent blurred obstacles from remaining.
- GPS measurements reported large signal errors and the state estimator consequently converged very slowly, mistakenly determining that the state had converged after a few seconds. With the state estimate in an unconverged state, Alice proceeded forward.
- A considerable eastward drift of the state estimate resulted from a low confidence placed on the GPS measurements. This caused its velocity vector and yaw angle to converge to values that were several degrees away from their true values. Based on the placement of the north-south aligned row of K-rails in the map by the short-range LADAR (see fig. 4.8(a)), Alice's actual average yaw for the 5 or so seconds leading into the crash—between times C and D on fig. 4.7—appeared to be about 8 degrees west of south (-172 degrees). For the same period, its average estimated yaw was about -174 degrees and its average heading (from $\dot{N}$ and $\dot{E}$) was about -178 degrees. Roughly, as Alice drove south-southwest, its state estimate said it was driving due south, straight down the race corridor (figure 4.8(a)).
- Alice's long-range sensors detected the concrete barriers and placed them improperly in the map due to the error in state estimate. Alice's midrange sensors were still in an error mode. Alice picked up speed to 10–15 mph.

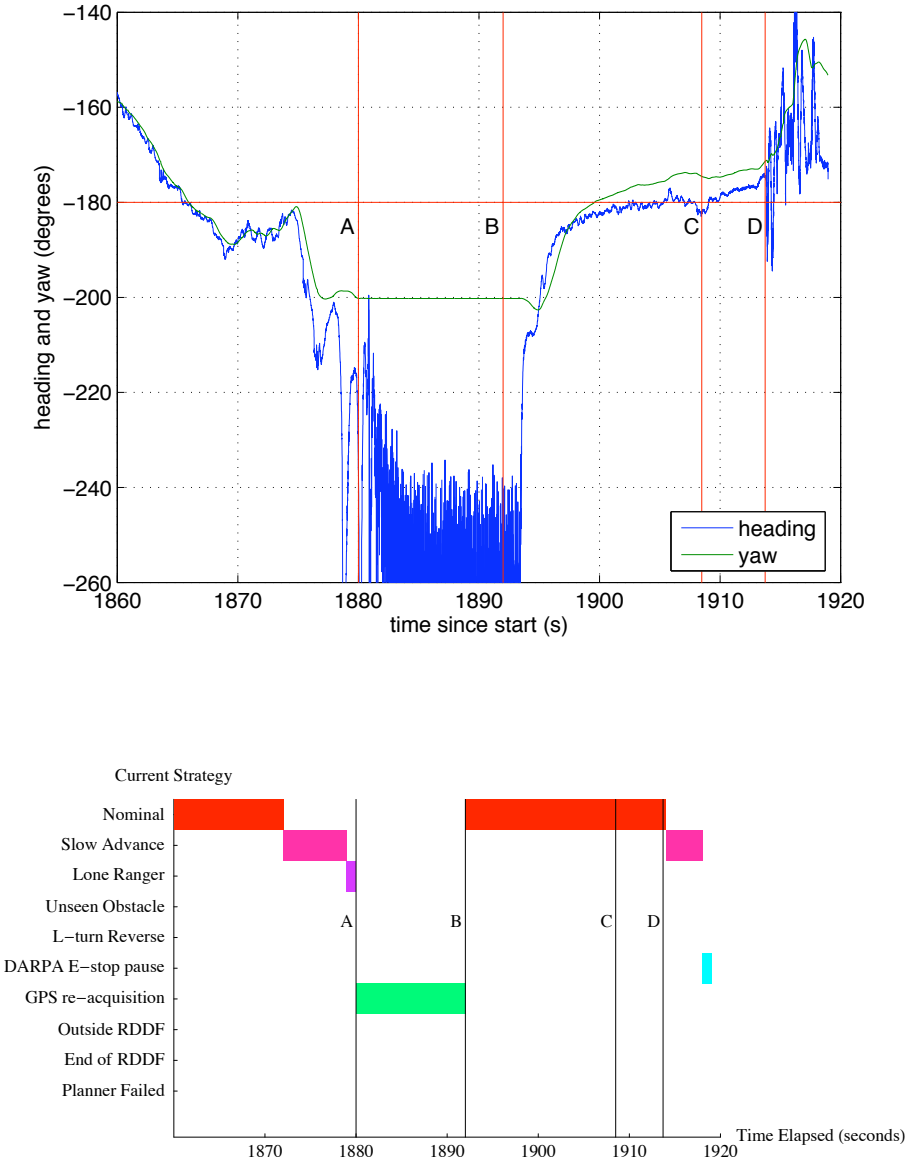

Figure 4.7. Alice's estimated heading and yaw (top), both measured clockwise from north, in its final moments of the Grand Challenge. Yaw is estimated by the state estimator and heading is computed directly as the arctangent of the easting and northing speeds of the rear axle. Shown are the approximate times at which the state estimator requested a vehicle pause (A), the map was cleared and pause was released (B), Alice sped up after clearing a region of no data (C), and impact happened (D). Between A and B the direction of motion was noisy, as expected, because Alice was stopped. Between B and D the state estimate was converged around 180 degree yaw, which we know to be about 8 degrees off, leading into the crash. Evolution of supervisory control modes (bottom). Impact speed was 6.3 m/s.

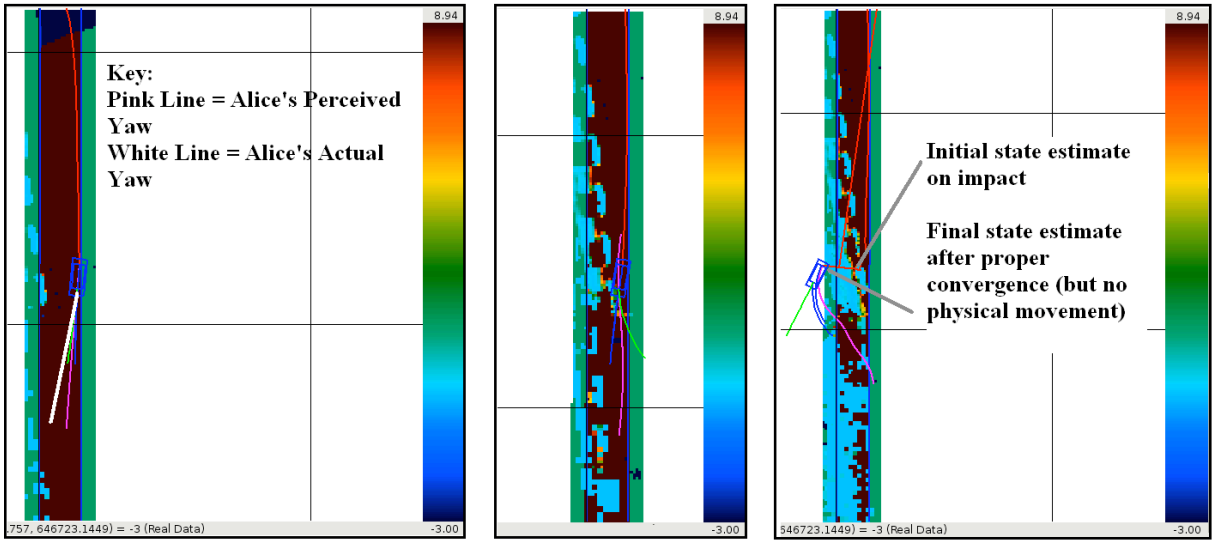

Figure 4.8. Alice's speed limit maps over the last few seconds of the race, with notation added. For all three diagrams, the brown area is the road, the light blue vertically oriented rectangular areas are the concrete barriers, the blue box is Alice, and the pink line is its perceived yaw (direction of travel). The color bar on the right indicates the speed limit that was assigned to a given cell, where brown is highest and blue is lowest. The leftmost diagram indicates Alice's expected yaw and Alice's actual yaw during the last few seconds of the race. The center diagram is Alice's speed limit map less than one second before it crashed, indicating the differing placement of the obstacles by the short- and long-range sensors as two parallel lines of concrete barriers (blue rectangles). In fact, there was only a single line of barriers oriented directly north-south, not angled as Alice perceived. The rightmost diagram is Alice's final estimate of its location: accurate, but thirty seconds too late.

Figure 4.9. Alice as it topples a concrete barrier during the Grand Challenge.

- At 32 minutes, because it was not driving where it thought it was, Alice crashed into a concrete barrier. Its short-range sensors detected the barrier, but not until it was virtually on top of it (figure 4.8(b) and figure 4.9).

- Almost simultaneously, DARPA issued an E-Stop PAUSE, the front right wheel collided with the barrier, the power steering gearbox was damaged, and the driving software detected this as a fault and executed its own pause, independent of that from DARPA. The strong front bumper prevented Alice from suffering any major damage as it drove over the barrier.

- Once Alice had come to a stop, the state-estimation software once again attempted to reconverge to get a more accurate state estimate—this time it corrected about 9.5 meters and converged close to the correct location, outside the race corridor. Alice was subsequently issued an E-Stop DISABLE command (figure 4.8(c)).

Fig. 4.7(b) shows the supervisory control component state during this final segment, including a Slow Advance through some spurious obstacles, a brief Lone Ranger push to get through them, the GPS reacquisition while stopped, and the final Slow Advance only after Alice had picked up speed and was too late to avoid crashing. These supervisory modes were described in section 4.3.9.

It is clear that while Alice's ultimate demise was rooted in its incorrect state estimates (due to poor GPS signals), other factors also contributed to its failure, or could conceivably have

prevented it. These include the midrange LADAR sensor failures, the lack of a system-level response to such failures, and the high speeds assigned to long range sensor data even in the face of state uncertainty. Additionally, in race configuration the forward facing bumper LADAR sensor was only used to assist in detection of the boundaries of the roads for the road following module. Data from this sensor could have helped to provide the persistent sensing of the row of K-rails and ensure appropriate speeds in the map.

4.5 Novel Contributions

While the design and implementation of Alice had a conventional "map, plan, follow" deliberative architecture, there are several features of Alice's architecture and implementation that distinguish it from other entries in the DARPA Grand Challenge and other autonomous vehicles. These fall into the categories of vehicle design, map representation, optimal path planning approach, and software fault tolerance.

Vehicle Platform Our platform (Sportsmobile 4x4) was extremely rugged and powerful, perhaps to a fault (since this put the vehicle on the wrong side of some K-rails at the end of our race). A central feature of its design is to satisfy the specification for rapid testing capability. Alice was equipped as a four-passenger experimental testbed and development laboratory, which enabled very quick improvement of its navigation capabilities.

Map Representation Our mapping software mapped the environment to a globally-fixed Cartesian grid, but instead of evaluating each grid cell to "go/no" or "go/no/unknown," each cell contained the maximum speed at which any portion of the vehicle could traverse that cell. This resulted in a richer representation of the environment, and a very flexible and clean abstraction between the mapping and planning levels.

Path Planning Our planning software executed a constrained nonlinear optimization problem in the space of the coefficients of a quadratic spline that represented the heading and speed as a function of distance along the desired trajectory. The speed limits encoded in the map translated into constraints in this optimization problem.

Fault Tolerance We developed an extensive supervisory control layer that handled five categories of system faults, which resulted in a vehicle that would always attempt to make (sometimes cautious) forward progress along the course. Unfortunately, the confluence of events that led to

our vehicle's crash eluded the sensibilities of this system, because the vehicle had high confidence in an erroneous state estimate.

Chapter 5

Conclusions

5.1 Summary

High-speed autonomous navigation provides a rich problem that cuts across disciplines of control theory, robotics, computer vision, and systems engineering. The navigation task can be specified in a range of ways that tunes the level and nature of autonomy achieved.

I have presented the notions of *dynamic feasibility* and *predictability* as dictating the degree of difficulty of autonomous navigation tasks, and demonstrated that these considerations have direct impact on achievable system performance. Alice is presented as a system adept at managing dynamic feasibility restrictions. Its performance in the Grand Challenge event serves to emphasize the importance of predictability in deliberative approaches to autonomous navigation.

5.2 Contributions

The major contributions of this thesis are in three areas:

Robot Control Architectures. Qualitative comparisons are made between the major methods of designing robot control architectures. These comparisons are presented in the context of dynamic feasibility and predictability, and are supported by real-world examples, thought experiments, and simulations.

Environment Modeling. Model-based methods are developed and executed in simulated and real-world experiments to demonstrate their effectiveness in high-predictability environments. Two examples are presented: (1) spatial filtering of digital elevation models by use of a high number of low-dimensional Kalman filters, and (2) detection, spatio-temporal estimation, and tracking of road centerline geometry in a desert environment. Initial headway is made

in developing and applying moving horizon estimation techniques for application to the same problem.

Alice. The detailed design and implementation of and the experimental results from Alice are presented as a demonstration of novel and advanced technical capabilities for autonomous navigation in real-world environments. Alice serves as a reference model, and the performance successes and pitfalls will serve as a guide for further advanced robotics research.

5.3 Future Work

This thesis suggests several different avenues for future work:

Formal Selection of Robot Architectures. Chapters 1 and 2 provide clarification of different robot control architectures and provide motivation and high-level guidelines for designing robot sensing and control systems based on dynamic feasibility and predictability, without formal analysis of the tradeoffs between architectures. Such an analysis could provide a structured way to design sensing, perception, and environment models, and control systems based on the specific requirements of the task and available resources.

Deliberative Control Parameterizations. When the inner-outer loop control design was implemented on Alice in the receding horizon framework, instabilities were discovered when the optimal plan was calculated using the current vehicle position as input, causing Alice to oscillate laterally and eventually drive off the road. This phenomenon happened even on straight roads where the optimal path appears to be along the centerline. Can this be recreated in simulation? What is the relative impact of various delays in the system, planning rate and miscalibration in the model? How should one decide how far into the past to use sensory data and how far into the future to plan?

Contingency Management. Alice employed a relatively sophisticated and extensively tested supervisory control layer which cross-checked a number of assumptions made during nominal operation and provided an appropriate response when these assumptions were violated. Even with such a system in place, Alice suffered a debilitating crash due to poor state estimate and the inability of the perception system to correct or compensate. More work is needed to develop those perception systems that detect discrepancies in sensor data and internal models and to provide the appropriate means to reconcile them.

From Modeling to Perception. In a wide variety of robotic tasks it is necessary to distinguish sensory data as belonging to distinct elements of the environment, such as a person walking, an object requiring manipulation, or a landmark for aiding localization. In low predictability environments this data association ability is critical to understanding of the robot's situation (and therefore critical to its response).

For example, obstruction of GPS signals by buildings in urban environments represents a situation of low predictability for the robot's position. This must be compensated by advanced perception capability (e.g., simultaneous localization and mapping or visual odometry). A combination of improved perception with model-based tracking (for high predictability targets) will provide fundamental solutions to new challenges for advanced robotics.

Dynamic Environments. One of the central limitations of many current robotic systems is that they assume static or quasistatic environments. Elements of dynamic environments (people, other cars, animals) were intentionally eliminated from the Grand Challenge because they pose a considerably more difficult navigation problem.

Current robots that operate in dynamic environments typically do so at slow speeds so that dynamic feasibility is not an issue in the face of low predictability. Reliable operation in low predictability environments with limited dynamic feasibility (e.g., at higher speeds) will provide grand challenges in robotics for years to come.

Bibliography

[1] Defense Advanced Research Projects Agency. DARPA grand challenge 2005 rules. http://www.darpa.mil/grandchallenge05/Rules2004.

[2] Yair Amir, Claudiu Danilov, Michal Miskin-Amir, John Schultz, and Jonathan Stanton. The spread toolkit: Architecture and performance. Technical Report CNDS-2004-1, Johns Hopkins University, 2004.

[3] Yair Amir and Jonathan Stanton. The spread wide area group communication system. Technical Report CNDS-98-4, The Johns Hopkins University, 1998.

[4] Ronald C. Arkin. *Behavior-Based Robotics*. The MIT Press, Cambridge, Massachusetts, 1998.

[5] Various Authors. Grand challenge 2005 team technical papers. http://www.darpa.mil/grandchallenge05/techpapers.html, December 2005. Last accessed 26 April 2006.

[6] Paolo Bellutta, Roberto Manduchi, Larry Matthies, Ken Owens, and Art Rankin. Terrain perception for demo III. In *Proc. of the Intelligent Vehicles Conference*, 2000.

[7] Valentino Braitenberg. *Vehicles: Experiments in Synthetic Psychology*. The MIT Press, 1986.

[8] Rodney Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, March 1986.

[9] Rodney A. Brooks. Intelligence without reason. In John Myopoulos and Ray Reiter, editors, *Proc. of the 12th International Joint Conference on Artificial Intelligence*, pages 569–595, Sydney, Australia, 1991. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA.

[10] Rodney A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47(47):139–159, 1991.

[11] Heiko Cramer and Gerd Wanielik. Road border detection and tracking in noncooperative areas with a laser radar system. In *Proc. of German Radar Symposium*, 2002.

[12] Lars B. Cremean, Tully B. Foote, Jeremy H. Gillula, George H. Hines, Dmitriy Kogan, Kristopher L. Kriechbaum, Jeffrey C. Lamb, Jeremy Leibs, Laura Lindzey, Alexander D. Stewart, Joel W. Burdick, and Richard M. Murray. Alice: An information-rich autonomous vehicle for high-speed desert navigation. *Submitted, Journal of Field Robotics*, 2006.

[13] Ernst D. Dickmanns and Birger D. Mysliwetz. Recursive 3-d road and relative ego-state recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):199–213, 1992.

[14] Magnus Egerstedt, Xiaoming Hu, Henrik Rehbinder, and A. Stotsky. Path planning and robust tracking for a car-like robot. In *Proc. of the 5th Symposium on Intelligent Robotic Systems*, pages 237–243, Stockholm, Sweden, July 1997.

[15] Ryan Franz, Mark Milam, and John Hauser. Applied receding horizon control of the Caltech ducted fan. In *Proc. of the American Controls Conference*, 2002.

[16] Carlos E. Garcia, David M. Prett, and Manfred Morari. Model predictive control: Theory and practice—a survey. *Automatica*, 25(3):335–348, 1989.

[17] Brian Gerkey, Richard T. Vaughan, and Andrew Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proc. of the 11th International Conference on Advanced Robotics*, pages 317–323, Coimbra, Portugal, June 2003.

[18] Steven B. Goldberg, Mark W. Maimone, and Larry Matthies. Stereo vision and rover navigation software for planetary exploration. In *Proc. of IEEE Aerospace Conference*, volume 5, pages 2025–2036, 2002.

[19] Mohinder S. Grewal, Lawrence R. Weill, and Angus P. Andrews. *Global Positioning Systems, Inertial Navigation, and Integration*. Wiley-Interscience, 2000.

[20] David S. Hall and Bruce H. Hall. Team dad technical paper. Technical report, DARPA, 2005.

[21] J. Sean Humbert. *Bio-Inspired Visuomotor Convergence in Navigation and Flight Control Systems*. Ph.D. thesis, California Institute of Technology, 2005.

[22] Alonzo Kelly and Anthony Stentz. An analysis of requirements for rough terrain autonomous mobility. *Autonomous Robots*, 5(2):129–161, December 1997.

[23] Alonzo Kelly and Anthony Stentz. Rough terrain autonomous mobility—part 2: An active vision, predictive control approach. *Autonomous Robots*, 5(2):163–198, May 1998.

[24] Deepak Khosla. Accurate estimation of forward path geometry using two-clothoid road model. In *Proc. of IEEE Intelligent Vehicles Symposium*, pages 154–159, 2002.

[25] Alexander Kirchner and Thomas Heinrich. Model based detection of road boundaries with a laser scanner. In *Proc. of IEEE Internatonal Conference on Intelligent Vehicles*, pages 93–98, 1998.

[26] William Klarquist and James McBride. Intelligent vehicle safety technologies 1 technical description. Technical report, DARPA, 2005.

[27] Dmitriy Kogan. Realtime path planning via nonlinear optimization methods. Master's thesis, California Institute of Technology, 2005.

[28] Simon Lacroix, Anthony Mallet, David Bonnafous, Grard Bauzil, Sara Fleury, Matthieu Herrb, and Raja Chatila. Autonomous rover navigation on unknown terrains: Functions and integration. *International Journal of Robotics Research*, 21(10-11):917–942, Oct-Nov. 2002.

[29] Bill Lincoln, Alberto Elfes, Guillermo Rodriguez, and Charles Weisbin. Relative benefits of potential autonomy technology investments. In *International Conference on Space Mission Challenges for Information Technology*, Pasadena, California, 2003.

[30] William Lincoln, Ayanna Howard, Guillermo Rodriguez, Ramachandra Manvi, Charles Weisbin, and Mark Drummond. A new method to determine impact of autonomy technologies on NASA space exploration missions. Technical report, Jet Propulsion Laboratory, 2003.

[31] Mark W. Maimone and Jeffrey J. Biesiadecki. The mars exploration rover surface mobility flight software: Driving ambition. In *Proc. of 2006 IEEE Aerospace Conference*, Big Sky, Montana, 2006.

[32] Roberto Manduchi, Andres Castano, Ashit Talukder, and Larry Matthies. Obstacle detection and terrain classification for autonomous off-road navigation. *Autonomous Robots*, 18:81–102, May 2005.

[33] Martin C. Martin and Hans Moravec. Robot evidence grids. Technical Report CMU-RI-TR-96-06, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, March 1996.

[34] Maja J. Matarić. Behavior-based control: Main properties and implications. In *Proc. of the IEEE International Conference on Robotics and Automation*, Nice, France, 1992.

[35] Maja J. Matarić. Integration of representation into goal-driven behavior-based robots. *IEEE Transactions on Robotics and Automation*, 8(3):304–312, June 1992.

[36] Maja J. Matarić. Situated robotics. In *Encyclopedia of Cognitive Science*. Nature Publishing Group, Macmillan Reference Limited, 2002.

[37] David Q. Mayne and Hanna Michalska. Receding horizon control of nonlinear systems. *IEEE Transactions on Automatic Control*, 35(7):814–824, 1990.

[38] David Q. Mayne, James B. Rawlings, Christopher V. Rao, and Pierre O. M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

[39] Mark B. Milam. *Real-time optimal trajectory generation for constrained dynamical systems*. Ph.D. thesis, California Institute of Technology, 2003.

[40] Mark B. Milam, Kudah Mushambi, and Richard M. Murray. A computational approach to real-time trajectory generation for constrained mechanical systems. In *Proc. of Conference on Decision and Control*, 2000.

[41] Michael Montemerlo and Sebastian Thrun. A multi-resolution pyramid for outdoor robot terrain perception. In *Proc. of the AAAI National Conference on Artificial Intelligence*, San Jose, CA, 2004. AAAI.

[42] Richard M. Murray, John Hauser, Ali Jadbabaie, Mark B. Milam, Nicolas Petit, William B. Dunbar, and Ryan Franz. Online control customization via optimization-based control. In T. Samad and G. Balas, editors, *Software-Enabled Control: Information Technology for Dynamical Systems*. IEEE Press, 2003.

[43] James A. Primbs. *Nonlinear Optimal Control: A Receding Horizon Approach*. Ph.D. thesis, California Institute of Technology, 1999.

[44] Christopher Rasmussen. Grouping dominant orientations for ill-structured road following. In *Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 470–477, 2004.

[45] Christopher Rasmussen. A hybrid vision + ladar rural road follower. In *Proc. of IEEE International Conference on Robotics and Automation*, 2006.

[46] Robert D. Rasmussen. Goal-based fault tolerance for space systems using the Mission Data System. In *Proc. of IEEE Aerospace Conference*, 2001.

[47] Julio K. Rosenblatt. Utility fusion: Map-based planning in a behavior-based system. In *Proc. of International Conference on Field and Service Robotics*, pages 411–418. Springer-Verlag, 1997.

[48] Julio K. Rosenblatt. Behavior-based planning for intelligent autonomous vehicles. In *Proc. of Symposium on Intelligent Autonomous Vehicles*, Madrid, Spain, 1998.

[49] Anthony Stentz, Alonzo Kelly, Peter Rander, Herman Herman, Omead Amidi, Robert Mandelbaum, Garbis Salgian, and Jorgen Pedersen. Real-time, multi-perspective perception for unmanned ground vehicles. In *Proc. of AUVSI Unmanned Systems Symposium*, 2003.

[50] Sebastian Thrun. Particle filters in robotics. In *Proc. of the 17th Annual Conference on Uncertainty in Artificial Intelligence*, 2002.

[51] Sebastian Thrun. Robotic mapping: A survey. In *Exploring Artificial Intelligence in the New Millenium*, pages 1–35. Morgan Kaufmann Publishers, San Francisco, CA, USA, 2003.

[52] Chris Urmson. *Navigation Regimes for Off-Road Autonomy*. Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA, USA, November 2004.

[53] Chris Urmson, Joshua Anhalt, Michael Clark, Tugrul Galatali, Juan P. Gonzalez, Jay Gowdy, Alexander Gutierrez, Sam Harbaugh, Matthew Johnson-Roberson, Yu Hiroki Kato, Phillip Koon, Kevin Peterson, Bryon Smith, Spencer Spiker, Erick Tryzelaar, and William Red Whittaker. High speed navigation of unrehearsed terrain: Red team technology for grand challenge 2004. Technical Report CMU-RI-TR-04-37, Carnegie Mellon University, The Robotics Institute 5000 Forbes Avenue Pittsburgh, PA 15213, June 2004.

[54] D. J Walton and D. S. Meek. Computer-aided design for horizontal alignment. *Journal of Transportation Engineering*, 115(4):411–424, July 1989.

[55] Greg Welch and Gary Bishop. An introduction to the Kalman filter. Technical Report TR 95-041, University of North Carolina, 1995.

[56] Ross T. Whitaker and Jens Gregor. A maximum-likelihood surface estimator for dense range data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(10):1372–1387, October 2002.

[57] Ross T. Whitaker and Ernesto L. Juarez-Valdes. On the reconstruction of height functions and terrain maps from dense range data. *IEEE Transactions on Image Processing*, 11(7):704–716, July 2002.

[58] W. Sardha Wijesoma, K. R. Sarath Kodagoda, and Arjuna P. Balasuriya. Road-boundary detection and tracking using ladar sensing. *IEEE Transactions on Robotics and Automation*, 20(3):456–464, 2004.

[59] Zhengyou Zhang. A stereovision system for a planetary rover: Calibration, correlation, registration, and fusion. *Machine Vision and Applications*, 10(1):27–34, 1997.