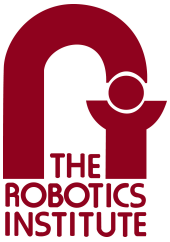
ADAPTIVE MODEL-PREDICTIVE MOTION PLANNING FOR NAVIGATION IN COMPLEX ENVIRONMENTS

THOMAS M. HOWARD

CMU-RI-TR-09-32

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Robotics



The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

August 2009

Thesis Committee

Alonzo Kelly, Co-Chair
Red Whittaker, Co-Chair
Tony Stentz
Steven Dubowsky, Massachusetts Institute of Technology

II

ABSTRACT

Outdoor mobile robot motion planning and navigation is a challenging problem in artificial intelligence. The search space density and dimensionality, system dynamics and environmental interaction complexity, and the perceptual horizon limitation all contribute to the difficultly of this problem. It is hard to generate a motion plan between arbitrary boundary states that considers sophisticated vehicle dynamics and all feasible actions for nontrivial mobile robot systems. Accomplishing these goals in real time is even more challenging because of dynamic environments and updating perception information.

This thesis develops effective search spaces for mobile robot trajectory generation, motion planning, and navigation in complex environments. Complex environments are defined as worlds where locally optimal motion plans are numerous and where the sensitivity of the cost function is highly dependent on state and motion model fidelity. Examples include domains where obstacles are prevalent, terrain shape is varied, and the consideration of terramechanical effects is important.

Three specific contributions are accomplished. First, a model-predictive trajectory generation technique is developed that numerically linearizes and inverts general predictive motion models to determine parameterized actions that satisfy the two-point boundary value problem. Applications on a number of mobile robot platforms (including skid-steered field robots, planetary rovers with actively articulating chassis, mobile manipulators, and autonomous automobiles) demonstrate the versatility and generality of the presented approach. Second, an adaptive search space is presented that exploits environmental information to maintain feasibility and locally optimize the mapping between nodes and states. Sequential search in the relaxed motion planning graph is shown to produce better (shorter/faster/lower-risk) trajectories in dense obstacle fields without modifying the graph topology. Results demonstrate that a coarse, adaptive search space can produce better solutions faster than dense, fixed search spaces in sufficiently complex environments. Lastly, a receding-horizon model-predictive control method that exploits structure from sequential search to determine trajectory following actions is presented. The action space is parameterized by the regional motion plan and subsequently relaxed through unconstrained optimization. Examples are shown to effectively navigate intricate paths in a natural environment while maintaining a constant horizon.

ACKNOWLEDGEMENTS

Graduate school is an odyssey that nobody pursues alone. I have many to acknowledge for my growth, development, and experiences over the past five years. First I would like to thank my two advisors, Al Kelly and Red Whittaker. I am very fortunate to have worked with two faculty members that provided me the tools, guidance, and opportunities to pursue research topics that I am genuinely passionate about. Al taught me how to be an effective researcher. His patient technical instruction helped me quickly establish a research foundation from which I was able to explore real-world robotics applications. He provided research challenges that helped me push the boundaries of my own capabilities. I owe a great deal of my success to his guidance throughout the Ph.D. program. Red greatly shaped my ideology for how robots can interact with society. Working with him helped me to understand what it takes to tackle *big* challenges. His counsel on effective communication, articulate expression, and following through on intentions has shaped me as a robotocist and a person. I would also like to thank Tony Stentz and Steven Dubowsky for their solid guidance, thoughtful advice, and diligent service as members of my thesis committee.

The Robotics Institute is an incredible and unique place because of its people. I would like to thank my officemates Ross Knepper, Colin Green, Dean Anderson, Mihail Pivtorakio, Dave Bradley, Dave Silver, Jean-Francios Lalonde, and other classmates for their advice, support, and collaboration over the past five years. In particular, Ross, Colin, and Mihail provided advice and insight to the trajectory generation, motion planning, and navigation research presented in this thesis. The RI staff, particularly Michele Gittleman, Suzanne Lyons-Muth, Cindy Glick, and Heather Farah, helped provide order to the seemingly chaotic nature of graduate school.

I would like to thank the numerous mentors and co-workers I have been fortunate to work with during my research. Issa Nesnas, Hari Nayar, Antonio Diaz-Calderon, and others from the NASA Jet Propulsion Laboratory introduced me to the space robotics domain. Mike Bode, Tom Pilarski, and the UGCV-Perceptor Integrated team taught me how to contribute and apply my work to a large mobile robot system. Herman Herman provided guidance and leadership that enabled success on the Autonomous Coordinated Mobility and Manipulation project. Constantine Domashnev was a great resource for software and hardware systems throughout my work. I learned invaluable lessons on collaboration, communication, and teamwork working with Dave Ferguson, Max Likhachev, Bryan Salesky, Chris Urmson, and others on the DARPA Urban Challenge. Kevin Peterson, Nick Miller, Vanessa Hodge, and Matt Johnson-Roberson patiently taught me a great deal about autonomous mobile robots early in my program through the preparations for the DARPA Grand Challenge. Michael Furlong and David Wettergreen helped me explore and extend my research for mobile robots with actively articulating chassis.

This thesis is dedicated to my parents, John and Gail, who have unconditionally supported me through this journey. Without their constant encouragement, this would have undoubtedly been a much more difficult and isolating endeavor. I also owe gratitude to my siblings Joe and Kate, who have always championed my passions, pursuits, and interests.

Finally, I would like to thank the person who supported me most through this challenging experience, my wife Vicki. In addition to patiently dealing with the quirks of living with a robotocist, her humor, compassion, friendship, and love have made the past few years the best of my life.

Portions of this work were made possible through several grants. The model-predictive trajectory generation and receding horizon model-predictive control research were conducted at the Robotics Institute of Carnegie Mellon University under contract to NASA/JPL as part of the Mars Technology Program. The state-space sampling navigation and receding horizon model-predictive controller experiments were sponsored by the DARPA Advanced Research Projects Agency (DARPA) under contract Unmanned Ground Combat Vehicle PerceptOR Integration (contract number MDA972-01-9-0005). Additional experiments in state-space sampling navigation were supported by Tartan Racing through sponsors including General Motors, Caterpillar, Continental, and DARPA under contract HR0011-06-C-0142. The views and conclusions contained in this paper are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government.

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Outdoor mobile robot motion planning and navigation is a challenging problem in artificial intelligence. The search space dimensionality, system dynamics and environmental interaction complexity, and the perceptual horizon limitation all contribute to the difficultly of this problem. It is hard to generate a motion plan between arbitrary boundary states that considers sophisticated vehicle dynamics and all feasible actions for nontrivial mobile robot systems. Accomplishing these goals in real time is even more challenging because of dynamic environments and updating perception information.

This thesis develops effective search spaces for mobile robot trajectory generation, motion planning, and navigation in complex environments. Complex environments are defined as worlds where locally optimal motion plans are numerous and where the sensitivity of the cost function is highly dependent on state and motion model fidelity (Figure 1.1). Examples include domains where obstacles are prevalent, terrain shape is varied, and the consideration of terramechanical effects is important.

Particular challenges addressed in this thesis include feasibility, optimality, and efficiency. Motion plan feasibility is important in complex environments because small deviations from the planned trajectory can be extremely hazardous. Optimality is particularly challenging because approximating the path continuum with a search space is extremely difficult. Numerous locally optimal, homotopically distinct solutions exist in complex, cluttered environments. Typically, dense search in action and state space is required to reliably generate effective motion planning search spaces. Efficiency is important as limited computational resources such as processor time and memory are available in mobile robot applications. Developing smart sampling techniques are crucial to developing efficient mobile robot motion planners and navigators that can more intelligently consume computational resources.

1.1 PROBLEM STATEMENT

There are many well developed and applied techniques for mobile robots to search the path continuum in difficult and challenging environments. Most techniques fall into one of two categories: sequential search and continuum optimization. Sequential search processes typically provide solutions that approximate the global optimum but are limited to search only graph edges and satisfy discretized state constraints. Optimization and relaxation techniques determine locally optimal actions in the continuum, but are difficult to initialize because homotopically distinct solutions can be numerous. Such techniques are arguably more informed and efficient as they follow the gradients of the cost functions to optimize trajectories. However, such techniques lack the global optimality guarantees of sequential search and often

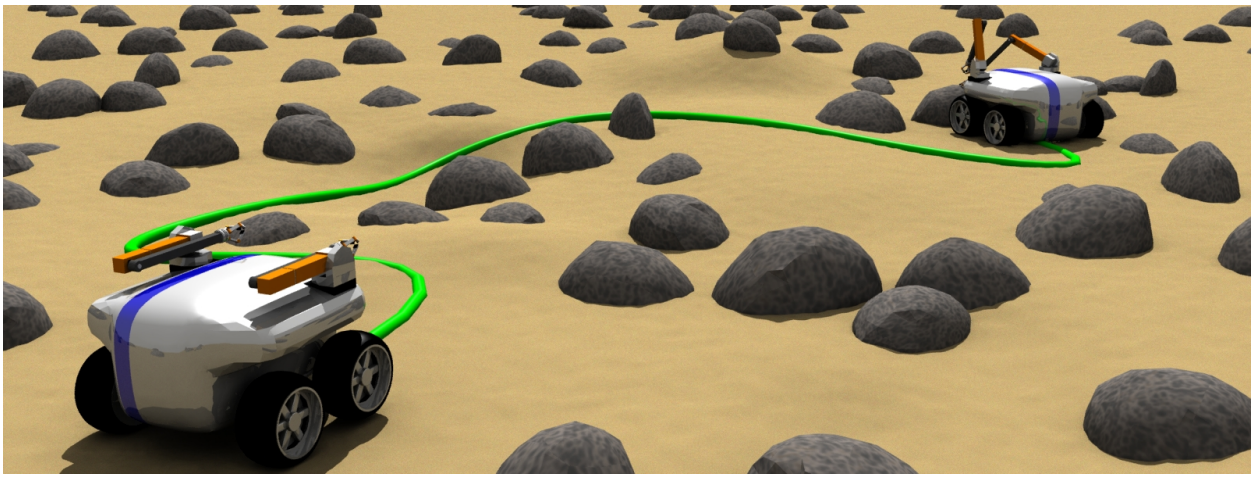


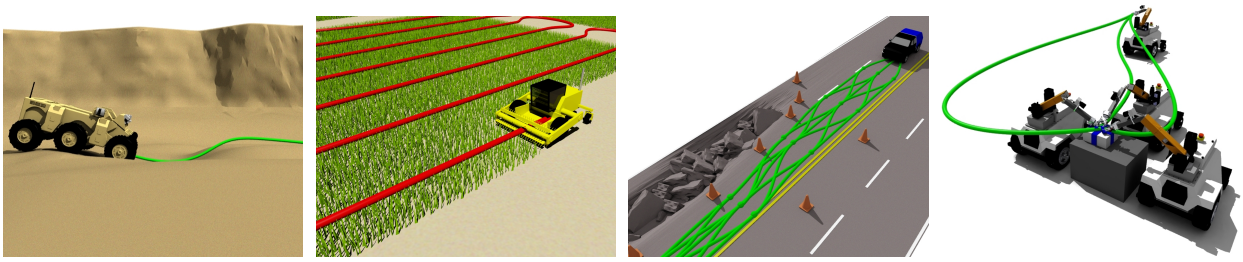
Figure 1.1: A mobile robot motion planning and navigation problem in a complex environment. In order to reach the target terminal state to deploy its instruments, a mobile robot must plan and navigate the sequence of actions which will lead the robot to its goal. This thesis is generally concerned with the problem of generating feasible, minimum-risk, and efficient motion plans in such complex environments.

cannot pass over boundaries that define homotopically distinct paths.

A potentially better solution is to leverage the benefits of each approach by relaxing sequential search spaces in the continuum and exploiting the structure of regional motion plans for continuum optimization. First, sequential search spaces could be relaxed to conform to the environment to improve the relative optimality of solutions produced by the motion planning graph. Second, local search techniques for mobile robot navigation could be informed by the global guidance information, generating actions that are parameterized and initialized by the regional motion plan. In order to achieve this and maintain feasibility across all levels of motion planning, navigation, and control, efficient real-time processes are needed for informed trajectory generation to determine actions that consider sophisticated, general models of motion, suspension, and environmental interaction.

1.2 APPLICATIONS

The contributions of this thesis are relevant to any system where the system dynamics and terrain interaction are non-negligible, the operational environment of the system is complex, or where achieving the safest and minimum-cost paths are of the greatest concern. Of particular interest are several current of mobile robot applications where motion planning and navigation challenges in complex environments is common, including field robotics, exploration robotics, agricultural and mining systems, autonomous automobiles, and mobile manipulators.



(a) field robotics (b) agricultural robotics (c) autonomous automobiles (d) mobile manipulators

Figure 1.2: Several applications where feasible, optimal, and efficient mobile robot motion planning techniques are important.

FIELD ROBOTICS

Field mobile robots are platforms particularly suited for operation in outdoor, rough, and unstructured environments. For many systems, reasoning about the terrain interaction is important. Exposure to hazardous tip-over conditions and estimating traversability effectively can mean the difference between vehicles successfully completing task objectives and returning home or mission failure and platform loss.

Terrain navigation is a common problem in field robotics (Figure 1.2a). Typical solutions to this problem involve engineered cost functions based on prior data for a particular platform. As the mobility model changes over the mission, the traversability of the environment may change, rendering the predetermined cost function ineffective. If the estimate was too conservative, the robot may become stuck and pause or reroute through a different area. Conversely if it was not conservative enough, the robot may try to navigate through a terrain feature that may risk safety of the platform. The ability to generate informed search spaces and trajectories for mobile robot motion planning enables field robots to better reason about their environmental interaction in difficult terrains autonomously.

EXPLORATION ROBOTICS

Mobile robots designed for the investigation, examination, and exploration of dangerous and/or remote environments particularly values safety and reliability over speed and performance, as fixing or rescuing these systems may not be feasible. Modern examples include planetary rovers for lunar and Martian exploration (Maimone et al., 2004), autonomous underwater vehicles for Cenote mapping (Fairfield et al., 2007), unmanned aerial vehicles for natural disaster survey (Pratt et al., 2009), and skid-steered field robots for mine inspection (Morris et al., 2006). Determining and executing the minimum-risk path is most important when the mission objectives are focused on the scientific benefits and results instead of the mobile robot performance.

One particular challenge for exploration robotics is that the system must deal with potentially heterogeneous and

unexpected environments. The motion planning, navigation, and control system for such platforms may need to compensate for denser obstacle fields and more challenging terrain than originally expected. A motion planning system designed for operation in a open plain may arrive in a cluttered environment and be unable to determine any action that navigates through the environment. Since energy is limited in many robotic exploration applications, stopping to determine the shortest path through the environment may be better than determining the longer, less optimal path while moving. Conversely, a system designed to produce actions in dense obstacle fields can be very inefficient in generally open terrain. The development of motion planners that could better reason about the environment leads to systems that operate more effectively in heterogeneous environments.

MINING AND AGRICULTURAL ROBOTICS

Resource extraction automation will revolutionize the mining and agriculture industries in the next century. Autonomous vehicles will lead to more stable, efficient, and cost effective platform design. Resources could be harvested in dangerous, remote environments including steep mountainside terraces, the ocean floor, asteroids, and the Moon.

Motion planning, navigation, and control algorithms influence the rate at which resources can be extracted from the environment. Mobile robots that better follow natural resource contours makes autonomous systems more profitable. Autonomous agricultural systems that generate minimum-length motion plans, satisfy mobility and environmental constraints, and extract resources at a faster rate can decrease overhead costs by reducing fuel consumption and maintenance. Figure 1.2b shows trajectory generation and motion planning for a thresher which lines the end effector up with the next row of crops. Reasoning about the kinematic constraints of the vehicle is important when the turning radius may be limited, since the vehicle may have to initially turn away from the intended target to achieve the desired state. Likewise, autonomous agricultural and mining systems often repeat similar motion plans in the same environment many times. By progressively adapting the search space to conform exactly to the particular environment, the optimality of the resulting motion plans is likely to increase, resulting in potentially faster, better, and cheaper systems.

AUTONOMOUS AUTOMOBILES

The automation of our streets and highways by robotic automobiles leads to safer and more efficient transportation systems for humans. To achieve this, developments in perception, localization, mapping, communication, motion planning, and control must occur. Motion plan feasibility is important at the speeds associated with highway navigation. Reasoning about kinematic and dynamic constraints while performing swerves, lane changes, or simple lane following is necessary to maintain vehicle safety. Planning infeasible actions and relying on lower-level path followers and obstacle avoidance algorithms can be disastrous if the difference between the planned and actual actions is significant.

Reliability of mobile robot systems is related to feasibility. Reliable systems must be able to also consider changes in their own mobility and adapt accordingly. Informed navigators that reason about situations including blown tires, poor weather, and engine failure can determine the necessary actions required to recover in environments unforgiving of path follow error. An autonomous automobile that can reason about significant changes in mobility and appropri-

ately modify its driving technique will be safer and more reliable.

In addition to changes in vehicle mobility, reasoning about abrupt or unexpected changes in road shape (Figure 1.2c) will require navigators to adjust the shape of the constructed motion planning graphs. Techniques that exploit environmental information to locally adjust graph structure can lead to motion planners that more efficiently sample the space of actions.

MOBILE MANIPULATORS

The fields of autonomous construction, mining, and explosive ordinance demolition all require mobile robots to not only move from one place to another, but interact with and manipulate the environment. Motion planning problems for such systems are typically under-determined problems (since the platform has more freedoms than constraints) and are typically handled as decoupled problems. More effective motion planners that reason about the combined mobility of the motion and manipulation freedoms will require new techniques that effectively search the space of feasible actions and judiciously sample in the high dimensional action or state space.

1.3 APPROACH

This thesis presents an adaptive model-predictive approach to mobile robot search space generation. The technique modifies the underlying state mapping and transition equations without graph topology modification to produce search spaces that are more efficient and effective in complex environments. Adaptation comes in two forms.

First, the mapping between continuous states and discretized nodes in a graph is locally optimized to minimize edge costs. Local graph optimization leads to motion planning graphs that exploit environmental information to produce minimum-cost paths in environments where obstacles are dense, and locally optimal, homotopically distinct solutions are prevalent. Exploiting environmental information also leads to motion planning graphs that more intelligently consume resources by reducing the necessary state and action space sampling. A progressively adaptive variation of this technique is presented that adjusts the search space incrementally for faster performance and perception update handling. Second, graph edge connectivity is reevaluated based on environmental information to better reason about vehicle mobility. Feasible motion planning graphs are important because they represent more reliable estimates of true path cost.

The development and application of an adaptive model-predictive search space requires two additional advances. This thesis presents a model-predictive trajectory generation technique to efficiently determine actions that reason about general mobility, configuration, constraints, and optimality. A model-predictive trajectory generator is necessary to maintain feasibility while modifying the motion planning graph. A capability to follow the intricate paths is also necessary to utilize the paths generated for complex environments. This thesis presents a receding horizon model-predictive controller that is an extension of unconstrained optimization model-predictive trajectory generation that is parameterized and initialized by the structure of a regional motion plan. In the same manner that continuum optimization is used to relax motion planning graphs for sequential search, global motion plan information is used

to parameterize and initialize a model-predictive controller to effectively navigate intricate paths in complex environments.

1.4 THESIS STATEMENT

This thesis advocates that more effective motion plans can be generated by combining sequential search and continuum optimization techniques for mobile robot motion planning, navigation, and control:

"Model-predictive trajectory generation and relaxation can be utilized in motion planning, navigation, and control to increase motion plan optimality while reducing sampling, inform actions to better reason about the environment, and effectively navigate intricate paths in complex environments."

1.5 ASSUMPTIONS

This thesis makes several assumptions to limit the scope of this research:

- This research assumes a priori knowledge of the environment, both in terrain shape and underlying wheel/terrain models. The problem of observing and identifying the shape and soil properties of a mobile robot's surroundings is an important problem beyond the scope of this research.
- This research assumes a known predictive motion model. It does not consider the difficult problem of on-line system identification.
- This research advances the state of the art in regional motion planning, but does not advocate solutions for long range (kilometer scale) motion planning problems.

1.6 ORGANIZATION

This dissertation is organized as three sequential blocks of research following the problem of model-predictive trajectory generation, motion planning, and navigation for mobile robots in complex environments. First, terminology, concepts, and challenges addressed in this thesis are described in Chapter 2. Related work in mobile robot motion planning, navigation, and control follow this discussion in Chapter 3. The first contribution of this thesis, model-predictive trajectory generation and optimization, is discussed in Chapter 4. In this section, variants of the model-predictive trajectory generation algorithm and examples for a series of platforms are presented. The main contribution of this thesis, adaptive model-predictive search spaces, is presented in Chapter 5 for the problem of motion planning in complex environments. Chapter 6 formulates the navigation challenge as an optimal control problem and presents a receding-horizon model-predictive control algorithm as an extension of unconstrained optimization model-predictive trajectory

generation. This technique is intended to replace multi-modal path following techniques for high-performance intricate path navigation. The thesis concludes with a summary and a discussion of the contributions and future directions of this work in Chapter 7.

1.7 PUBLICATION NOTE

Portions of the model-predictive trajectory generation research in Chapter 4 for mobile robots operating on rough terrain appear in Howard and Kelly (2005a,b, 2007). The impaired mobility field experiments are discussed in (Pivtoraiko et al., 2008). The application of model-predictive trajectory generation for path following control and state-space sampling navigation is presented in Howard and Kelly (2006); Howard et al. (2008); Urmson et al. (2008). Extensions of this work for mobile manipulators and mobile robots with articulating suspension appear in Anderson et al. (2008) and Furlong et al. (2009) respectively. The development and application of a receding horizon model-predictive controller for intricate path navigation in Chapter 6 is discussed in Howard et al. (2009).

CHAPTER 2

PROBLEM DESCRIPTION

This section serves several purposes. Section 2.1 describes basic terminology used throughout the document. Section 2.2 reviews several core concepts that are fundamental to understanding the combined problem of motion planning, navigation, obstacle avoidance, and control. Lastly, Section 2.3 describes the fundamental challenges in motion planning, navigation, and control for mobile robots in complex environments that are addressed by this thesis.

2.1 TERMINOLOGY

This section describes important terminology used throughout the thesis. Specifically, it reviews the definitions of states, actions, trajectories, trajectory generation, motion planning, and navigation.

STATE

The state ($\mathbf{x}$) is a representation of a mobile robot's location, movement, and configuration in the world. Typically the state is defined as a vector containing the vehicle's position (x, y, z), orientation (ϕ, θ, ψ), linear and angular velocities $(v_x, v_y, v_z, \omega_x, \omega_y, \omega_z)$, curvature (κ), or any other quantities of interest.

$$\mathbf{x} = \begin{bmatrix} x & y & z & \psi & \theta & \phi & v_x & v_y & v_z & \omega_x & \omega_y & \omega_z & \kappa & \dots \end{bmatrix}^T \tag{2.1}$$

States are used in this thesis to represent the initial, current, or target terminal positions, orientations, velocities, and configurations of a vehicle. Throughout this document, boundary states will be defined as the initial state ($\mathbf{x_I}$) and target terminal state ($\mathbf{x_T}$).

ACTION

In mobile robot motion planning, navigation, and control, actions are defined as system inputs that modify the vehicle state as a function of time (t). Two common types of actions are controls ($\mathbf{u_{control}}(\mathbf{t})$) and controllers ($\mathbf{u_{controller}}(\mathbf{x}, \mathbf{t})$). A control determines actions as a function of time, whereas controllers determines inputs as a function of state and time:

$$\mathbf{u}_{\mathbf{control}}(t) = \mathbf{f}(t) \tag{2.2}$$

$$\mathbf{u}_{\mathbf{controller}}(\mathbf{x},t) = \mathbf{f}(\mathbf{x},t) \tag{2.3}$$

Inputs to a mobile robot system are platform and implementation specific. A wheeled mobile robot can be controlled directly in the input space of wheel torques or at the higher level of body-frame linear and angular velocities, where wheel torques are computed based on the current state and rigid-body motion. Examples of controls and controllers are further discussed in Section 4.2.

Parameterized controls and controllers determine actions as a function of a variable vector in addition to time and/or state. Parametric representations, including clothoids, cubic splines, and Bezier splines, have been effectively used in mobile robot navigation and control (Nagy and Kelly, 2001) as they represent an expressive set of geometric functions using minimal degrees of freedom:

$$\mathbf{u}_{\mathbf{control}}(\mathbf{p},t) = \mathbf{f}(\mathbf{p},t) \tag{2.4}$$

$$\mathbf{u}_{\mathbf{controller}}(\mathbf{p},\mathbf{x},t) = \mathbf{f}(\mathbf{p},\mathbf{x},t) \tag{2.5}$$

Throughout this thesis, $\mathbf{u}(\mathbf{x},t)$ will represent an action evaluated as a control or controller. Likewise, $\mathbf{u}(\mathbf{p},\mathbf{x},t)$ will represent a parameterized action evaluated as a control or controller.

TRAJECTORY

A state-space trajectory is defined as a vector valued function of state ($\mathbf{x}$) or time (t). Time is the most typical trajectory representation because time is monotonically increasing and always defined. Other state elements such as distance (s) or heading (ψ) can be useful to represent vehicle motion through space as they allow velocities to remain unspecified.

Trajectories are used throughout this document because paths, represented as a sequence of states, can be discontinuous for particular sequences of actions. A cusp is a point in a trajectory where linear velocity changes sign. While cusps are discontinuous in path curvature, they are not discontinuous in state space trajectories and are perfectly feasible actions. Similarly, a turn-in-place is a path point where the heading changes discontinuously, however a turn-in-place is continuous in the state-space trajectory representation. A state-space trajectory differs from a geometric path representation because it stores the full state of the vehicle at each time step along the action.

In the context of mobile robots, trajectories are determined by integrating sets of functions that represent vehicle motion rates. For a vehicle starting from an initial location, orientation, and configuration ($\mathbf{x_I}$), the trajectory can be determined by numerical integration of the equations of motion ($\mathbf{f_{PMM}}(\mathbf{x},\mathbf{u},t)$) with an initial condition:

$$\dot{\mathbf{x}}(\mathbf{x},\mathbf{u},t) = \mathbf{f_{PMM}}(\mathbf{x}(t),\mathbf{u}(\mathbf{x},t),t) \tag{2.6}$$

$$\mathbf{x}(t_F) = \mathbf{x_I} + \int_{t_I}^{t_F} \dot{\mathbf{x}}(\mathbf{x}(t),\mathbf{u}(\mathbf{x},t),t)\,dt = \mathbf{x_I} + \int_{t_I}^{t_F} \mathbf{f_{PMM}}(\mathbf{x}(t),\mathbf{u}(\mathbf{x},t),t)\,dt \tag{2.7}$$

The predictive motion model cannot be integrated in closed form for many non-trivial mobile robot systems. Numerical methods such as Euler or multi-step Runge-Kutta must be used to determine the state response to the action. Balancing the fidelity of the predictive motion model simulation with runtime performance is one of the most challenging aspects of model-predictive control. Throughout this work, a *reference trajectory* will refer to the solution determined by a motion planner or other higher-level guidance methods.

TRAJECTORY GENERATION

Mobile robot trajectory generation is the problem of determining actions that satisfy continuum boundary state and vehicle mobility constraints and/or optimize a utility function. The two-point boundary state constraint problem is generally defined as determine the set of actions that satisfies the following set of equations:

$$\dot{\mathbf{x}}(\mathbf{x},\mathbf{u},t) = \mathbf{f_{PMM}}(\mathbf{x}(t),\mathbf{u}(t),t) \tag{2.8}$$

$$\mathbf{x}(t_I) = \mathbf{x_I} \tag{2.9}$$

$$\mathbf{x}(t_F) = \mathbf{x_I} + \int_{t_0}^{t_f} \dot{\mathbf{x}}(\mathbf{x},\mathbf{u}(t),t)\,dt \tag{2.10}$$

The constraint defined by Equation 2.9 is typically satisfied implicitly, as no action may change the vehicle position, orientation, or configuration at the beginning of the trajectory. Equation 2.10 is satisfied by determining the actions ($\mathbf{u}(\mathbf{x},t)$), which when combined with the predictive motion model and the initial state constraint, achieve the continuum terminal state constraints. This form of the problem is referred to as constrained model-predictive trajectory generation. Other variations include constrained optimization model-predictive trajectory generation, which satisfies the constraints defined in Equation 2.10 while minimizing a penalty function, and unconstrained optimization model-predictive trajectory generation, which replaces the terminal state constraints with cost function minimization.

MOTION PLANNING

Like trajectory generation, mobile robot motion planning is the problem of determining acceptable actions which move a mobile robot from an initial state to a target terminal state in the presence of obstacles, constraints, and/or while optimizing utility. The scale, prevalence of local optima, and solution methods differentiate motion planning from trajectory generation problems.

Graph search in regularly discretized state spaces is an important class of mobile robot motion planning techniques. Mobile robot motion planning problems can often be approximated in low-dimensional graphs, which are efficient to

search and provide globally optimal or near-optimal solutions. In these approaches, a graph search algorithm (A*, D*, etc ...) searches a *state transition graph* that represents reachable states from the current state. Nodes in the state transition graph represent reachable positions, orientations, and configurations of the vehicle. Edges in the graph represent actions and costs of transitioning between nodes in the state transition graph. In mobile robot motion planning, edge costs are typically evaluated as distance traveled, energy consumed, or risk experienced. Following similar notation outlined in LaValle (2006), a discretized state space ($\mathbf{S}$) defines the set of reachable states in the graph. The initial ($\mathbf{s_I}$) and target terminal nodes ($\mathbf{s_G}$) are members of this discretized state space:

$$\mathbf{s_I} \in \mathbf{S} \tag{2.11}$$

$$\mathbf{s_G} \in \mathbf{S} \tag{2.12}$$

The forward and inverse *state mapping equation* ($\mathbf{f_{SME}}$) computes the mapping between continuous and discrete representations. Specifically the state mapping equation transforms continuum vehicle states ($\mathbf{x}$) into discretized nodes ($\mathbf{s}$) in the state transition graph:

$$\mathbf{s} = \mathbf{f_{SME}}(\mathbf{x}) \tag{2.13}$$

$$\mathbf{x} = \mathbf{f_{SME}}^{-1}(\mathbf{s}) \tag{2.14}$$

This mapping is important in two ways. First, it determines the discretized representations of initial states and target terminal states in the graph. Second, it provides the inverse mapping from discretized representations into continuum states to evaluate costs, terrain interaction, and vehicle configuration.

For every state in the state transition graph, an action set ($\mathbf{U}(\mathbf{x},t)$) describes edges in the state transition graph. Each edge is defined by the application particular action ($\mathbf{u}(\mathbf{x},t)$) with the *state transition equation* ($\mathbf{f_{STE}}(\mathbf{s},\mathbf{u}(\mathbf{x},t))$):

$$\acute{s} = \mathbf{f_{STE}}(\mathbf{s},\mathbf{u}(\mathbf{x},t)) \tag{2.15}$$

The state transition equation represents the motion from one representation of a state to another. In the context of mobile robots, it is mathematically equivalent to the numerical integration of the equations of motion from Equation 2.7:

$$\mathbf{f_{SME}}^{-1}(\acute{s}) = \mathbf{f_{SME}}^{-1}(s) + \int_{t_I}^{t_F} \mathbf{f_{PMM}}(\mathbf{x}(t),\mathbf{u}(\mathbf{x},t),t)\, dt \tag{2.16}$$

The dimensionality, density, and scale of the search space all determine if a motion planning problem is tractable. The computational costs associated with searching high dimensional, dense, and/or long-scale search spaces can cause platforms to exhaust memory or be too processor intensive for real-time or near real-time operation. Many contemporary implementations of regularly discretized search spaces for mobile robots motion planning search a low

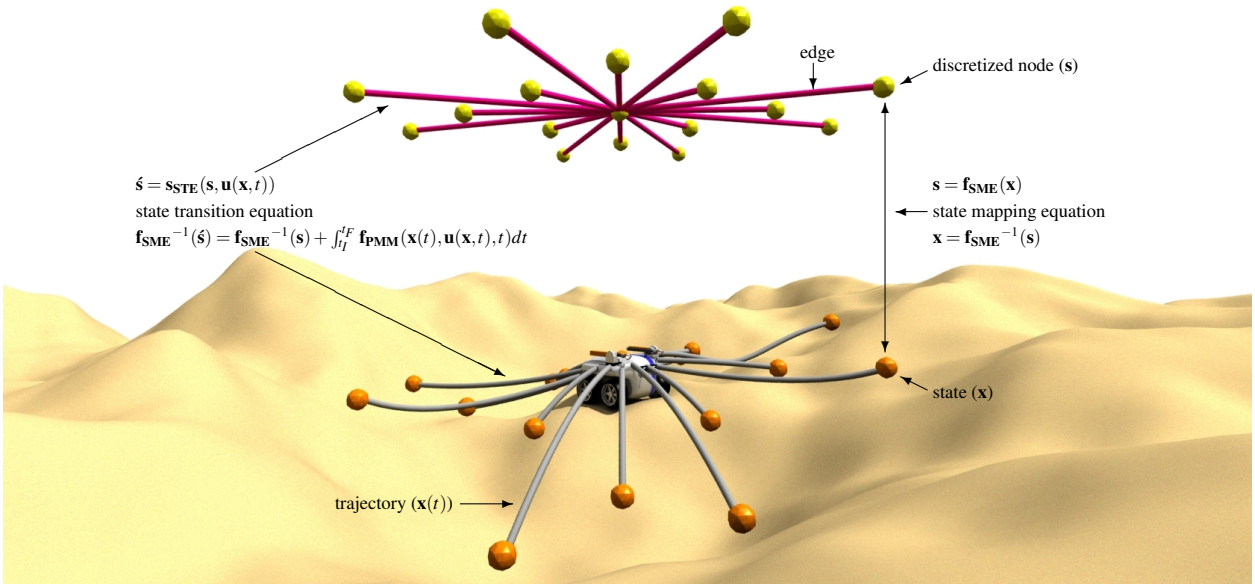


Figure 2.1: State mapping and transition equations. The state mapping equation defines the correspondence between discretized nodes and real-valued states. The state transition equation represents transitions (edges) in the discretized motion planning graph that represent vehicle motion (trajectories) in the environment.

dimensional state space representing two-dimensional positions and orientations:

$$(x, y, \psi) \in SE(2) \tag{2.17}$$

Connectivity in regularly sampled search spaces is provided by a *control set*. A control set ($\mathbf{U}(\mathbf{s_p}, \mathbf{s_c})$) determines the edges and nodes added to the search space based on the value of the current node. The number of edges in the control set is called the *outdegree*. The outdegree of a control set influences the degree of sampling, and optimality of a search space. Further details and examples of control sets appear in Section 5.1.

Motion planning search spaces evaluate the quality of edges based on a cost map. A cost map is a discretized assessment of utility or cost of vehicle states in the world. The values in a cost map are typically provided by engineered functions that map perceptual information into a single metric useful for a motion planning algorithm.

NAVIGATION

Mobile robot navigation is the challenge of selecting intelligent actions from the continuum to make progress towards achieving some goal while considering the limited perceptual information, computational resources, and planning time for the system. It is also often viewed in mobile robotics as the combined problem of obstacle avoidance and path following.

It is important for a vehicle to understand the effects of its own dynamics to maintain safety in complex environments. In the worst case, a vehicle must plan beyond the stopping distance. Navigators are typically formulated in the same manner as unconstrained optimization model-predictive trajectory generation. Initial state, predictive motion model, and planning horizon constraints must be satisfied and a cost functional, based on the combined action safety and goal progress, is minimized. Navigators are an important part of modern autonomy systems as fast techniques do not exist to generate long distance motions that consider predictive motion models of high enough quality to ensure safety.

2.2 CHALLENGES

Three important challenges in mobile robot trajectory generation, motion planning, and navigation are completeness, optimality, and feasibility. In brief, completeness is the degree to which a motion planner can produce a trajectory if one exists, optimality is the relative quality of the determined trajectory, and soundness or feasibility is a measure of constraint compliance.

COMPLETENESS

Motion planning completeness is the property of a solution algorithm of whether an action that satisfies the two-point boundary value problem can be determined if one exists.

Motion planning completeness is the property of searching the search space completely. A motion planner which can find an action (in the continuum) for any pair of boundary states if it exists is considered to be *complete*. Full completeness for nontrivial mobile robot systems (such as those with nonholonomic, kinematic, or dynamics constraints and/or operate in obstacle-filled environments) is however rarely achieved. For sampling-based motion planners, the concept of fully complete motion planners is replaced with the notion of *resolution complete* planners. A resolution complete regular discretized motion planner is a guarantee that if an action exists within the discretized representation of the continuum, that it can find the corresponding inputs which satisfy the motion planning problem constraints. Similarly, motion planners with randomized sampling may be *probabilistically complete*, which guarantees that if the search space is infinitely expanded, an action that satisfies the motion planning problem can be found. Another concept introduced in Green and Kelly (2007) is that of *relative completeness*. Relative completeness is the likelihood of one search space to produce a solution given a fixed amount of time or consumed computational resources. The fundamental design space of sampling-based motion planners is one which trades off completeness and sampling density for cost of evaluation and runtime performance.

OPTIMALITY

While related to completeness, optimality is a different criteria for a motion planner that measures the quality, as opposed to existence, of solutions. Often-used forms of optimality criterion include minimum distance, time, energy, risk, although generally any measure of interest could be used. Optimality can be strongly tied to the expressiveness

and density of actions in a search space. Local and global optimality are two different forms of this concept. Global optimality is achieved when the minimum-cost solution for all feasible actions is determined. Conversely, local optimality occurs when first derivatives of the cost function with respect to action freedoms are flat and second derivatives are positive. A particularly complex environment may have many locally optimal actions and trajectories but only one globally optimal solution. Research in search space design intends to develop techniques that best represents the globally (and locally) optimal solution(s) in an environment as efficiently as possible. Similar to relative completeness, *relative optimality* is the measure of the quality of solutions generated with a fixed amount of time or consumed computational resources.

FEASIBILITY

As mobile robots move into more difficult environments, travel at higher speeds, and perform more complex tasks, action feasibility of motion planner solutions becomes very important. The inability of a navigator or path follower to track the trajectory determined by higher level guidance vastly increases the complexity of their respective problems.

Actions which do not satisfy the state transition equation (2.15) are not feasible. Four-connected state transition graphs may violate soundness for some platforms. For differentially steered mobile robots (or others which can execute turn-in-place actions), their actions can be modified such that a motion plan from this graph is followable. Conversely, an Ackermann-steered vehicle, which cannot independently control angular velocity from linear velocity and cannot turn in place, would not be able to exactly follow the trajectory. If reference trajectory deviation greatly increases platform risk, a better strategy would have been to incorporate higher degrees of feasibility into the search space design.

Another example includes environments where terrain shape is non-negligible. Whereas some actions may satisfy the state transition equation within reasonable bounds, environmental interaction can significantly influence or even remove an edge from the graph. Unlike kinematic and dynamic constraints, feasibility for terrain shape and environmental effects cannot always be determined in the control set construction stage.

2.3 PROBLEMS ADDRESSED

This thesis addresses several important problems in mobile robot motion planning, navigation, and control in complex environments. The foremost goal of this thesis is to develop techniques to improve the optimality and relative optimally of mobile robot search spaces. To adjust the search space while maintaining topology and feasibility, techniques for real-time model-predictive trajectory generation are necessary to effectively modify edges and nodes in the motion planning graph are necessary. To navigate such paths effectively, methods to follow intricate paths in complex environments unforgiving of path following error are required.

IMPROVE SEARCH SPACE OPTIMALITY AND RELATIVE OPTIMALITY

One approach to improving search space optimality is to increase the sampling (either through resolution or outdegree) of the motion planning graph. One drawback of this approach is efficiency, as the search algorithm must search a mo-

tion planning graph that is exponentially denser to find a solution. This is especially important for non-homogeneous and cluttered environments, as search space complexity may not adapt to the required problem complexity. Increasing the sampling may actually reduce relative optimality, as search slows and computational resources required increase.

A standard approach for addressing search space optimality is to apply trajectory deformation techniques. These methods however suffer from two important limitations. The first issue is feasibility of the resulting action. Some smoothing techniques modify the states that define the trajectory and may not consider the kinematic, dynamic, and environmental interaction constraints that make the action feasible. Violating trajectory feasibility is risky because the navigation and control algorithms now must deviate from or alter the originally defined action. The second and potentially more important issue is optimality guarantees of the smoothed solution. If the smoothed trajectory is homotopically distinct from the globally optimal solution it will likely never be found using gradient-based deformation techniques.

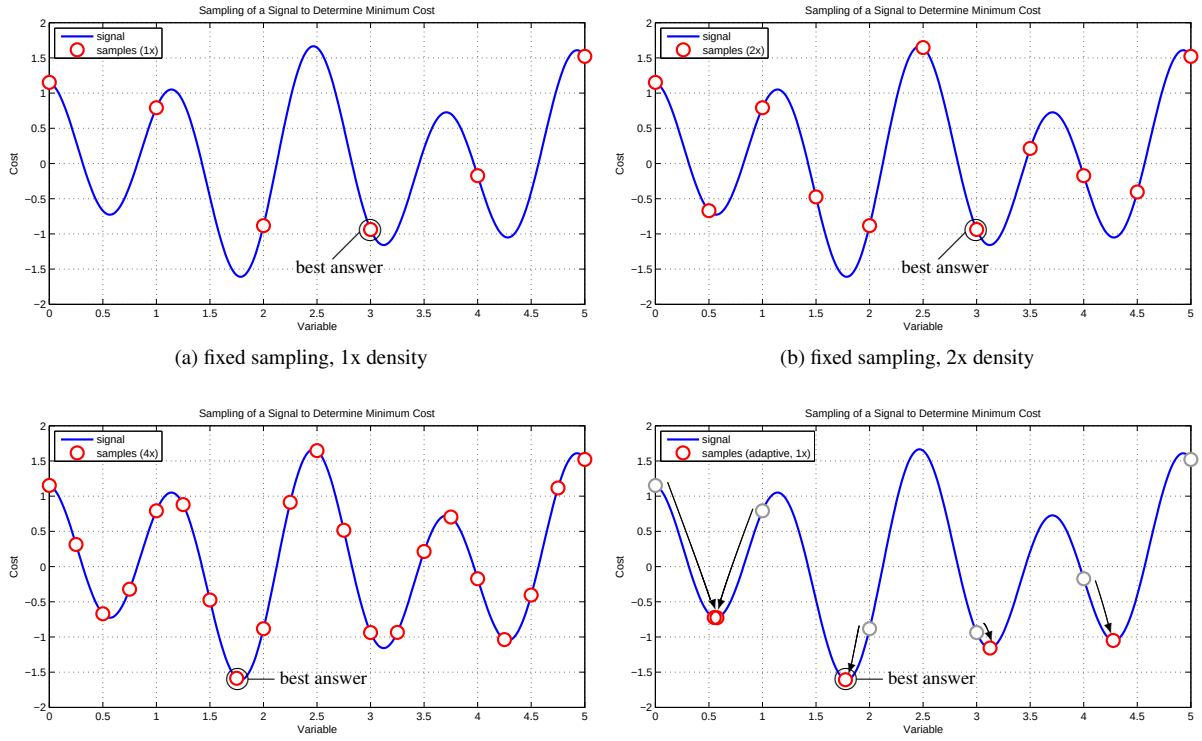
An alternative approach is to deform the search space and adapt it to the environment during construction. If the edges in the search space could be regenerated efficiently, nodes and edges could also be modified to provide locally optimal connectivity in the motion planning graph. By modifying the structure of the search space before or during the graph search, better solutions potentially closer to the global optimum can be found.

Consider the one-dimensional search problem shown in Figure 2.2. Here, increasing the density of the sampling improves the probability of finding the minimum cost solution, but guarantees about determining the globally optimal solution cannot generally be made. In Figure 2.2a, the optimal solution at this discretization is found at a value in the third cost function valley from the left. Gradient-based smoothing of the determined solution would result in simply descending into the local optima and would not reach the globally optimal solution at the bottom of the second valley from the left. Doubling the sampling of values has no effect (Figure 2.2b) as it results in the same solution as the first sampling. Only when sampling is increased by a factor of four (Figure 2.2c) results in the best solution found being located in the globally optimal solution region. Two problems with constantly increasing the search space sampling are the increased computational resources (runtime and memory) consumed and determining when to stop increasing the sampling.

Local relaxation of samples to their minimum-cost configuration greatly increases the likelihood of finding the globally optimal solution. Even using the coarsest search space (Figure 2.2d), the globally optimal solution is determined without post-processing the solution or increasing sampling. This thesis extends this concept to motion planning search space generation, where samples are actually nodes in the motion planning graph and the gradients used to adjust sample locations are determined by perturbations of state mapping equations based on the aggregate cost of edges from that node. Search space adaptation is about smart sampling in the space of states and actions of motion planning graphs.

Enable Efficient, High-Fidelity Model-Based Mobile Robot Trajectory Generation, Motion Planning, and Navigation

Mobile robots must reason about their interaction with the environment in trajectory generation, motion planning, and navigation to operate safely in unrehearsed terrain. Significant variations in the fidelity of actions determined at



(a) fixed sampling, 1x density

(b) fixed sampling, 2x density

(c) fixed sampling, 4x density

(d) adaptive sampling, 1x density

Figure 2.2: Search for a minimum-cost value using fixed and adaptive sampling techniques is a simplified version of the adaptive model-predictive search space concept. Here, a minimum-cost solution is determined by increasing the density of fixed sampling or by exploiting local gradient information to more effective sample the space of solutions.

each level can be hazardous and impede mission progress. Two prominent disturbances include environmental and internal changes in mobility. Environmental disturbances are those that dampen or modify the forces transmitted on the environment by the vehicle. Internal mobility changes are those that effect the forces imposed on the environment by the vehicle, such as motor degradation and failure, blown tires, and significant changes in mass. Modeling these effects is important to accurately estimate connectivity in motion planning graphs, predict safety in navigators, and to determine continuum actions for trajectory generation problems.

PROVIDE CAPABILITY TO NAVIGATE INTRICATE PATHS IN COMPLEX ENVIRONMENTS

The ability for a mobile robot to generate an intricate path in a complex environment is marginal if the mobile robot cannot effectively navigate or follow the trajectory. Intricate paths composed of cusps, turn-in-place, and multi-turn maneuvers pose challenges for traditional path follow techniques as some path points becomes poorly defined at

geometric singularities and discontinuities. Even well-applied action-space sampling techniques fail to accurately predict vehicle motion unless they can also sample in the space of intricate paths efficiently. A potentially better approach explored in this thesis is to exploit the structure of the reference trajectory to parameterize and initialize the search space for relaxation of a corrective action. Such as technique could explore actions that include the intricate maneuvers of the reference trajectory but deform the action shapes to compensate for path deviations or unmodeled disturbances.

CHAPTER 3

RELATED WORK

The traditional cycle for mobile robot autonomy systems is *sense*, *plan*, and *act*. This thesis is generally concerned with developing improved techniques for the latter two in difficult, challenging environments. Substantial prior research has advanced mobile robot trajectory generation, motion planning, navigation, and control to the point where mobile robots can successfully traverse open deserts, navigate urban environments, and explore the oceans and other bodies in our solar system. This section reviews prior work in mobile robot trajectory generation, motion planning, and navigation related to the topics of this thesis.

3.1 TRAJECTORY GENERATION

In the context of mobile robot autonomy systems, trajectory generation is the problem of determining obstacle-free or minimum-cost paths subject to vehicle mobility constraints. Whereas motion planning is typically formulated as graph construction and search, trajectory generation is most often approached using analytical geometric techniques or continuum optimization processes. Geometric techniques based on interpolating function and boundary state constraint satisfaction rely on using a simplified model of actions and vehicle mobility. Continuum optimization techniques are more general as they are steered by gradient information, although such methods are typically guaranteed to determine only locally optimal solutions.

GEOMETRIC METHODS

Some of the earliest work in trajectory generation are geometric techniques involving the composition of optimal paths from a sequence of line segments, arcs (Dubins, 1957), clothoids (Kanayama and Miyake, 1985; Shin and Singh, 1990), and cubic spirals (Kanayama and Hartman, 1989). The desire for high-order geometric primitives was intended to enable higher levels of continuity between boundaries of trajectory segments. Bézier splines have also been used to satisfy arbitrary position and heading boundary state constraints by defining a sequence of knot points along the path (Komoriya and Tanie, 1989). Other techniques based on sinusoidal and Fourier series inputs also appear throughout the literature (Brockett, 1981; Tilbury et al., 1992; Murray and Sastry, 1993). These method exploit the geometry of the problem to solve for the unknown path parameters directly, but cannot generally solve for collision-free paths in obstacle fields.

OPTIMIZATION TECHNIQUES

Variational (optimization) techniques for trajectory generation, which search the continuum for a locally optimal solution, are as old as optimal control theory and have been used in most fields that employ automatic control (Betts, 1998). These approaches generally apply numerical methods to satisfy some set of boundary conditions and/or minimize some cost function by searching in the space of actions or states. In manipulation, joint trajectories were automatically generated using optimal control and cubic polynomials in (Lin et al., 1983). Minimum-time paths between boundary states are treated as a control problem in (Baker, 1999). (Jackson and Crouch, 1991) applied the shooting method to solve for trajectories using cubic spline primitives. Energy minimization was used in (Delingette et al., 1991) to successively deform a curve until it met the boundary constraints. A near real-time optimal control trajectory generator is presented in (Reuter, 1998) that solves eleven first-order differential equation subject to the state constraints. A real-time trajectory generation algorithm for differentially flat systems is presented in (Faiz et al., 2001), where an approximation of nonlinear constraints are replaced by linear inequality constraints. An algorithm that modifies control inputs to maximize distance to obstacles for multi-body wheeled mobile robots while satisfying kinematic constraints is presented in (Lamiraux and Laumond, 2004). (Kim and Tilbury, 2001) used methods based on solving an approximate linearized problem (when systems are input-output linearizable) for UAV trajectory planning. Some of the most recent work in optimal control trajectory generation include (Kalmár-Nagy et al., 2004), where near-optimal paths are constructed for omni-directional vehicles using bang-bang optimal control methods. Their methods generate minimum-time omni-directional trajectories subject to the sophisticated dynamics and actuator models. In Nagy and Kelly (2001); Kelly and Nagy (2003), a real-time algorithm to solve planar trajectory generation problem between arbitrary boundary states by linearizing and inverting the equations of motion is presented.

Most of the mobile robot trajectory generation methods discussed so far have assumed a flat world or simple model dynamics. By contrast, at the level of global or regional motion planning where primitive trajectories are sequenced together, the uneven terrain shape is often known. It is normally considered only in terms of its effect on the overall objective function being optimized rather than in terms of its lower level effect on the motion itself. Optimal Bézier spline paths are generated using a Bézier patch representation of the terrain shape in Shiller and Chen (1990); Shiller and Gwo (1991). Amar et al. (1993) adapts a sub-optimal sequence of cubic spline paths by optimizing the heading at several sub-goal positions along the path. A three-dimensional kinematic model of the vehicle is used to determine the attitude, elevation, and configuration of the body to determine stability along the path.

3.2 MOTION PLANNING

Motion planning is one of the most actively researched problems in mobile robotics. The challenge is the search for a globally optimal action subject to obstacle fields, vehicle mobility constraints, and non-trivial environmental interaction. Motion planning techniques are typically formulated as sequential search operating on a graph representing the path continuum. Such graphs are typically constructed either by deterministic or probabilistic techniques. Deterministic methods are those with specific rules and structure to the search space that provide resolution optimality guarantees. Probabilistic techniques instead search a much larger portion of the state space through randomized

branching methods.

DETERMINISTIC MOTION PLANNING SEARCH SPACES

The most widely applied variety of motion planning algorithms are generally classified as deterministic graph search. These methods also apply to the problem of solving for obstacle-free and minimum-length paths that satisfy nonholonomic and boundary constraints (Canny et al., 1988; Jacobs and Canny, 1989; Barraquand and Latobe, 1989; Reeds and Schepp, 1990; Laumond et al., 1990). The drawback of such techniques for motion planning compared to continuum techniques used in trajectory generation is the resolution lost due to discretization of state space and/or control space. Environmental representation in regularly discretized state spaces has progressed from two-dimensional grids and three-dimensional pose graphs to arbitrary defined state lattices (Pivtoraiko and Kelly, 2005). Designing efficient control sets for state lattices representations is described in (Pivtoraiko et al., 2007).

Many techniques exist for searching and replanning in these motion planning graphs. A commonly applied method to search mobile robot motion planning graphs is A* (Hart et al., 1968), which searches the state space guided by a heuristic estimate of the minimum-cost path to the goal. Efficient replanning is presented in Stentz (1994); Koenig and Likhachev (2002) that plans backwards from the goal and modifies only the motion plan segments affected by changing or observed state costs. Anytime variations of A* (Zhou and Hansen, 2002) and D* (Likhachev et al., 2005) show that a suboptimal solution can be generated quickly and iteratively refined by modifying the heuristic inflation factor. Field D* (Ferguson and Stentz, 2006) exploits linear interpolation to more accurately estimate edge costs in a grid to generate paths that better reflect the continuum optimum solution.

Techniques for rough terrain motion planning have also been widely researched. Siméon and Dacre-Wright (1993) compose and search a discretized state space by sampling in the space of forward and reverse actions. Each edge is simulated with the assumption that it is executed on a flat plane in the local coordinated frame of the chassis. One technique that does account for the influence of terrain on motion on a local and regional scale is the two-level planner presented in (Cherif et al., 1994; Cherif, 1999). This work determines a global path in a two-dimensional configuration space of the robot using A* using a local motion planner to generate trajectories between nodes that consider terrain shape and kinematic and dynamic constraints. The local path planner generates a path composed of straight line segments and arcs that is tracked by a dynamic model of the vehicle. The disturbances from terrain shape and vehicle dynamics are compensated by iteratively deforming the path until the simulated vehicle satisfies the boundary state constraint. Iagnemma et al. (1999) propose a two-step motion planning technique that first applies A* search on a graph that penalizes terrain unevenness, distance, and turning. The resulting path is subsequently evaluated using a high-fidelity physical simulation to determine if the path is safe and feasible. High-risk segments of the path are handled by increasing the cost with the unsafe associated states and re-planning the motion. A similar approach that considers wheel slip dynamics is presented in Ishigami et al. (2007), where a motion plan is generated based on a three-step procedure of graph search, dynamic simulation, and path evaluation.

Path smoothing in the continuum is an often-applied process to compensate for the mere resolution optimality guarantees of deterministic motion planning. In (Laumond, 1995), a holonomic geometric path is found using techniques developed in Avnaim et al. (1988) in an obstacle field. A feasible path is constructed by techniques including

Reeds and Schepp (1990) and Fernandes et al. (1993) and subsequently optimized. Shiller (1999) presents a method that locally relaxes several global motion planning solutions and returns the minimum-cost solution. Several solutions are relaxed because there is no guarantee that the resolution optimal solution returned by the global motion planner is actually the continuum optimum solution. More recently, a technique for trajectory optimization that exploits covariant gradient techniques is developed to refine paths initialized by straight lines through configuration space and bi-directional Rapidly-Exploring Random Trees (Ratliff et al., 2009).

Heuristic graph search in continuous space for mobile robot motion planning (Hybrid A*) is presented in Dolgov et al. (2008) . Nodes in the motion planning graph associate a cost for a continuous state in each cell. This enabled motion planning graphs to be feasible, but it does not present optimality guarantees from the discretization of the action space and the pruning of actions so that only a single continuous state can exists in a cell. Path smoothing is also applied based on conjugate gradient methods that minimize an aggregate cost based on distance to the nearest obstacle and vehicle heading change. Interesting related work in developing motion planning search spaces that adapt to environmental constraints appears in Chestnutt (2007). In this work, a portion of the state space (footstep locations) for a humanoid robot is optimized during the search if a footstep location is determined to be invalid.

PROBABILISTIC MOTION PLANNING SEARCH SPACES

Probabilistic techniques have been become a popular technique for generating motion plans in complex environments or high-dimensional systems where finding a solution may be more important than finding the optimal solution. Specifically the method of Rapidly-Exploring Random Trees (RRT) has been widely applied to motion planning problems spanning mobile manipulation and biped locomotion (LaValle and Kuffner, 2001). Rapidly-Exploring Random Trees have been extended to consider terrain shape, physical models, and sensor uncertainty using Particle-based Rapidly Exploring Random Trees in Melchior et al. (2007).

Another related class of motion planning techniques are probabilistic roadmaps (PRM). These techniques sample in the configuration space of the vehicle to determine collision-free states (Kavraki et al., 2006). The remaining collision-free states are connected using a local motion planner to compose a motion planning graph, which can be subsequently searched by the previously described techniques. One significant advantage of such techniques in a high-dimensional configuration spaces is that by removing states that collide with obstacles from the randomized sampling, fewer edges need to be generated and evaluated in the search space.

3.3 NAVIGATION AND CONTROL

Navigation and control is the problem of intelligently selecting an action based on limited perceptual information, computational resources, and planning time. Mobile robot navigators attempt to determine actions that ensure safety and guide the platforms towards the goal. Typically this is achieved using one of two techniques. Path following control techniques attempt to follow the global guidance information. Sampling-based navigators search the local action or state space to determine if a motion that deviates from the reference trajectory is the correct action. The process itself typically takes the form of constrained or unconstrained optimization, where an optimal strategy must be

determined from a set of candidate actions. While first approaches were based on sampling in the space of geometric actions, variations of model-predictive control techniques are far more common in contemporary applications as they consider the constraints of vehicle motion while optimizing some utility in the process.

PATH FOLLOWING CONTROL AND TRAJECTORY PLANNING

Early path following controllers for mobile robots operate by tracking a single lookahead point (Amidi, 1990). These techniques have been greatly extended in the literature (Coulter, 1992) and have been applied on numerous mobile robot systems. A good survey and independent evaluation of geometric and feedback-based path following techniques for terrestrial mobile robot applications appears in Snider (2009).

Model-predictive control (Mayne et al., 2000) and trajectory planning techniques have recently gained interest for mobile robot control problems where autonomy systems require feasible actions. A utility function minimization technique for terrestrial mobile robots applications is described in (Cremean et al., 2006). Horizon-limited trajectories are planned by minimizing the steering and acceleration control effort subject to a vehicle dynamics model that considers limited steering rates and rollover constraints. Obstacles are avoided by representing them as very low-speed regions and are naturally avoided in this framework. The local minima problem is handled by seeding the optimization function with a coarse spatial path. The *dynamic window* approach (Fox et al., 1997) selects translational and rotational velocities by maximizing an objective function based on the heading to the target position, distance to the closest obstacle, and the velocity of the robot. The search space is generated using arcs and is restricted to reachable and safe velocities at its current state. Kuwata et al. (2005) proposes methods for unmanned aerial vehicles that optimize a sequence of inputs but relaxes terminal state constraints in order to achieve computational efficiency. A feed-forward model-predictive control for sandy slope traversal based on a thrust-cornering characteristic diagram (an experimental model developed with particular soil parameters and slopes) is presented in Ishigami et al. (2009). The same article also describes the implementation of a feedback-based path follower, which showed that the feedback system can compensate for wheel slip if the state can be observed reliably.

There has been recent work in developing path following controllers for actively articulated mobile robots to maximize stability. A behavior-based controller that reconfigures the robot chassis arm and shoulder freedoms based on estimated stability is presented in Schenker et al. (2000). Iagnemma et al. (2003) proposes a technique for maximizing the stability angles of a planetary rover suspension by actively articulating chassis freedoms. A different method based on repositioning the center of mass to achieve maximal stability and reduce energy consumption (via motor torque) is presented in Nakamura et al. (2007).

Potential fields (Koren and Borenstein, 1991; Haddad et al., 1998) have regularly been applied to obstacle avoidance and navigation, where attractive forces represent goals and repulsive forces represent obstacles. In (Shimoda et al., 2005), potential fields generate actions which considered dynamic hazards such as rollover and terrain shape in their evaluation. (Lacroix et al., 2002) applies potential fields for navigation in simple terrain. For more difficult environments, they applied arcs and arc-trees that consider terrain shape in their forward simulation to generate the navigation search space.

SAMPLING-BASED NAVIGATION

As path following control techniques are blended with obstacle avoidance, the problem is best described as mobile robot navigation. There has been substantial research in control set generation for mobile robot navigation search spaces. Some early work in this appears in Kelly and Stentz (1998), where the local motion planning search space is generated by sampling in the control space of curvature. Each control is passed through a vehicle dynamics model to estimate the response of the vehicle to the control. The shape of the response is highly dependent on the vehicle model and the initial vehicle state (curvatures and velocities). Similar approaches have been adapted in a variety of other unmanned ground vehicles (Wettergreen et al., 2005; Kelly et al., 2006). Egographs (Lacze et al., 1998) represent another method for generating local motion planning search spaces. This approach generates, off-line, a well-separated dynamically feasible search space for a limited set of initial states. Online adaptation to changes in terrain or vehicle models are not considered. Arcs and point turns comprised the control primitive sets that were used for autonomous hazard avoidance for Spirit and Opportunity during the Mars Exploration Rovers (MER) mission (Besiadecki et al., 2007). This approach was an extension of Morphin, an arc-planner variant where terrain shape was considered in the trajectory selection process (Simmons et al., 1995; Singh et al., 2000). Another closely related algorithm is the one presented in (Bonnafous et al., 2001), where the an arc-based search space is evaluated based on considering risk and interest. Most local motion planners exploit some form of global guidance provided by a regional or global motion planner (Stentz and Herbert, 1995). In Kelly et al. (2006), the Field D* algorithm (Ferguson and Stentz, 2005) is applied to efficiently regenerate path plans over long distances to provide global guidance for model-predictive path sets sampled in input space of the vehicle.

Probabilistic techniques for sampling the reachable state space have recently been applied to the problem of generating dynamically feasible actions through complex environments. This method is well suited to the problem of navigating complex environments because of its ability to search high dimensional input spaces and it can consider vehicle dynamics and terrain shape in its solution (Berg et al., 2006; Melchior et al., 2007). Frazzoli et al. (2001) uses RRT's to navigate among static and dynamic obstacles, where tree expansions are done in state space rather than input space. A controller is applied to determine the feasible action, if one exists, that connects the current state to the new branch state. Chen and Fraichard (2007) applied partial motion planning, an iterative technique based on Rapidly-Exploring Random Trees (RRT) that considers vehicle model constraints such as acceleration, steering velocity, and steering angle bounds and the real-time operation constraint of the system for navigation in urban environments.

The satisfaction of environmental constraints in local motion planning search spaces is related to lane following, a problem that has generated a great deal of interest with the recent DARPA Grand Challenge competitions. In Miller et al. (2006), Bézier splines are constructed with terminal points defined by a lateral offset and the road shape. Dynamically infeasible trajectories are culled through a post-processing technique and an optimal navigation strategy is determined through minimizing a weighted combination of integrated path cost and steering effort. A related approach involves Bézier splines whose control points are located and adjusted by the location of obstacles (Trepagnier et al., 2006; Berglund et al., 2003). In Urmson et al. (2006), a geometric planner that conforms to environmental constraints (a trail or road network) is applied to the problem of navigating in desert environments. The problem of dynamic feasibility is handled by a path tracker and a speed planner ensures safety by slowing the vehicle during

aggressive maneuvers. Another similar system is described in Leedy et al. (2006), where the deliberative planner generates a path by searching a cost map using A* and following the trajectory using pure pursuit (Coulter, 1992) then checking it for safety. This method trades off inherent dynamic feasibility for global guidance. A reactive approach is also presented in Leedy et al. (2006) that uses a fuzzy logic controller to follow roads and avoid local obstacles. Nudges and swerves (Thrun et al., 2006), representing smooth and aggressive lateral offsets around a base trajectory, have proven effective for generating expressive local motion planning search spaces for obstacle avoidance and lane following. Search in trajectory space (κ-v) bounded by hazards and dynamic effects including steering limits, wheel-terrain interaction, rollover, and sideslip constraints is presented in Spenko et al. (2006) and demonstrated to be an effective approach at speeds up to 9 meters per second.

More recently, the DARPA Urban Challenge introduced an arguably more difficult problem: navigating in constrained, urban environments. This challenge included many difficult problems for motion planning and control, including navigating among dynamic obstacles at speeds up to 30 miles per hour, performing intricate maneuvers including k-turns and parking, and lane changes. The aforementioned model-predictive trajectory generator was used to generate the local motion planning search space by sampling in the state space of terminal vehicle states along the lane or target path (Ferguson et al., 2008; Urmson et al., 2008).

3.4 DISCRIMINATORS

This thesis develops motion planning, navigation, and control algorithms to generate better (lower cost) motion plans through complex environments. This research is distinct from previous work in several ways. First, it presents a highly general model-predictive trajectory generation technique based on formulating optimal control as a parameter optimization problem. The technique linearizes and inverts a highly general model of vehicle mobility to achieve generality and has been applied on a number of diverse mobile robot platforms. Second, it utilizes the model-predictive trajectory generation techniques to construct informed and adaptive motion planning graphs that compensate for disturbances in vehicle mobility and locally optimizes the node state mapping equations. This leads to regional motion planning search spaces that more efficiently and reliably navigate through very complex environments where terrain shape is important, obstacles are dense, and vehicle mobility constraints cannot be neglected. Lastly, this thesis presents a technique for path following navigation and control that exploits the actions that compose the reference trajectory produced by the regional motion planner to guide parameter search for an action that compensates for path following disturbances and/or avoids obstacles.

CHAPTER 4

MODEL PREDICTIVE TRAJECTORY GENERATION

Trajectory generation is the problem of determining actions that result in obstacle-free or minimum-cost paths subject to a set of state constraints and/or that optimize a utility function. Model-predictive trajectory generation is an extension of this problem that adds predictive motion model constraints that map actions onto state response. Predictive mobility model compensation in trajectory generation enables autonomy systems to determine feasible trajectories that are more reliable to execute.

In the context of mobile robots, model-predictive trajectory generation is often the problem of determining steering and velocity inputs that maneuver a vehicle through an environment while satisfying the predictive motion model. This is further complicated in complex environments where the effects of terrain shape and terramechanical properties on the predictive motion model are non-negligible. This chapter describes three techniques developed for efficient mobile robot model-predictive trajectory generation: constrained model-predictive trajectory generation, unconstrained optimization trajectory generation, and constrained optimization trajectory generation. The efficiency of the presented techniques comes from transforming the general optimal control problem to continuum search in a parameterized subspace of all feasible actions. Generality is achieved by numerical estimation of the partial derivatives required by the continuum techniques. Simulation and field results are presented for several mobile robot platforms including predictive compensation of terrain shape and impaired mobility, motion and articulation for mobile manipulators, chassis optimization for planetary rovers, and local motion planning action and search space generation.

4.1 CONSTRAINED MODEL-PREDICTIVE TRAJECTORY GENERATION

Constrained model-predictive trajectory generation is a common and important problem in mobile robot motion planning, navigation, and control. It is the challenge of determining actions that satisfy a predictive motion model and a set of state constraints. In motion planning, it is used to determine control sets used to construct connected state transition graphs. In navigation, it is used to construct local motion planning search spaces that satisfy initial state constraints and exploit global guidance information and environmental constraints. In control, constrained model-predictive trajectory generation is applied to the problem of determining (locally) optimal actions that reacquire reference trajectories.

Figure 4.1 shows an example of a common application of constrained trajectory generation: instrument placement. Here, a mobile robot requires a set of actions that move the vehicle from an initial state ($\mathbf{x_I}$) to a target terminal state ($\mathbf{x_T}$) while satisfying the mobility constraints of the vehicle and the environment:

The problem becomes continuum search for actions that satisfy the following two-point boundary value constraints

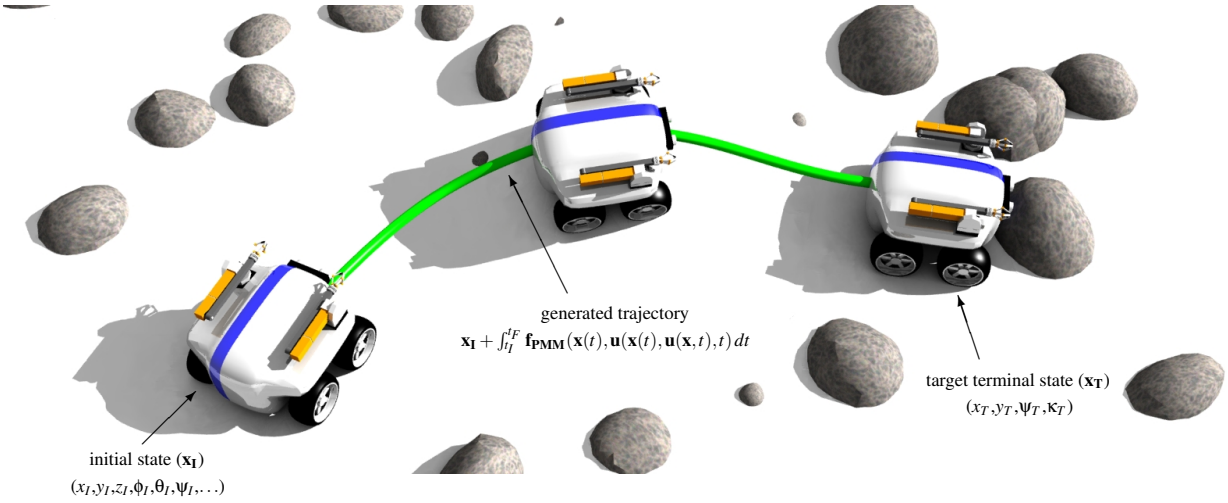


Figure 4.1: The constrained model-predictive trajectory generation problem is defined as the search for actions that moves a vehicle from an initial state ($\mathbf{x_I}$) to a target terminal state ($\mathbf{x_T}$) subject to a predictive motion model ($\mathbf{f_{PMM}}(\mathbf{x}(t),\mathbf{u}(\mathbf{x},t),t)$). Here, a vehicle is required to move from a full initial state to a target position (x_T, y_T), heading (ψ_T), and curvature (κ_T).

($\mathbf{x_C}$):

$$\mathbf{x_C} = \begin{bmatrix} \mathbf{x_I} \\ \mathbf{x_T} \end{bmatrix} = \begin{bmatrix} \mathbf{x}(t_I) \\ \mathbf{x}(t_I) + \int_{t_I}^{t_F} \mathbf{f_{PMM}}(\mathbf{x}(t),\mathbf{u}(\mathbf{x},t),t)\,dt \end{bmatrix} \tag{4.1}$$

The initial state constraints are often satisfied implicitly as vehicles can not instantaneously change their position, orientation, and configuration. The target terminal state constraints requires the integral term to account for the difference between the initial and target terminal states. The integral term represents an estimate of vehicle motion as a time integral of the predictive motion model.

Generally, a solution for actions can be determined by inverting the following relationship:

$$\mathbf{x_T} - \mathbf{x}(t_I) = \int_{t_I}^{t_F} \mathbf{f_{PMM}}(\mathbf{x}(t),\mathbf{u}(\mathbf{x},t),t)\,dt \tag{4.2}$$

Inverting Equation 4.2 is the challenge of constrained model-predictive trajectory generation. One issue is that many mobility systems have predictive motion models that cannot be integrated analytically for nontrivial actions. Since the actions cannot generally be isolated from the predictive motion model integration, numerical methods must be used to determine the system inputs. In addition, the space of actions is often undefined, and continuum search of all feasible actions is generally impractical.

Two strategies are employed to overcome these challenges. The solution space is reduced by parameterizing the

actions:

$$\mathbf{u}(\mathbf{x},t) \longrightarrow \mathbf{u}(\mathbf{p},\mathbf{x},t) \tag{4.3}$$

Controlling the action shapes through a set of parameter values ($\mathbf{p}$) transforms the original problem of determining a general set of actions into finding the set of parameters that satisfy the relationship. Provided an initial guess of parameter values ($\mathbf{p_I}$), a solution to this problem can be found by determining the parameter correction ($\acute{\mathbf{p}}$). The problem transforms now into one of determining the proper correction factor that satisfies the relationship in Equation 4.5:

$$\mathbf{p} = \mathbf{p_I} + \acute{\mathbf{p}} \tag{4.4}$$

$$\mathbf{x_T} - \mathbf{x}(t_I) = \int_{t_I}^{t_F} \mathbf{f_{PMM}}(\mathbf{x}(t), \mathbf{u}(\mathbf{p_I} + \acute{\mathbf{p}}, \mathbf{x}, t), t)\, dt \tag{4.5}$$

The constrained model-predictive trajectory generation approach presented here is formed as a root finding problem. A constraint error ($\Delta\mathbf{x_T}$) representing the difference between the target and realized terminal states is defined as a function of the action parameter vector:

$$\Delta\mathbf{x_T}(\mathbf{p}) = \mathbf{x_T} - \left[\mathbf{x}(t_I) + \int_{t_I}^{t_F} \mathbf{f_{PMM}}(\mathbf{x}(t), \mathbf{u}(\mathbf{p},\mathbf{x},t), t)\, dt\right] \tag{4.6}$$

An action that satisfies the model-predictive trajectory generation problem is determined by reducing the constraint error to zero. In practice, any root finding technique can be applied to determine the parameter values that satisfy the relationship in Equation 4.5. Newton's root finding technique is an efficient method for solving systems of equations. The correction factor is proportional to both the constraint Jacobian and the constraint error:

$$\mathbf{p}_{i+1} = \mathbf{p}_i - \left[\frac{\delta(\Delta\mathbf{x_T}(\mathbf{p}_i))}{\delta\mathbf{p}}\right]^{-1} \Delta\mathbf{x_T}(\mathbf{p}_i) \tag{4.7}$$

The constraint Jacobian is an $n \times m$ matrix, where n and m are the number of rows in the constraint and parameter vector respectively. Just as closed-form time integrals of predictive mobility models cannot be determined for non-trivial mobile robot systems, their partial derivatives with respect to the constraint errors cannot be determined analytically. For example, if the constrained model-predictive trajectory generation problem posed in Figure 4.1 used a five parameter action ($\mathbf{p} = [\, p_0\, p_1\, p_2\, p_3\, p_4\,]^T$) to satisfy the four terminal state constraints ($\mathbf{x_T} = [\, x_T\, y_T\, \psi_T\, \kappa_T\,]^T$), the constraint Jacobian for this particular problem is a 4×5 matrix:

$$\frac{\delta(\Delta\mathbf{x_T}(\mathbf{p}))}{\delta\mathbf{p}} = \begin{bmatrix} \frac{\delta(\Delta x_T(\mathbf{p}))}{\delta\mathbf{p}} \\ \frac{\delta(\Delta y_T(\mathbf{p}))}{\delta\mathbf{p}} \\ \frac{\delta(\Delta\psi_T(\mathbf{p}))}{\delta\mathbf{p}} \\ \frac{\delta(\Delta\kappa_T(\mathbf{p}))}{\delta\mathbf{p}} \end{bmatrix} = \begin{bmatrix} \frac{\delta(\Delta x_T(\mathbf{p}))}{\delta\mathbf{p_0}} & \frac{\delta(\Delta x_T(\mathbf{p}))}{\delta\mathbf{p_1}} & \frac{\delta(\Delta x_T(\mathbf{p}))}{\delta\mathbf{p_2}} & \frac{\delta(\Delta x_T(\mathbf{p}))}{\delta\mathbf{p_3}} & \frac{\delta(\Delta x_T(\mathbf{p}))}{\delta\mathbf{p_4}} \\ \frac{\delta(\Delta y_T(\mathbf{p}))}{\delta\mathbf{p_0}} & \frac{\delta(\Delta y_T(\mathbf{p}))}{\delta\mathbf{p_1}} & \frac{\delta(\Delta y_T(\mathbf{p}))}{\delta\mathbf{p_2}} & \frac{\delta(\Delta y_T(\mathbf{p}))}{\delta\mathbf{p_3}} & \frac{\delta(\Delta y_T(\mathbf{p}))}{\delta\mathbf{p_4}} \\ \frac{\delta(\Delta\psi_T(\mathbf{p}))}{\delta\mathbf{p_0}} & \frac{\delta(\Delta\psi_T(\mathbf{p}))}{\delta\mathbf{p_1}} & \frac{\delta(\Delta\psi_T(\mathbf{p}))}{\delta\mathbf{p_2}} & \frac{\delta(\Delta\psi_T(\mathbf{p}))}{\delta\mathbf{p_3}} & \frac{\delta(\Delta\psi_T(\mathbf{p}))}{\delta\mathbf{p_4}} \\ \frac{\delta(\Delta\kappa_T(\mathbf{p}))}{\delta\mathbf{p_0}} & \frac{\delta(\Delta\kappa_T(\mathbf{p}))}{\delta\mathbf{p_1}} & \frac{\delta(\Delta\kappa_T(\mathbf{p}))}{\delta\mathbf{p_2}} & \frac{\delta(\Delta\kappa_T(\mathbf{p}))}{\delta\mathbf{p_3}} & \frac{\delta(\Delta\kappa_T(\mathbf{p}))}{\delta\mathbf{p_4}} \end{bmatrix} \tag{4.8}$$

In general the constraint Jacobian cannot be inverted for under-determined systems of equations such as the example described in Equation 4.8. Instead, a pseudo- or Moore-Penrose matrix inverse can be applied to realize a least-squares approximation of the inverse:

$$\frac{\delta(\Delta\mathbf{x_T}(\mathbf{p}))}{\delta\mathbf{p}} = \left[\frac{\delta(\Delta\mathbf{x_T}(\mathbf{p}))}{\delta\mathbf{p}}^T \frac{\delta(\Delta\mathbf{x_T}(\mathbf{p}))}{\delta\mathbf{p}}\right]^{-1} \frac{\delta(\Delta\mathbf{x_T}(\mathbf{p}))}{\delta\mathbf{p}}^T \tag{4.9}$$

Since analytical partial derivatives cannot generally be found for state constraint error with respect to action parameter freedoms, numerical techniques are substituted to estimate their values. The constraint error partial derivatives can be effectively determined by applying forward (Equation 4.11) or central (Equation 4.12) differences of the predictive motion model, where h represents a small parameter perturbation. Selection of the proper parameter perturbation value is a function of the predictive motion model fidelity and the desired partial derivative accuracy. Each constraint error partial derivative estimate is a $n \times 1$ vector that fills the m columns of the full constraint Jacobian:

$$\mathbf{h_j} = \begin{bmatrix} 0 & 0 & h & \dots & 0 \end{bmatrix}^T \tag{4.10}$$

$$\frac{\delta(\Delta\mathbf{x_T}(\mathbf{p}))}{\delta\mathbf{p_j}} \simeq \frac{\Delta\mathbf{x_T}(\mathbf{p_j}+\mathbf{h_j},\mathbf{p}) - \Delta\mathbf{x_T}(\mathbf{p})}{h} \tag{4.11}$$

$$\frac{\delta(\Delta\mathbf{x_T}(\mathbf{p}))}{\delta(\mathbf{p_j})} \simeq \frac{\Delta\mathbf{x_T}(\mathbf{p_j}+\mathbf{h_j},\mathbf{p}) - \Delta\mathbf{x_T}(\mathbf{p_j}-\mathbf{h_j},\mathbf{p})}{2h} \tag{4.12}$$

The generality of the presented model-predictive trajectory generation comes from the feature that the correction factor is determined entirely numerically based on differences of predicted vehicle motion. This enables the constrained model-predictive trajectory generator to determine effective parameter corrections when the environment or mobility is particularly complex, such as when the terrain shape and/or terramechanical properties are non-negligible. Figure 4.2 exhibits the general structure for the constrained model-predictive trajectory generation algorithm using Newton's root finding technique and a forward difference partial derivative estimate.

The algorithm in 4.2 is generally divided between two parts: the root finding optimization and the constraint Jacobian estimation. In CONSTRAINEDMPTG, the algorithm takes in the initial state, the target state constraints, the predictive motion model, and the action parameterization and returns valid action parameters that satisfy the state constraints. An initial guess of the parameters is first determined by an initialization function and subsequently corrected by Newton's root finding method. The partial derivatives of the constraint error with respect to the action parameters are evaluated in the ESTIMATECONSTRAINTJACOBIAN function. Since entire columns of the constraint Jacobian can be determined through a single pair of predictive motion model integrations, only m cycles are necessary to estimate the entire $n \times m$ matrix. An efficient implementation of ESTIMATECONSTRAINTJACOBIAN would store the original constraint error ($\mathbf{x_T}$) so that only $m+1$ predictive motion model integrations are necessary to estimate the constraint Jacobian.

The choice and implementation of action parameterization, predictive motion models, and initialization functions

CONSTRAINEDMPTG($\mathbf{x_I}$,$\mathbf{x_T}$,$\mathbf{u}(\mathbf{p},\mathbf{x},t)$,$\mathbf{f_{PMM}}(\mathbf{x}(t),\mathbf{u}(\mathbf{p},\mathbf{x},t),t)$)

INPUT: $\mathbf{x_I}$ (initial state), $\mathbf{x_T}$ (target terminal state), $\mathbf{u}(\mathbf{p},\mathbf{x},t)$ (parameterized action), $\mathbf{f_{PMM}}(\mathbf{x}(t),\mathbf{u}(\mathbf{p},\mathbf{x},t),t)$ (predictive motion model)
OUTPUT: $\mathbf{p}$ (action parameters)

$\mathbf{p} \leftarrow$ INITIALIZE($\mathbf{x_I}$,$\mathbf{x_T}$) ;
while $\mathbf{x_T} \neq \mathbf{x}(t_I) + \int_{t_I}^{t_F} \mathbf{f_{PMM}}(\mathbf{x}(t),\mathbf{u}(\mathbf{p},\mathbf{x},t),t)\,dt$ **do**
  $\Delta\mathbf{x_T} \leftarrow \mathbf{x_T} - \left[\mathbf{x}(t_I) + \int_{t_I}^{t_F} \mathbf{f_{PMM}}(\mathbf{x}(t),\mathbf{u}(\mathbf{p},\mathbf{x},t),t)\,dt\right]$;
  $\frac{\delta(\Delta\mathbf{x_T}(\mathbf{p}))}{\delta\mathbf{p}} \leftarrow$ ESTIMATECONSTRAINTJACOBIAN($\mathbf{x_I}$,$\mathbf{x_T}$,$\mathbf{u}(\mathbf{p},\mathbf{x},t)$,$\mathbf{f_{PMM}}(\mathbf{x}(t),\mathbf{u}(\mathbf{p},\mathbf{x},t),t)$,$\mathbf{p}$) ;
  $\Delta\mathbf{p} \leftarrow -\left[\frac{\delta(\Delta\mathbf{x_T}(\mathbf{p}))}{\delta\mathbf{p}}\right]^{-1}\Delta\mathbf{x_T}$;
  $\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}$;
end
return $\mathbf{p}$;

ESTIMATECONSTRAINTJACOBIAN($\mathbf{x_I}$,$\mathbf{x_T}$,$\mathbf{u}(\mathbf{p},\mathbf{x},t)$,$\mathbf{f_{PMM}}(\mathbf{x}(t),\mathbf{u}(\mathbf{p},\mathbf{x},t),t)$,$\mathbf{p}$)

INPUT: $\mathbf{x_I}$ (initial state), $\mathbf{x_T}$ (target terminal state), $\mathbf{u}(\mathbf{p},\mathbf{x},t)$ (parameterized action), $\mathbf{f_{PMM}}(\mathbf{x}(t),\mathbf{u}(\mathbf{p},\mathbf{x},t),t)$ (predictive motion model), $\mathbf{p}$ (action parameters)
OUTPUT: $\frac{\delta(\Delta\mathbf{x_T}(\mathbf{p}))}{\delta\mathbf{p}}$ (constraint jacobian estimate)

$m \leftarrow size(\mathbf{p})$;
for $j = 1 : m$ **do**
  $\frac{\delta(\Delta\mathbf{x_T}(\mathbf{p}))}{\delta\mathbf{p}[j]} \leftarrow \frac{\Delta\mathbf{x_T}(\mathbf{p}[j]+\mathbf{h_j},\mathbf{p}) - \Delta\mathbf{x_T}(\mathbf{p})}{h}$;
end
return $\frac{\delta(\Delta\mathbf{x_T}(\mathbf{p}))}{\delta\mathbf{p}}$;

Figure 4.2: Constrained Model-Predictive Trajectory Generation algorithm.

are all very important to the performance of the algorithm.

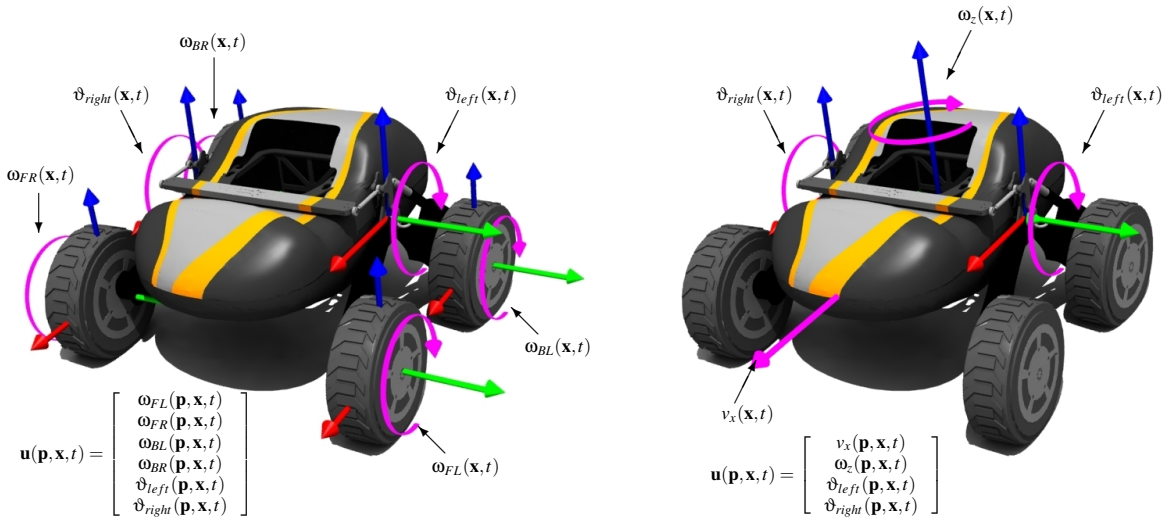
4.1.1 ACTION PARAMETERIZATION

Efficient action parameterization is necessary for effective use of the presented model-predictive trajectory generation techniques. A poor choice of parameterization can lead to high activity inputs, underdetermined solutions that are difficult to initialize and succumb to local optimality, or are not expressive enough to satisfy the boundary state constraints. Selection of the action parameterization for a mobile robot is a function of the mobility system, the boundary state constraints, and the choice of controls or controllers as primitive functions.

MOBILITY SYSTEMS

The mobility system type is arguably most influential in determining the action parameterization for the model-predictive trajectory generator. Mobility systems are identified by the method by which the vehicle changes its position, orientation, and velocities. Commonly applied types of mobility systems include Ackermann, corner-steered, differential drive, and skid-steered vehicles.

The action parameterization design is dependent on the degrees of freedom in the mobility system. Consider the planetary rover depicted in Figure 4.3. Here, the vehicle can directly control the motor torques that determine the four wheel velocities (ω_{FR},ω_{FL},ω_{BR},ω_{BL}) and the two independent chassis sidearm angles (ϑ_{right},ϑ_{left}). One action parameterization for this system involves independent functions for these six variables as shown in 4.3a. While this parameterization is very expressive, it is also very redundant, as wheels can be driven in modes where resulting forces cancel out. Such a parameterization can lead to highly under-constrained actions when the number of parameters exceed the number of boundary state constraints.



(a) an action parameterization for planetary rover with actively articulating chassis based on individual wheel velocities

(b) an action parameterization for planetary rover with actively articulating chassis based on body-frame linear and angular velocities

Figure 4.3: Planetary mobile robot action parameterization. Two parameterizations are shown for a skid-steered mobility system with an actively articulating chassis. In (a), the action profiles are divided between the four wheel velocities and the two suspension arm angles. Alternatively in (b), the action profiles are divided between body-frame linear and angular velocities and the two suspension arm angles.

Alternatively, a action parameterization based on body-frame motion can be defined (Figure 4.3b). Instead of independently controlling motor torques and wheel velocities, actions are determined by mapping the body-frame velocities to consistent wheel velocities. The desired rotational velocity of each independent wheel can be determined by mapping the body-frame motion ($\mathbf{v_{body}}, \boldsymbol{\omega}_{\mathbf{body}}$) to individual wheel velocities (ω_{wheel}):

$$\omega_{wheel} = \frac{v_{wheel}}{r_{wheel}} = \mathbf{v_{body}} + \omega_{\mathbf{body}} \times \mathbf{r_{wheel}} \tag{4.13}$$

Determining the motor torque necessary to achieve these actions is a function of the lower-level controller attempting to follow that velocity. Expressing parameterized actions in body-frame linear and angular velocities enables expressive inputs with a relatively small number of parameters. While it sacrifices some generality to achieve complex maneuvers (such as plowing), it can typically be formulated in a way to effectively generate all desirable actions.

BOUNDARY STATE CONSTRAINTS

The types of boundary state constraints imposed is typically an important aspect of the trajectory generation problem. In instrument placement, reaching a target terminal heading constraint may be required for the point of interest to exist in the end effector workspace. In motion planning, it is desirable to have multiple degrees of continuity between path segments. Vehicles with small curvature rate limits may desire curvature continuity in the search space.

The types of boundary state constraints in the model-predictive trajectory generator determine the minimum number of degrees of freedom in the system required to satisfy the state constraints. In order to generally satisfy terminal boundary state constraints, parameterized actions must possess a minimum number of freedoms equal to or greater than the non-trivial initial and terminal boundary state constraints. Unlike the target terminal boundary state constraint, the full initial state must be known in order to initialize the predictive motion model.

CONTROLS AND CONTROLLERS

The functions that define individual inputs in the parameterized action are also important. Profiles that represent expressive inputs with a minimum of freedoms are ideal representations for parameterized actions. Many different methods of expressing actions by parameterized functions of curvature, angular velocity, and linear velocity have been effectively applied in prior research, including Fourier series, Bézier splines, clothoids, and cubic polynomials. Any of these profiles can be applied in this work as nothing particular is assumed about the shape of the actions in the formulation.

Splines based on controlling knot-points have proven to be an effective profile definition for mobile-predictive trajectory generation. The benefits of such as a parameterization comes from the uniform units and more equally distributed sensitivity. For example, previous work in trajectory generation (Kelly and Nagy, 2003) used polynomial functions of state or time as parameterized actions, including a cubic polynomial of curvature as a function of arc length (s):

$$\kappa(\mathbf{p},\mathbf{x},t) = a + bs(t) + cs(t)^2 + ds(t)^3 \quad \text{for} \quad s_I \leq s(t) \leq s_F \quad \text{and} \quad \mathbf{p} = \begin{bmatrix} a & b & c & d \end{bmatrix}^T \tag{4.14}$$

In such a parameterization, the parameters that define the shape of the action (a,b,c,d) all exhibit different sensitivities based on the size of s. For example, if s is large, a small change in d will cause a much larger change in $\kappa(\mathbf{p},\mathbf{x},t)$ than the same change in a. A better solution is to re-parameterize the action as a spline with equally dis-

tributed knot points (κ_0,κ_1,κ_2,κ_3). Equations 4.16 through 4.19 are determined by solving for the curvature values at four equidistant points along the action:

$$\kappa(\mathbf{p},\mathbf{x},t) = a(\mathbf{p}) + b(\mathbf{p})s(t) + c(\mathbf{p})s(t)^2 + d(\mathbf{p})s(t)^3 \quad \text{for} \quad s_I \leq s(t) \leq s_F \quad \text{and} \quad \mathbf{p} = \begin{bmatrix} \kappa_0 & \kappa_1 & \kappa_2 & \kappa_3 \end{bmatrix}^T \tag{4.15}$$

$$a(\mathbf{p}) = \kappa_0 \tag{4.16}$$

$$b(\mathbf{p}) = -\frac{1}{2}\frac{-2\kappa_3 + 11\kappa_0 - 18\kappa_1 + 9\kappa_2}{s_F - s_I} \tag{4.17}$$

$$c(\mathbf{p}) = \frac{9}{2}\frac{-\kappa_3 + 2\kappa_0 - 5\kappa_1 + 4\kappa_2}{(s_F - s_I)^2} \tag{4.18}$$

$$d(\mathbf{p}) = -\frac{9}{2}\frac{-\kappa_3 + \kappa_0 - 3\kappa_1 + 3\kappa_2}{(s_F - s_I)^3} \tag{4.19}$$

The two functions described in Equations 4.14 through 4.15 are mathematically equivalent. The primary benefit of the spline representation over alternative functions for parameterized actions in model-predictive trajectory generation involves a better distribution of parameter sensitivity. This enables easier and more accurate partial derivative estimation for the constraint Jacobian in the constrained model-predictive trajectory generation formulation.

The choice of control or controller determines whether the trajectory termination time (t_F) is implicitly or explicitly defined. Control-based actions have an explicit definition of the trajectory termination time in the parameter vector as the inputs are executed without feedback. Conversely, controller-based actions may execute until a particular state has been reached. For example, a control may require that a vehicle drives forward at a particular speed for k seconds. A controller may require that the vehicle travels l meters, which can take longer (positive longitudinal slip) or shorter (negative longitudinal slip) than k seconds[1]. Controller-based parameterized actions determine the simulation termination time inside of a predictive motion model integration loop by measuring in a particular state (such as terminal arc-length (s_F) or heading (ψ_F)) has been reached.

4.1.2 Predictive Motion Models

The predictive motion model provides the mapping from actions to state change. It approximates the basic physics of vehicle motion with platform-specific kinematic constraints, rate limits, latency estimates, and environmental interaction models. Accurate predictive motion models have become increasingly important for planning, navigation, and control as mobile robots operate at higher speeds, traverse difficult environments, and function in workspaces shared with humans. It is necessary for a mobile robot to completely understand the consequences of selected actions in order to intelligently and confidently perform tasks.

Physical modeling of mobile robot systems is reasonably well understood (Tarokh and McDermott, 2005). Gen-

[1]Longitudinal slip is defined as a scale factor of the body-frame linear velocity. A positive longitudinal slip increases the realized speed of a vehicle, while a negative longitudinal slip decreases the realized speed.

erating effective predictive motion models for model-predictive trajectory generation is a design challenge between balancing fidelity and computational complexity. Since the runtime of the constrained model-predictive trajectory generation algorithm is roughly linear with the speed of the predictive motion model, it may become necessary to reduce fidelity in order to achieve real-time performance. The predictive motion model may be required to operate many times faster than real-time in applications such as motion planning and navigation where many trajectories need to be generated simultaneously. A high-fidelity Lagrangian dynamics model may provide exceptionally realistic motion prediction, but likely would not be fast enough for real-time operation on current hardware. Conversely, a low-fidelity predictive motion model may simulate quickly, but might not effectively model vehicle motion or environmental interaction enough to guarantee action safety.

This section describes three basic components for an effective, real-time predictive motion model for model-predictive trajectory generation: the mapping from actions to body-frame velocities, the mapping from body-frame velocities to world-frame motion, and the vehicle chassis inversion and interaction. Vehicle chassis and suspension models determine how a vehicle interacts or is configured in the environment. The mapping between actions and body-frame motion models how the vehicle responds and acts to commanded inputs. The transformation to state change via world-frame rates are computed as rigid-body transforms of body-frame linear and angular velocities.

CHASSIS AND SUSPENSION MODEL INVERSION

Modeling of the chassis and suspension is important for mobile robots operating in terrains where the environment can significantly modify the center of gravity and/or the normal wheel forces. For planetary rovers and other slow-moving terrestrial mobile robot systems, simulation of ballistic motion is unnecessary as the vehicle can be assumed to be in a quasi-static configuration. The computational resources required to model these effects would significantly reduce the runtime of the predictive motion model while returning insignificant corrections to the motion estimation. Previous work by Tarokh and McDermott (2005) demonstrated that the chassis configuration for a planetary mobile robot with a rocker-bogie could be efficiently determined by treating it as an optimization problem. Chassis models for planetary rovers simulated in this thesis treat this as a root finding problem, where the chassis parameters and body freedoms that minimize the distance between contact points on the wheels and points on two-and-a-half dimensional height map.

MAPPING ACTIONS TO BODY-FRAME MOTION

All real mobile robot systems do not have ideal transfer functions from actions to state response. Real systems have limitations in state magnitudes, rates, and accelerations. Latency also affects when inputs are observable in real mobile robot platforms. Reasoning about system delay is immensely important when vehicle velocities become significant.

In addition to modeling the way in which inputs are mapped to states, low-level logic regarding how inputs are transfered should be modeled. For mobile robots with exposure to tip-over conditions (field robots, forklifts, planetary rovers, etc . . .), curvature rates are often limited at the lowest levels of the vehicle controllers to prevent the robot from entering dangerous conditions. Modeling the response of these subsystems is often important in high speed mobile robot applications to provide realistic motion predictions for the model-predictive trajectory generator.

MAPPING BODY-FRAME MOTION TO WORLD-FRAME RATES

While the previous section described the mapping from actions to body-frame velocities and accelerations, motion prediction is the process of transforming and integrating body-frame velocities into world-frame state changes. For a rigid body moving in the plane, the state rate equations are observed by rotating the body-frame velocities by the body yaw [2]:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} c(\psi) & -s(\psi) & 0 \\ s(\psi) & c(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix} \tag{4.20}$$

The more general three-dimensional rigid body motion transform can be determined by transforming the body-frame linear and angular velocities into the world frame. The state rate equations are now coupled and non-integrable in closed form for all non-trivial solutions:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} c(\psi)c(\theta) & c(\phi)s(\theta)s(\phi) - s(\psi)c(\phi) & c(\phi)s(\theta)c(\phi)s(\psi)s(\phi) \\ s(\psi)c(\theta) & c(\psi)c(\phi) + s(\psi)s(\theta)s(\phi) & s(\psi)s(\theta)c(\phi) - c(\psi)s(\phi) \\ -s(\theta) & c(\theta)s(\phi) & c(\theta)c(\phi) \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \tag{4.21}$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & t(\theta)s(\phi) & -t(\theta)c(\phi) \\ 0 & c(\phi) & s(\phi) \\ 0 & s(\phi)/c(\theta) & c(\phi)/c(\theta) \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{4.22}$$

Simplifying the rate mapping kinematics can be important to achieve real-time performance. If the world can be assumed to be relatively planar, the simpler transforms in Equation 4.20 can be used to estimate motion. Not evaluating the trigonometric functions of attitude at every point along the path enables modeling of other aspects of vehicle mobility or increases the motion simulation speed. The simpler planar model was effectively used in a predictive motion model built for urban navigation in Urmson et al. (2008) whereas the more general three-dimensional transformation in Equations 4.21 and 4.22 were used for rough terrain trajectory generation in Howard and Kelly (2007).

4.1.3 INITIALIZATION FUNCTION

For continuum optimization and gradient-based techniques to work effectively, a good initial estimate of the solution parameters must be provided. The initialization function is a method that determines an initial set of parameters for a specific action parameterization, predictive mobility model, and boundary state constraint set. Generally the initialization function operates as an approximate mapping between state change and action parameters. Determining an initial set of parameters in the region of convergence and as close to the continuum solution as possible is beneficial since the runtime of the algorithm is proportional to the initial boundary state constraint error. If the predictive motion

[2] for equations 4.20 through 4.22, $s(x) = sin(x)$,$c(x) = cos(x)$,$t(x) = tan(x)$

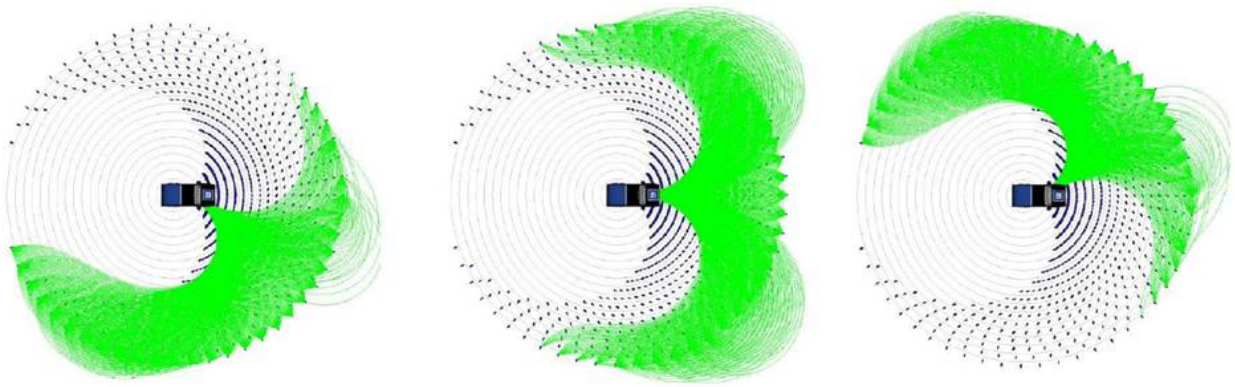


Figure 4.4: Slices of a five-dimensional (Δx,Δy,$\Delta\psi$,κ_I,v_x) action parameter lookup table initialization function for varying initial curvature at a constant velocity. At this speed, all elements in the lookup table are not necessarily reachable from the curvature rate limitations of the predictive motion model.

model does not vary significantly during the course of operation, precomputing a portion of the state change and action mapping can significantly improve the runtime and speed of convergence for the technique.

Developing processes for generating this mapping can be a complicated, cumbersome, and computationally expensive process. Machine learning techniques (such as neural networks) have been applied to the problem of shape estimation of the mapping between state change to action parameters. The most effective applied initialization function method to date involves parameter lookup tables. These data structures store a coarse mapping of the vehicle state space (positions, headings, curvatures, velocities, etc ...) and are mapped to a set of action parameters that satisfy Equation 4.2. The parametric representation of actions (and subsequent trajectories) actually is key in efficient storage of the mapping. Instead of logging many intermediate trajectory positions, headings, velocities, curvatures, and inputs, the entire trajectory can be reconstructed at arbitrary resolution using only the parameters that define the action shape and the predictive motion model. Action parameter lookup table generation can itself be recursive with the constrained model-predictive trajectory generation algorithm, where sampling in the action parameter space can be used to initialize continuum search for parameters that occupy cell values. Determining the trade-off between the action parameter lookup table dimensionality, resolution, and sampling is a problem that requires intuition about the vehicle mobility and workspace and it is typically an iterative process. Smooth interpolation on a dense parameter lookup table can sometimes replace continuum search in parameter space if the approximate mapping satisfies the boundary state constraint thresholds.

An example of the action parameter lookup table designed for an autonomous automobile is shown in Figure 4.4. Slices of a five-dimensional structure indexed by relative position, heading, initial curvature, and constant velocity are shown to represent the variety of feasible solutions stored in the table. The particular values shown in Figure 4.4 are for a high constant velocity with varying initial curvature.

Notice that at this speed, portions of the state space are not reachable within time constraints. The curvature rate

limits of the predictive mobility model prevent trajectories from being generated far away from the original line of motion defined by the initial curvature. An added benefit of an action parameter lookup table is that it provides an estimate of the feasible solution space for a particular model of vehicle mobility. If boundary state constraints were queried in regions where no parameter lookup table exists, it could either be used to return the closest existing solution at the lookup table (often at the limit of vehicle mobility) or notify the system that no (reasonable) solution exists.

4.1.4 EXAMPLES AND EXPERIMENTS

In this section, several simulation and field experiments demonstrate the effectiveness for the constrained model-predictive trajectory generation technique to generate parameterized actions subject to an arbitrary predictive motion model that satisfies continuum boundary state constraints. This section begins with constrained model-predictive trajectory generation applied to determining actions for continuum boundary state constraints in rough terrain. Later, an example of instrument placement with impaired mobility is presented, which is handled by simply modifying the predictive motion model to simulate the disturbance. A third experiment involves local motion planning search space generation using state-space sampling techniques and the constrained model-predictive trajectory generator to provide connectivity. The last experiment in this section involves chassis optimization for planetary rover motion, where an optimizer is built on top of the constrained model-predictive trajectory generator to minimize exposure to tip-over.

MODEL-PREDICTIVE TRAJECTORY GENERATION ON UNEVEN TERRAIN

To evaluate the constrained model-predictive trajectory generation algorithm's ability to determine actions that satisfy boundary state constraints on uneven terrains, first the solution for flat terrain is found. In this example, a mobile robot attempts to move from an initial state to a target terminal state eight meters forward and four meters to the left without a constraint on terminal heading:

$$\mathbf{x_I} = \mathbf{x}(t_I) = \begin{bmatrix} x(t_I) & y(t_I) & \psi(t_I) & \kappa(t_I) & v_x(t_I) \end{bmatrix}^T = \begin{bmatrix} 0.0\text{m} & 0.0\text{m} & 0.0\text{rad} & 0.0\frac{\text{rad}}{\text{m}} & 0.1\frac{\text{m}}{\text{sec}} \end{bmatrix}^T \tag{4.23}$$

$$\mathbf{x_T} = \mathbf{x}(t_F) = \begin{bmatrix} x(t_F) & y(t_F) \end{bmatrix}^T = \begin{bmatrix} 8.0\text{m} & 4.0\text{m} \end{bmatrix}^T \tag{4.24}$$

The heading, curvature, and other target terminal state constraints are not included in this example to reduce the visualization complexity. As latter experiments in this section demonstrate, the techniques generally extend beyond satisfaction of two constraints. Also note that in Equation 4.23 the full initial state must be known to properly initialize the predictive motion model.

The parameterized actions for this experiment include controllers as a constant function of velocity and a first-order spline of curvature. Each function determines the value based on the arc-length of the execution motion. The parameters for this action are listed in Equation 4.27:

$$v_x(\mathbf{p}, \mathbf{x}, t) = v_0 \tag{4.25}$$

$$\kappa(\mathbf{p},\mathbf{x},t) = \kappa_0 + \frac{\kappa_1 - \kappa_0}{s_F - s_I}(s(t) - s_I) \tag{4.26}$$

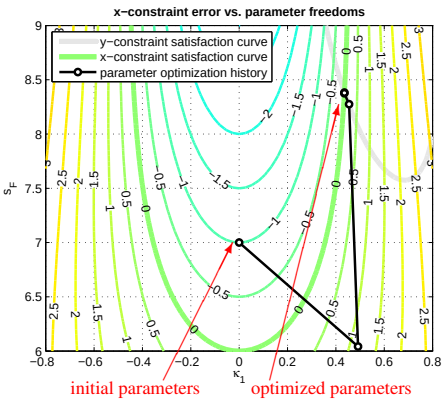
$$\mathbf{p} = \begin{bmatrix} v_0 & \kappa_0 & \kappa_1 & s_I & s_F \end{bmatrix}^T \tag{4.27}$$

Several of these action parameters are trivially satisfied if continuity between state and actions is desired. The initial velocity, curvature, and distance traveled values are all known, leaving two free action parameters, resulting in a system with a square constraint Jacobian:

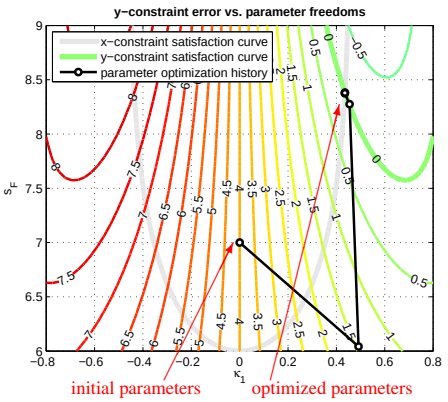
$$\mathbf{p} = \begin{bmatrix} \kappa_1 & s_F \end{bmatrix}^T \tag{4.28}$$

Figure 4.5 shows the planned trajectory and the continuum search for free action parameters that satisfy the boundary state constraints on flat terrain. The contour plots represent the terminal position constraint error for provided values of action parameter values. The optimization history follows the numerically estimated gradients of the constraint error until a solution satisfies both constraints is reached. This particular problem has only two degrees of freedom (κ_1,s_F), so the solution for the parameters occurs at the intersection of the two constraint satisfaction curves. For problems with three degrees of freedom and three constraints, the solution occurs at the intersection of three zero constraint error surfaces in a three-dimensional constraint error volume. Under-determined problems (such as those with more action parameters than constraints) can have multiple intersections of the constraint curves, surfaces, or hypersurfaces in n-dimensional space.

(a) flat terrain trajectory



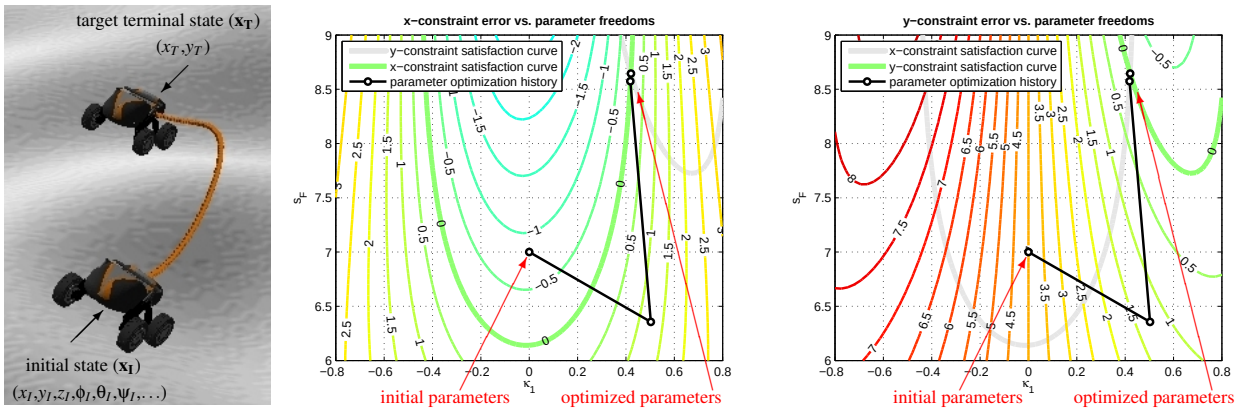
(b) x constraint error vs parameter freedom (κ_1,s_F)



(c) y constraint error vs parameter freedom (κ_1,s_F)

Figure 4.5: Trajectory generation on flat terrain. Here, the parameterized controls start from a initial guess of parameterized freedom values and are iteratively optimized until they satisfy the terminal position boundary state constraints. The solution that satisfies the terminal boundary state constraints within sub-millimeter accuracy is determined with free action parameters $[\kappa_1, s_F]^T = \left[0.435\frac{\text{rad}}{\text{m}}, 8.382\text{m}\right]^T$.

A second experiment using the same boundary state constraints, parameterized actions, and predictive mobility model was performed on uneven terrain to show the solution and search space differences. Figure 4.6 shows the newly generated trajectory, the two constraint error contours, and the parameter optimization history for uneven terrain. Although the location and intersection of the zero constraint error lines shift, a solution is efficiently determined using the exact same technique. The numerical gradient estimates points the optimization to follow the new contours of the parameter-constraint error space to generate a solution. Optimization and root-finding methods are the best methods for determining these actions as discretized sampling in action space would be very expensive to evaluate.

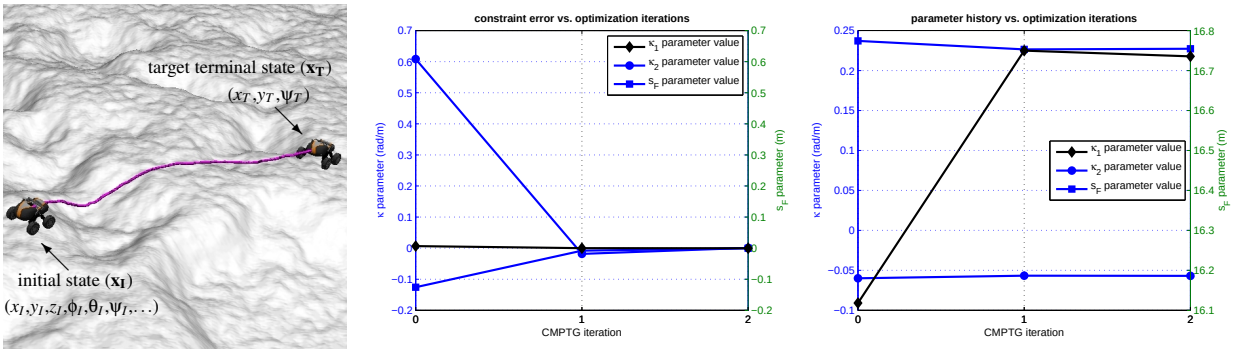


(a) uneven terrain trajectory (b) x constraint error vs parameter freedom (κ_1, s_F) (c) y constraint error vs parameter freedom (κ_1, s_F)

Figure 4.6: Trajectory generation on uneven terrain. Here, the parameterized controls start from a initial guess of parameterized freedom values and are iteratively optimized until they satisfy the terminal position boundary state constraints. Notice that the search space and the solution differs from the example showed in Figure 4.5, despite the invariant boundary state constraints. The solution that satisfies the terminal boundary state constraints within sub-millimeter accuracy is determined with free action parameters $[\kappa_1, s_F]^T = \left[0.420\frac{\text{rad}}{\text{m}}, 8.645\text{m}\right]^T$.

This technique extends to more boundary state constraints, expressive parameterized actions, and complex terrains. Figure 4.7 shows another examples that requires a specific position and heading at the terminal boundary state in more difficult terrain than the previous example in Figure 4.6. A more expressive curvature input based on a second-order spline controller as a function of path distance was used to provide the extra degrees of freedom to satisfy the boundary state constraints. The same action parameterization is used later in Equation 4.36 in an unconstrained optimization model-predictive trajectory generation example. From an initialization function that approximates the flat-terrain solution, the constrained model-predictive trajectory generation algorithm determined an action with millimeter predicted terminal state constraint error that reasoned about the terrain shape effects in two iterations of the parameter optimization. The determined action operates for a longer distance (to account for the added trajectory length) and makes small corrections to the latter two curvature spline points to satisfy the position and heading boundary state constraints.

These experiments demonstrated that the shape of the terrain influences the generated trajectory and the actions



(a) view of constrained model-predictive trajectory generation in complex terrain

(b) constraint error vs constrained model-predictive trajectory generation iteration

(c) parameter freedom values vs constrained model-predictive trajectory generation iteration

Figure 4.7: An example of constrained model-predictive trajectory generation with more boundary state constraints, expressive actions, and complex terrain. The environment, boundary state constraints, and generated trajectory for the example are shown in (a). The plots in (b) and (c) show the terminal state constraint error (x, y, and ψ) and parameterized freedom values (κ_1, κ_2, and s_F) as a function of constrained model-predictive trajectory generation iteration.

necessary to reach the target terminal boundary state constraints. This is because steering acts in a body frame that is rotated by the chassis interaction with the environment. Mapping the original solution to the new constraint error surfaces show that the solutions would not satisfy the continuum boundary state constraints. Including better approximations of wheel/terrain interaction and wheel slip often result in a more significant difference between the flat terrain and uneven terrain solutions (Howard and Kelly, 2007). The same numerical techniques can be applied to those problems simply by updating the predictive motion model to reflect the differences in simulated actions.

PREDICTIVE COMPENSATION OF IMPAIRED MOBILITY

Mobility systems degrade over time. The ability for mobile robots to recover from impaired mobility is crucial as systems move from controlled laboratory experiments to real-world operations. Just as the constrained model-predictive trajectory generation algorithm can reason about and correct for environmental interaction, changes in the internal mobility system can be modeled as well.

Of particular interest in the application of planetary exploration. If the mobility system degrades during a mission, it is unlikely that the robot could be fixed to regain full functionality. It would be important for the mobility system to reason about its limited capability at planning time in order to select actions that efficiently accomplish mission objectives and keep the platform safe.

Planetary rovers exploring distant worlds are a prime example of where reasoning about impaired mobility can potentially improve the effectiveness of an autonomous mobile robot platform. One example of this includes the

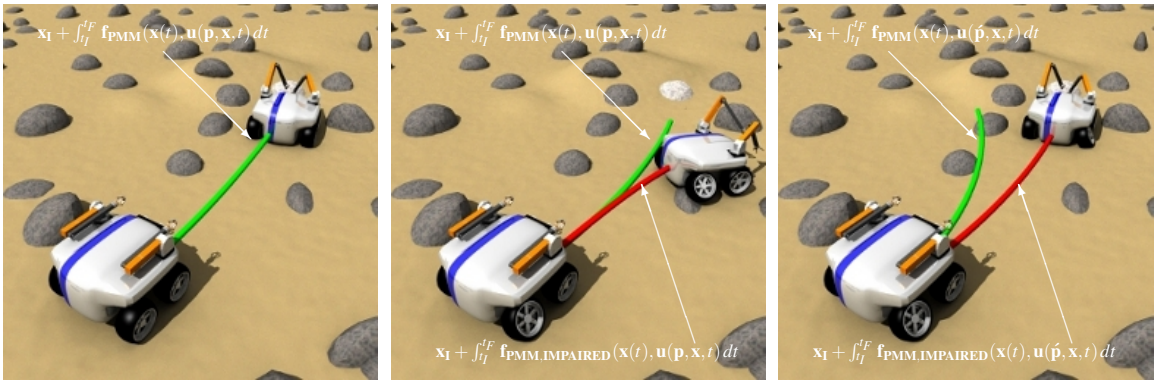
Mars Exploration Rovers (MER) mission, where Opportunity's front-right drive motor became broken (Kim et al., 2009). One approach is to have human operators compensate for these effects by modifying the generated actions. Another possibly more efficient alternative is to design an autonomy system that could reason about these effects and determine corrective behaviors itself. Reasoning about impaired or modified mobility will become increasingly important for other missions where humans cannot be "in the loop", such as crater and deep space exploration, which will require increased levels of robot autonomy.

There are several approaches for solving the impaired mobility issue. One common approach is to design a high-rate feedback controller to compensate for the unmodeled disturbances in the system. This requires fast and reliable state feedback, which for terrestrial mobile robots often comes in the form of the Global Positioning Systems (GPS). For extra-terrestrial, indoor, or underwater operations, GPS is unavailable. Other approaches such as visual odometry must be applied to accurately estimate the current state of the mobile robot. The alternative is increase the action confidence by simulating inputs with a higher fidelity predictive motion model that reasons about the impaired mobility system. With the model-predictive trajectory generator, actions can be simulated as well as corrected if the vehicle deviates from the intended target. While such an approach would likely not entirely replace feedback control, better motion prediction and action generation can reduce the path following error and the required update rate of state feedback.

To illustrate the differences between the two approaches, a typical instrument placement problem is depicted in Figure 4.8 while simulating the effects of an impaired drive wheel. In Figure 4.8a, a mobile robot plans and executes an action under the assumption of normal mobility. Since the effects of the impaired mobility system were not modeled in the motion plan, the vehicle pulls to the right and would need to plan a more complex recovery action to reacquire the instrument placement target. Figure 4.8c, the mobile robot utilizes the higher fidelity predictive motion model and plans an action which oversteers to the left, knowing that the result of the action will compensate the unbalanced force of the impaired drive wheel.

To illustrate the latter concept, several scenarios for a typical instrument placement problem are shown in Figure 4.8. In Figure 4.8a, a mobile robot plans and executes an action under the assumption of normal mobility. Since the predictive motion model provides a reasonable estimate of the vehicle trajectory, the executed actions lead the vehicle towards the goal. A different scenario is depicted in Figure 4.8b, where a vehicle plans an action based on normal motion and executes the action with an impaired mobility system. Since the effects of impaired mobility were not modeled in the generated action, the vehicle does not accurately track the generated trajectory. In this situation, a more complex recovery action would need to be generated to reacquire the trajectory and reach the target terminal state. A third situation in shown in Figure 4.8c, where a better impaired mobility model is used to generate the action inside of a model-predictive trajectory generator that overcompensates by steering from the motion executed in Figure 4.8b.

The situation depicted in Figure 4.8 was tested on the Rocky 8 planetary rover prototype at the Jet Propulsion Laboratory (Pivtoraiko et al., 2008). In this experiment, two predictive motion models were developed. First, a simple predictive motion model was developed for the generally flat environment and soil properties. Second, an engineered wheel slip model was determined for the same terrain for an impaired mobility system with a disabled rear right



(a) prediction and execution of a trajectory with a normal mobility system

(b) prediction of a trajectory with a normal mobility system with execution under an impaired mobility system

(c) prediction and execution of a trajectory with an impaired mobility system

Figure 4.8: Overview of predictive compensation for impaired mobility experiment. Under normal mobility, a set of parameters ($\mathbf{p}$) define an action that move a mobile robot from an initial state to a target terminal state. In the event of a disturbance to the predictive motion model, the shape of the resulting path can change. A corrected set of action parameters ($\acute{\mathbf{p}}$) can be found by regenerating the trajectory with the new, updated predictive motion model.

drive motor. First, an action was planned and executed using the constrained model-predictive trajectory generation algorithm that reliably reached the target terminal state without relying on feedback control (Figure 4.9a). Figure 4.9b shows a second experiment using the action determined from the previously generated trajectory. The disturbance in the mobility system prevents the vehicle from reaching the intended terminal state and now must generate an entirely new action. The vehicle deviated from the intended path by dragging the disabled wheel (Figure 4.9c), incurring an unbalanced force that caused the vehicle to turn unintentionally.

To show the benefits and flexibility of model-predictive trajectory generation, a new action was planned and executed using the constrained model-predictive trajectory generation algorithm that reliably reached the target terminal state without relying on state feedback control (Figure 4.9d). Like the illustrated example in Figure 4.8c, the technique automatically generated an action that over-steered the vehicle harder to left to compensate for the unbalanced force. This example shows that if an accurate predictive motion model can be engineered or identified on-line, model-predictive trajectory generation techniques can be used to modify the actions to predictively compensate for disturbances. Since the model is inverted numerically in real-time, the trajectory generation algorithm adapts automatically to a changing model.

STATE-SPACE SAMPLING LOCAL MOTION PLANNING SEARCH SPACES

An important application of the constrained model-predictive trajectory generation algorithm involves the synthesis of local motion planning search spaces. Local motion planning search spaces are control sets sampled by mobile robots

(a) execution of a trajectory generated with a normal mobility system model under normal mobility

(b) execution of a trajectory generated with a normal mobility system model under impaired mobility

(c) creating an impaired mobility system by disabling the right rear drive motor

(d) execution of a trajectory generated with a impaired mobility system model under impaired mobility

Figure 4.9: Predictive compensation for impaired mobility experiment. In (a), a prototype planetary rover drives forward several five meters for an instrument placement task under normal mobility. The rear right drive motor is disabled in (c), which results in the execution of the motion plan shown in (b). The resulting execution of a trajectory generated using a predictive mobility model that models the effects of the impaired wheel is shown in (d).

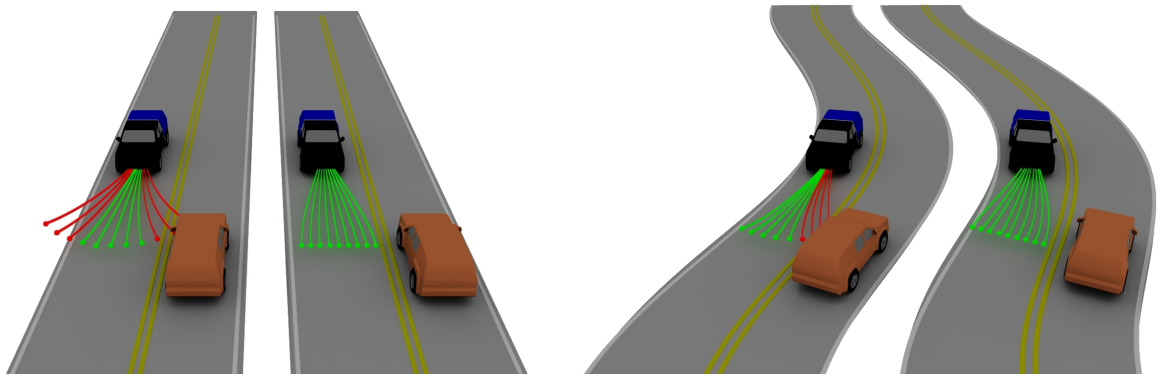
to navigate effectively within their perceptual horizon.

Feasibility, efficiency, and expressiveness are important factors in the design of local motion planning search spaces. Action-space sampling techniques (Kelly and Stentz, 1998) are inherently feasible because they sample in the action space of the vehicle. It is however difficult to control the state-space response and the related efficiency of the search space using such methods. Expressive actions are also difficult to produce as sampling in high-dimensional parameterized action spaces can be prohibitively expensive.

An alternative approach involves sampling in the terminal state space of vehicle motion and using the constrained model-predictive trajectory generator to determine the connective actions. This method provides the feasibility guarantees of action-space sampling techniques as it satisfies the predictive motion model constraints. It also enables more efficient and expressive search spaces to be constructed as sampling in the terminal state space enables finer control state-space response. State-space sampling local motion planning search spaces also can exploit global guidance and environmental shapes by controlling the distributions of boundary states towards regions where minimum-cost solutions are most probable.

Consider the mobile robot navigation problem posed in Figure 4.10. Input-space sampling techniques search low-order actions (e.g. arcs) to sample the space of feasible actions emanating from the initial state of the vehicle. The alternative state-space sampling technique (Howard et al., 2008) selects a set of target terminal states and uses a trajectory generation technique to determine the connective actions that satisfy the boundary state constraints. For in-lane navigation, it is beneficial to determine actions that reach a uniform distance along the road that reach a terminal heading aligned with the tangent of the road shape. To achieve this with input-space sampling techniques, more expressive actions of variable length would have to be sampled.

Formally, the state-space sampling local motion planning search space is generated by first determining the set of n boundary state pairs ($\mathbf{x_N}$):



(a) a comparison of action-space and state-space sampled local motion planning search spaces on a straight road

(b) a comparison of action-space and state-space sampled local motion planning search spaces on a curved road

Figure 4.10: A comparison of search spaces based on action-space and state-space sampled local motion planning search spaces on curved and straight road segments. While portions of the action-space sampling method violate the boundaries of the road shape, the state-space sampling technique, where terminal boundary states are sampled with a perpendicular offset from a forward point in the lane center, generates obstacle avoidance maneuvers that respect the lane boundary.

$$\mathbf{x_N} = \begin{bmatrix} \mathbf{x_I} \\ \mathbf{x_T} \end{bmatrix} = \begin{bmatrix} \mathbf{x_{I_0}} & \mathbf{x_{I_1}} & \dots & \mathbf{x_{I_n}} \\ \mathbf{x_{T_0}} & \mathbf{x_{T_1}} & \dots & \mathbf{x_{T_n}} \end{bmatrix} \tag{4.29}$$

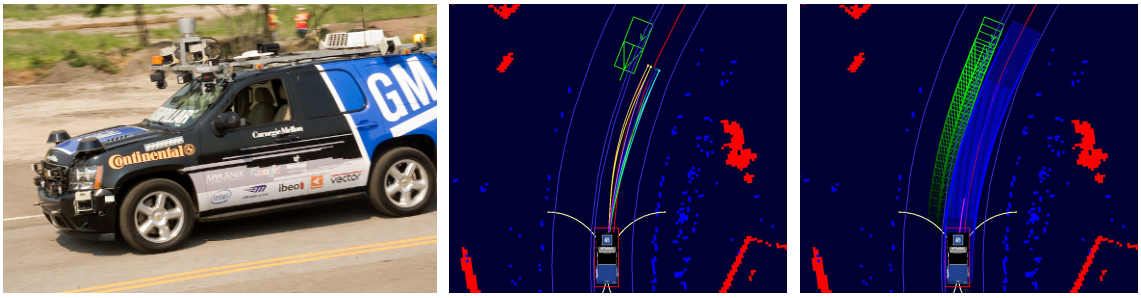
The problem becomes that of determining and searching the actions that satisfy the boundary state pairs and predictive motion model:

$$\mathbf{u_N}(\mathbf{p},\mathbf{x},t) = \begin{bmatrix} \mathbf{u_0}(\mathbf{p_0},\mathbf{x},t) & \mathbf{u_1}(\mathbf{p_1},\mathbf{x},t) & \dots & \mathbf{u_n}(\mathbf{p_n},\mathbf{x},t) \end{bmatrix} \tag{4.30}$$

In addition to determining the terminal boundary state sampling as a function of the environmental constraints (e.g. road shape), the terminal boundary states can be sampled at a uniform horizon with a density determined by exploiting global guidance information.

The state-space sampling local motion planning search space algorithm has been extensively applied in two mobile robot applications. In Urmson et al. (2008), Boss utilized an extension of the state-space sampling technique that exploited global guidance from a mission planner to determine the desired lane of travel. This enabled Boss to effectively search a number of expressive actions that were not only within the desired lane (Figure 4.11), but ones that evaluated the costs associated with the lane change maneuvers. In addition to sampling in the terminal boundary state placement constraint, actions were sampled in the space of initial curvature command.

By varying the initial value of the curvature input spline, the local motion planner was able to produce actions which smoothly transition from the current state (smooth actions) and aggressively swerved at the beginning of the



(a) Boss, Tartan Racing's entry in the 2007 DARPA Urban Challenge

(b) Local motion planning search space using state-space sampling with model-predictive trajectory generation

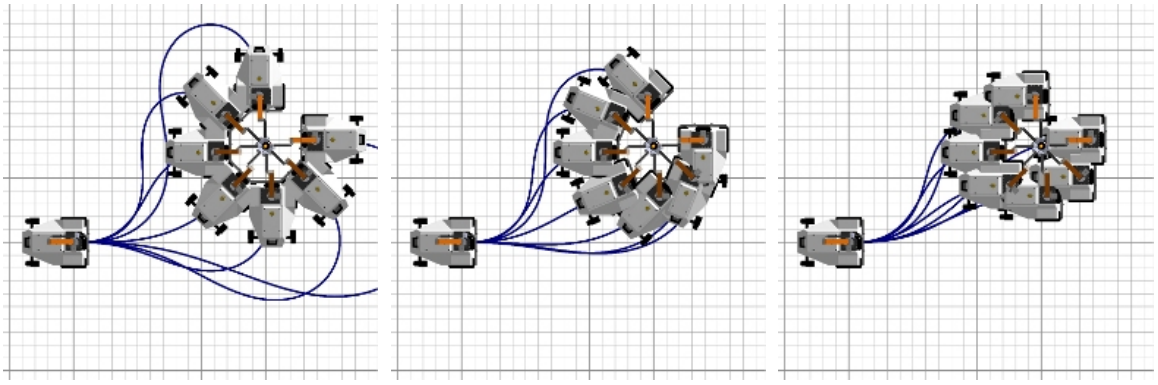
(c) An action is selected that keeps the vehicle within the lane while avoiding the oncoming vehicle

Figure 4.11: Model-predictive trajectory generation applied for state-space sampling local motion planning search space generation in the DARPA Urban Grand Challenge. This technique was used for in-lane navigation, generating safe, feasible actions that satisfy a vehicle-specific predictive motion model and kept the autonomous automobile within the road boundaries.

action (sharp actions). Smooth and sharp actions were generated, evaluated, and used by Boss's local motion planner to navigate on multi-lane roads, highways, and parking lots leading to the DARPA Urban Challenge. Boss finished the 85 kilometer course is four hours, ten minutes, and twenty seconds, averaging a speed of 22.5 kilometers per second, earning first place in the DARPA Urban Challenge. The constrained model-predictive trajectory generator was used to generate the local motion planning search space for Boss at 10 Hz throughout the event.

Secondly, it was applied to the problem of field robot navigation in cluttered, open environments in Howard et al. (2007). Target terminal states were sampled in position and heading at a uniform horizon around the vehicle biased by the Field D* path cost. The constrained model-predictive trajectory generator was used to determine expressive, feasible actions that satisfied the boundary state constraint pairs. Simulation experiments at varying speeds and obstacle densities showed that the improved search space expressiveness improved the experienced cost (a metric related to risk of mission failure) by an average of 24.8%. The subsequent field experiments demonstrated a similar capacity to follow paths and dodge obstacles at speeds up to 12 meters per second in off-road environments.

One last application of state-space sampling local motion planning search space design involves automatic base placement for mobile manipulators. Motion planning for mobile manipulators is sometimes an under-determined problem as only the location of the end effector is constrained. For mobile robots with unconstrained mobility systems and manipulators with multiple degrees of freedom, there are many potential base placement and arm configurations that satisfy the instrument placement problem. One solution is to generate a search space of actions that satisfy the end effector constraints by sampling in the terminal state space of base placements. This imposes artificial constraints on the terminal state to efficiently generate a series of actions that can be quickly searched for an acceptable solution. A view of this is shown in Figure 4.12, where a number of candidate actions to place a manipulator at a specific point in space are determined to define a potential action space. This technique was applied used in Anderson et al.



(a) sampling in the space of terminal states lined up with the end effector constraint

(b) sampling in the space of terminal states slightly offset with the end effector constraint

(c) sampling in the space of terminal states lined up with the initial heading

Figure 4.12: Automatic base placement for mobile manipulators. A search space to determine the optimal base placement, manipulator configuration, and trajectory for a mobile manipulator can be generated by sampling in the space of candidate terminal state constraints.

(2008) to determine a coordinated motion and articulation action for a modified LAGR mobile robot with an attached manipulator (LAGR-EOD).

CONFIGURATION OPTIMIZATION FOR PLANETARY ROVERS WITH AN ACTIVE CHASSIS

For mobile robots with actively articulating chassis, the quality of the path may not entirely be a function of how the vehicle steers and drives through the environment. The posture and configuration of the vehicle play an active role in determining whether a trajectory is dangerous or safe. Reasoning about chassis configuration during trajectory generation enables mobile robots with actively articulating chassis to act intelligently in challenging terrains.

A method for determining minimum-cost actions by exploiting chassis configuration is presented in Furlong et al. (2009). In this approach, a optimization routine is implemented on top of the constrained model-predictive trajectory generator that modifies the values of the chassis sidearm angle inputs to minimize integrated attitude along a trajectory:

$$J(\mathbf{x}(t),\mathbf{u}(\mathbf{x},t),t_I,t_F) = \int_{t_I}^{t_F} (\phi(t))^2 + (\theta(t))^2\, dt \tag{4.31}$$

Figure 4.13 exhibits the application of the actively articulating chassis optimization technique to generate trajectories for a set of boundary state constraints in three different environments. For effective visualization of the results, the number of free parameters in the chassis optimization was limited to two. For the duration of the action, the vehicle configuration was independently controlled by two constant chassis sidearm angle profiles:

$$\vartheta_{left}(\mathbf{x},t) = \vartheta_{left_0} \tag{4.32}$$

$$\vartheta_{right}(\mathbf{x},t) = \vartheta_{right_0} \tag{4.33}$$

The tiered optimization modified the parameters defining the chassis sidearm angles ($\vartheta_{left_0}, \vartheta_{right_0}$) to minimize the integrated attitude of the trajectory. A realistic predictive motion model that limited the magnitude and rate of chassis sidearm angle meant that the commanded actions were not instantaneously realized in the vehicle motion. In all three experiments, the vehicle is initialized with equal right and left sidearm angles. Figures 4.13a, 4.13b, and 4.13c show the optimized trajectories for a planetary rover operating on three increasingly rough terrains. Figures 4.13d, 4.13e, and 4.13f illustrate the shape of the cost surface as a function of chassis action parameters. The chassis action parameter optimization history is overlaid on the cost surfaces to show how the technique was able to iteratively reduce the integrated attitude of the trajectory until the locally optimal configuration was reached.

Another interesting feature of Figures 4.13d, 4.13e, and 4.13f is the behavior near the edges of the cost surface plots. As the cost of parameterized actions are queried beyond the feasible boundary of the chassis configuration, the costs remain at the maximum or minimum limits of the sidearm angle.

4.2 Unconstrained Optimization Model-Predictive Trajectory Generation

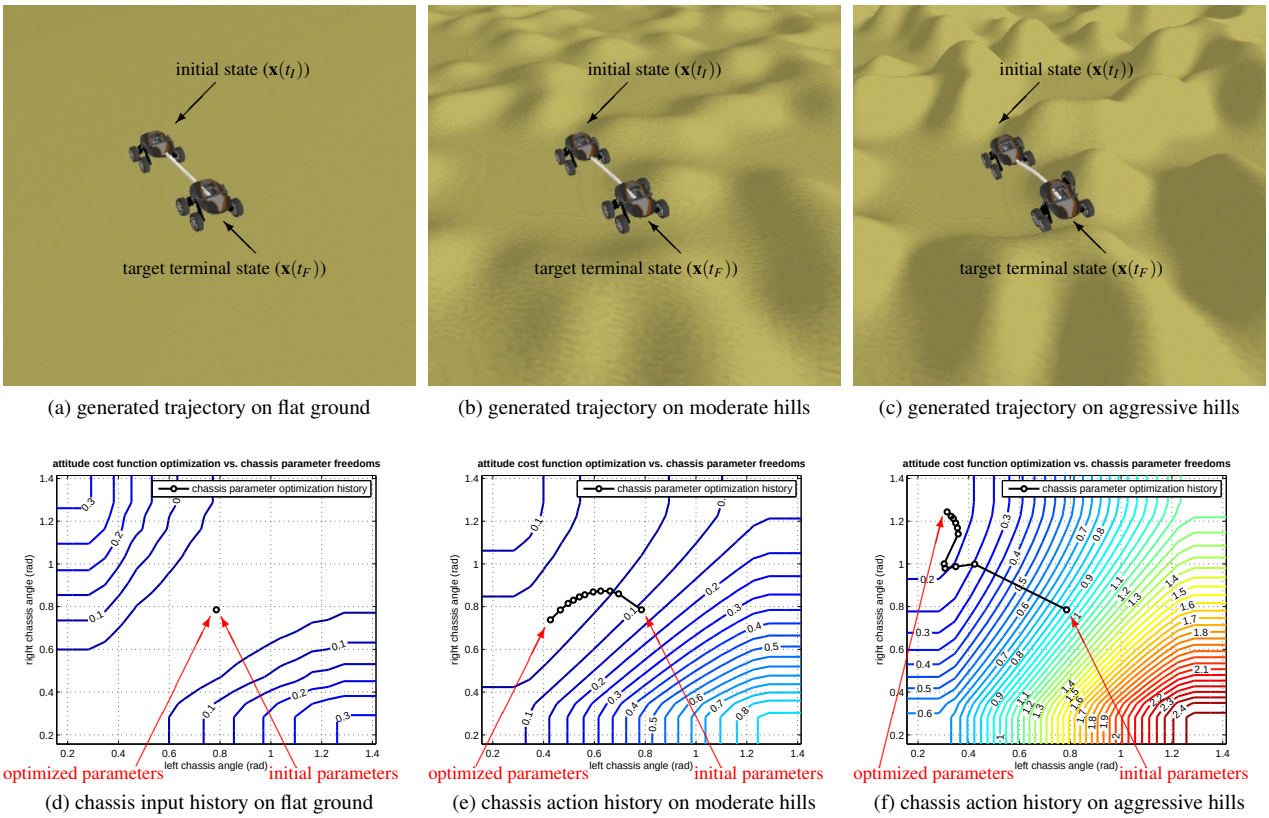
The second class of model-predictive trajectory generation problems are ones that minimize cost in place of satisfying state constraints. Related to optimal control, unconstrained optimization model-predictive trajectory generation is the problem of generating a set of actions that minimize some utility from an initial state and satisfy a predictive motion model without explicit terminal or intermediate state constraints. This is an important problem in mobile robot navigation and control, where a vehicle must generate safe maneuvers for a fixed period of time (Figure 4.14) towards achieving some task (like following a path). The algorithm presented in this section is a key component of the receding horizon model-predictive control algorithm described in Chapter 6.

In the context of mobile robot trajectory generation, cost is described as a cost functional ($J(\mathbf{x}(t),\mathbf{u}(\mathbf{x},t),t_I,t_F)$) consisting of an boundary state cost ($\Phi(\mathbf{x}(t),t_I,t_F)$) and an integrated Lagrangian ($\mathcal{L}(\mathbf{x}(t),\mathbf{u}(\mathbf{x},t),t_I,t_F)$):

$$J(\mathbf{x}(t),\mathbf{u}(\mathbf{x},t),t_I,t_F) = \Phi(\mathbf{x}(t_I),t_I,\mathbf{x}(t_F),t_F) + \int_{t_I}^{t_F} \mathcal{L}(\mathbf{x}(t),\mathbf{u}(t),t)dt \tag{4.34}$$

The cost represents a weighted combination of penalties that are properties of a given state or action. It may include instantaneous path deviation, energy consumption, wheel slip, mobility risk, obstacle proximity, or any other quantity of interest. Cost is most efficiently evaluated when computed simultaneously with the simulation of the predictive motion model.

The unconstrained optimization model-predictive trajectory generation problem minimizes the utility function by modifying the actions of the vehicle:



(a) generated trajectory on flat ground

(b) generated trajectory on moderate hills

(c) generated trajectory on aggressive hills

(d) chassis input history on flat ground

(e) chassis action history on moderate hills

(f) chassis action history on aggressive hills

Figure 4.13: The cost minimization and the resulting sidearm elements for the tiered optimization trajectory generation simulation experiment. Specifically notice that the right sidearm active chassis element has stopped optimizing in 4.13f because the sidearm angle limit (0.95 *rad*) has been reached.

$$\begin{aligned} \text{minimize:} \quad & J(\mathbf{x}(t),\mathbf{u}(\mathbf{x},t),t_I,t_F) \\ \text{subject to:} \quad & \dot{\mathbf{x}} = \mathbf{f}_{\mathbf{PMM}}(\mathbf{x}(t),\mathbf{u}(\mathbf{x},t),t) \\ & \mathbf{x}(t_I) = \mathbf{x_I} \end{aligned} \tag{4.35}$$

As in Section 4.1, parameterized inputs are used to reduce the general optimal control problem to continuum search in a reduced action space. The unconstrained optimization model-predictive trajectory generation problem involves determining the set of parameters ($\mathbf{p}$) that minimize the utility functional subject to the predictive motion model and initial state constraints.

The unconstrained optimization model-predictive trajectory generation algorithm using a simple gradient descent optimization algorithm is shown in Figure 4.15. As the constrained model-predictive trajectory generation technique,

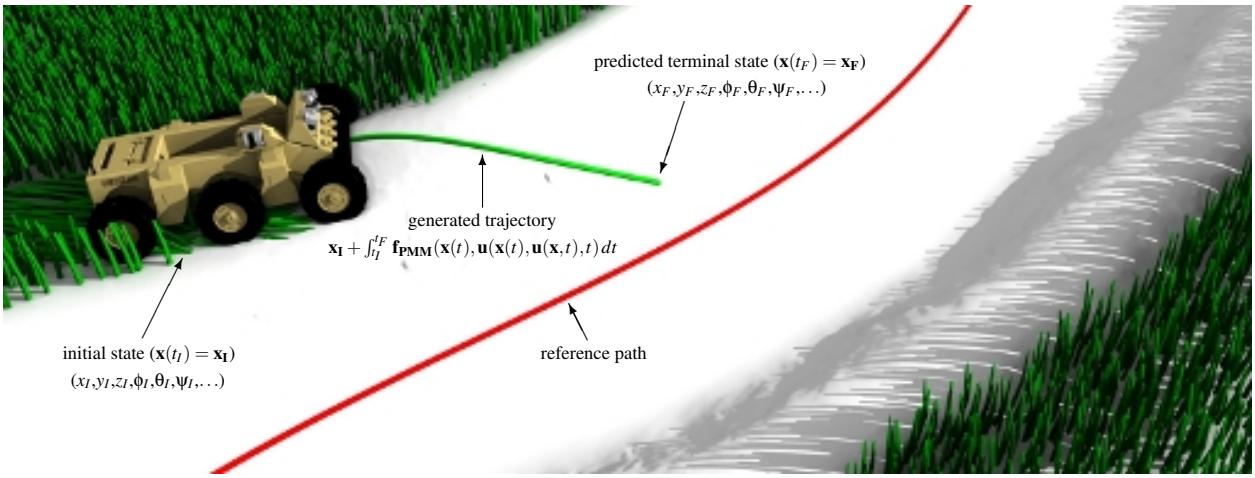


Figure 4.14: The unconstrained optimization model-predictive trajectory generation problem is defined as the search for actions that minimize a cost function ($\mathbf{J}(\mathbf{x}(t), \mathbf{u}(\mathbf{x}, t), t_I, t_F)$) as a vehicle moves the environment from an initial state ($\mathbf{x_I}$) for bounded time. In this formulation the mapping from actions to state adheres to a predictive motion model ($\mathbf{f_{PMM}}(\mathbf{x}(t), \mathbf{u}(\mathbf{x}, t), t)$). Here, a vehicle minimizes a utility function based on integrated path deviation in order to reacquire the path.

the technique is divided between the optimization routine and the cost Jacobian estimation methods. In UNCONSTRAINEDOPTIMIZATIONMPTG, the algorithm takes in the initial state, the predictive motion model, the action parameterization, and the utility function to be optimized. An initial guess of action parameters are determined by an initialization function (as described in Section 4.1.3 and subsequently optimized until the elements in the m column cost Jacobian approaches zero. The ESTIMATECOSTJACOBIAN function estimates the partial derivatives of the cost Jacobian numerically. As in ESTIMATECONSTRAINTJACOBIAN, $m+1$ predictive motion model integrations are necessary to estimate the partial derivative of the cost function with respect to the action parameters.

4.2.1 EXPERIMENTS

MOBILE ROBOT NAVIGATION IN A CLUTTERED OBSTACLE FIELD

One of the best application of unconstrained optimization model-predictive trajectory generation is mobile robot navigation and control. Generally, mobile robot navigation is the problem of determining an action from the continuum that keeps the vehicle safe and moving towards some goal. There are typically no terminal state constraints imposed in this problem, even in path following, where it may actually be more efficient or safe to deviate from the reference trajectory. A simplified version of this problem is shown in Figure 4.16, where a vehicle must determine the minimum-cost action to safely navigate through the environment.

UNCONSTRAINEDOPTIMIZATIONMPTG($\mathbf{x_I}$,$\mathbf{u}(\mathbf{p},\mathbf{x},t)$,$\mathbf{f_{PMM}}(\mathbf{x}(t),\mathbf{u}(\mathbf{p},\mathbf{x},t),t)$)

INPUT: $\mathbf{x_I}$ (initial state), $\mathbf{u}(\mathbf{p},\mathbf{x},t)$ (parameterized action), $\mathbf{f_{PMM}}(\mathbf{x}(t),\mathbf{u}(\mathbf{p},\mathbf{x},t),t)$ (predictive motion model)
OUTPUT: $\mathbf{p}$ (action parameters)

$\mathbf{p} \leftarrow$ INITIALIZE($\mathbf{x_I}$,$\mathbf{x_T}$) ;
while $\frac{\delta(\Delta \mathbf{x_T}(\mathbf{p}))}{\delta \mathbf{p}} \neq 0$ **do**
  $\frac{\delta(\Delta \mathbf{x_T}(\mathbf{p}))}{\delta \mathbf{p}} \leftarrow$ ESTIMATECOSTJACOBIAN($\mathbf{x_I}$,$\mathbf{u}(\mathbf{p},\mathbf{x},t)$,$\mathbf{f_{PMM}}(\mathbf{x}(t),\mathbf{u}(\mathbf{p},\mathbf{x},t),t)$,$\mathbf{p}$) ;
  $\Delta \mathbf{p} \leftarrow -\alpha \frac{\delta(\Delta \mathbf{x_T}(\mathbf{p}))}{\delta \mathbf{p}}$;
  $\mathbf{p} \leftarrow \mathbf{p} + \Delta \mathbf{p}$;
end
return $\mathbf{p}$;

ESTIMATECOSTJACOBIAN($\mathbf{x_I}$,$\mathbf{u}(\mathbf{p},\mathbf{x},t)$,$\mathbf{f_{PMM}}(\mathbf{x}(t),\mathbf{u}(\mathbf{p},\mathbf{x},t),t)$,$\mathbf{p}$)

INPUT: $\mathbf{x_I}$ (initial state), $\mathbf{u}(\mathbf{p},\mathbf{x},t)$ (parameterized action), $\mathbf{f_{PMM}}(\mathbf{x}(t),\mathbf{u}(\mathbf{p},\mathbf{x},t),t)$ (predictive motion model), $\mathbf{p}$ (action parameters)
OUTPUT: $\frac{\delta(J(\mathbf{p}))}{\delta \mathbf{p}}$ (cost jacobian estimate)

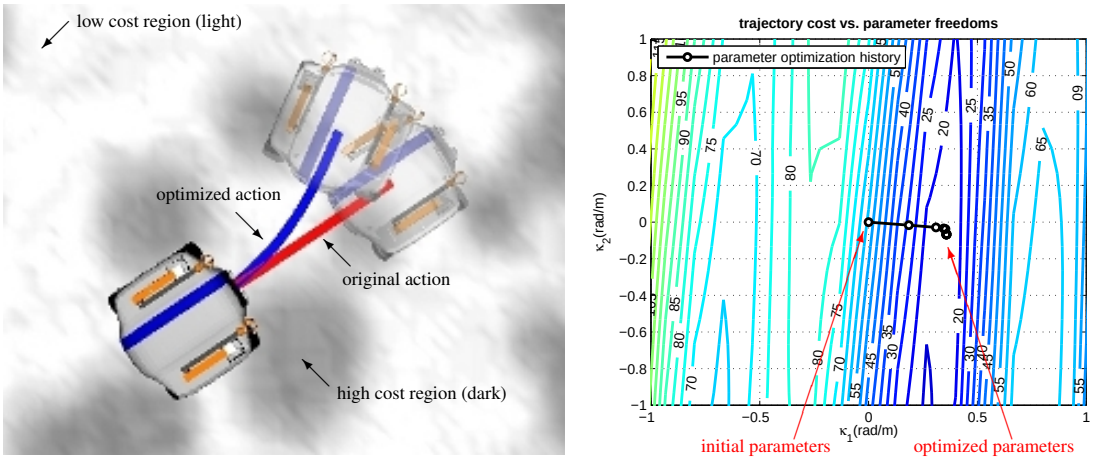
$m \leftarrow rows(\mathbf{p})$;
for $j = 1 : m$ **do**
  $\frac{\delta(J(\mathbf{p}))}{\delta \mathbf{p}[j]} \leftarrow \frac{J(\mathbf{p}[j]+\mathbf{h_j},\mathbf{p}) - \mathbf{J}(\mathbf{p})}{h}$;
end
return $\frac{\delta(J(\mathbf{p}))}{\delta \mathbf{p}}$;

Figure 4.15: Unconstrained Optimization Model-Predictive Trajectory Generation algorithm.

As previously mentioned, one of the most important applications of unconstrained optimization model-predictive trajectory generation in mobile robot autonomy involves navigation and control. There are typically no terminal state constraints imposed in this problem, even in path following, where it may actually be beneficial to deviate from the reference trajectory. A simplified example of this problem is shown in Figure 4.16, where a vehicle must determine the minimum-cost action to safely navigate through the environment.

Like the constrained model-predictive trajectory generation example, the number of degrees of freedom in the parameterized action was limited to two in order to easily visualize the optimization process. In this example, the controllable freedoms in the parameterized action are the middle (κ_1) and final (κ_2) spline point in a second-order curvature spline as a function of distance. The initial spline point (κ_0) was fixed to the curvature in the initial state to generate a smooth action with continuous curvature rates:

$$\kappa(\mathbf{p},\mathbf{x},t) = a(\mathbf{p}) + b(\mathbf{p})s(t) + c(\mathbf{p})s(t)^2 \quad \text{for} \quad s_I \leq s(t) \leq s_F \quad \text{and} \quad \mathbf{p} = \begin{bmatrix} \kappa_0 & \kappa_1 & \kappa_2 \end{bmatrix}^T \tag{4.36}$$



(a) the original and optimized parameterized action in the environment

(b) cost vs. parameter freedom (κ_1,κ_2)

Figure 4.16: Unconstrained optimization model-predictive trajectory generation for mobile robot navigation. In this example, a vehicle relaxes a parameterized action subject to the predictive motion model constraints to determine the safest route through the environment.

$$a(\mathbf{p}) = \kappa_0 \tag{4.37}$$

$$b(\mathbf{p}) = -\frac{3\kappa_0 - 4\kappa_1 + \kappa_2}{s_F - s_I} \tag{4.38}$$

$$c(\mathbf{p}) = 2\frac{\kappa_0 - 2\kappa_1 + \kappa_2}{(s_F - s_I)^2} \tag{4.39}$$

$$\tag{4.40}$$

The action length in this experiment is fixed. This is common in mobile robot navigation and is done to ensure that the vehicle could safely maneuver over a fixed period of time. The problem becomes determining the best way to steer through the environment over a fixed duration of time.

From an initially straight action (κ_0=κ_1=κ_2=$0\frac{rad}{m}$), the parameterized freedoms follow the numerically estimated cost function gradients to reduce the integrated trajectory cost. The original action deforms from one that passes over a high-cost region to a trajectory that navigates between the two high-cost regions. Just as with constrained model-predictive trajectory generation, the partial derivatives in the cost function Jacobian are determined numerically, which enables any utility function representation or predictive motion model to be used. This technique is extended for mobile robot navigation and control in Chapter 6, where actions parameterized by sequential search solutions are relaxed for trajectory following and obstacle avoidance using the unconstrained optimization trajectory generation technique.

4.3 CONSTRAINED OPTIMIZATION MODEL-PREDICTIVE TRAJECTORY GENERATION

The final class of model-predictive trajectory generation techniques are ones that require boundary state constraints satisfaction and cost function minimization. This approach matters in applications such as instrument placement in cluttered environments, where the shape of the generated trajectory matters when driving to a specific location (Figure 4.17).

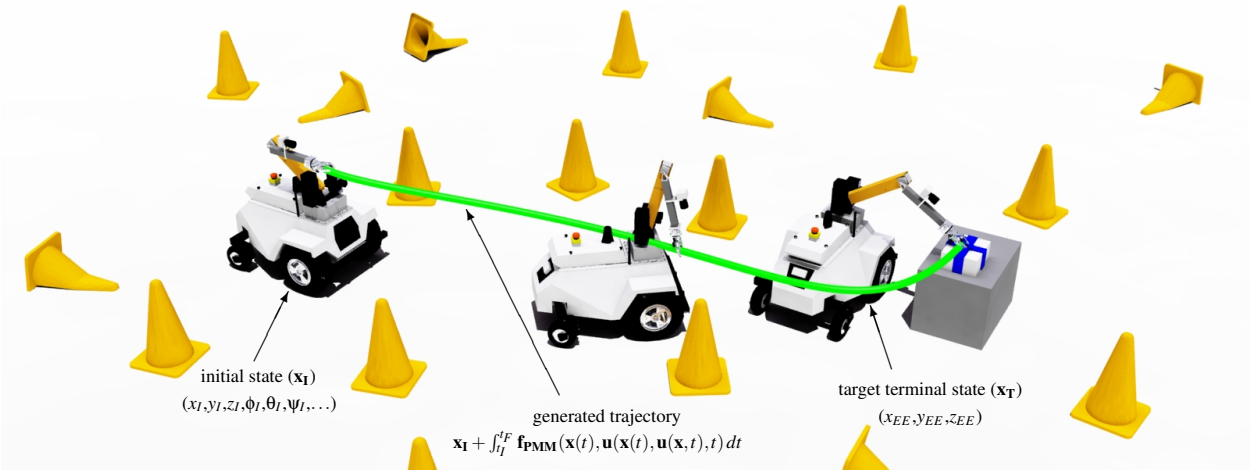


Figure 4.17: Overview of constrained optimization model-predictive trajectory generation. In this example, a mobile robot must place an end effector at a particular position (x_{EE}, y_{EE}, z_{EE}) while avoiding a sequence of obstacles.

The constrained optimization model-predictive trajectory generation problem is formulated as a utility function minimization and constraint satisfaction achieved by modifying the actions of the vehicle:

$$\begin{aligned} \text{minimize:} \quad & \mathbf{J}(\mathbf{x}(t), t_I, t_F) \\ \text{subject to:} \quad & \dot{\mathbf{x}} = \mathbf{f_{PMM}}(\mathbf{x}(t), \mathbf{u}(\mathbf{x}, t), t) \\ & \mathbf{x}(t_I) = \mathbf{x_I} \\ & \mathbf{x}(t_F) = \mathbf{x_I} + \int_{t_I}^{t_F} \mathbf{f_{PMM}}(\mathbf{x}(t), \mathbf{u}(\mathbf{p}, \mathbf{x}, t), t)\, dt \end{aligned} \tag{4.41}$$

For a third time, parameterized actions are used to reduce the scope of the action search and transform the problem into parametric optimal control. An effective method for solving problems of this form involve Lagrange multipliers. A Hamiltonian (H) is defined as the sum of the utility and the product of the Lagrange multiplier vector (λ) of length m (the length of the state constraint vector) with the constraint error:

$$H(\mathbf{x}(t), \mathbf{u}(\mathbf{p}, \mathbf{x}, t), t_I, t_F, \lambda) = \mathbf{J}(\mathbf{x}(t), \mathbf{u}(\mathbf{p}, \mathbf{x}, t), t_I, t_F) + \lambda^T \Delta \mathbf{x_T}(\mathbf{x}(t), \mathbf{u}(\mathbf{p}, \mathbf{x}, t), t_I, t_F) \tag{4.42}$$

The first order necessary conditions for optimality are well known. The partial derivatives of the Hamiltonian with respect to the action parameters and the Lagrange multipliers approach zero near the (locally) optimal solution:

$$\frac{\delta H(\mathbf{x}(t),\mathbf{u}(\mathbf{p},\mathbf{x},t),t_I,t_F,\lambda)}{\delta \mathbf{p}} = \frac{\delta \mathbf{J}(\mathbf{x}(t),\mathbf{u}(\mathbf{p},\mathbf{x},t),t_I,t_F)}{\delta \mathbf{p}} + \lambda \frac{\delta \Delta \mathbf{x_T}(\mathbf{x}(t),\mathbf{u}(\mathbf{p},\mathbf{x},t),t_I,t_F)}{\delta \mathbf{p}} = \mathbf{0} \tag{4.43}$$

$$\frac{\delta H(\mathbf{x}(t),\mathbf{u}(\mathbf{p},\mathbf{x},t),t_I,t_F,\lambda)}{\delta \lambda} = \Delta \mathbf{x_T}(\mathbf{x}(t),\mathbf{u}(\mathbf{p},\mathbf{x},t),t_I,t_F) = \mathbf{0} \tag{4.44}$$

There are a total of $n+m$ terms in this system of equations, where n is the length of the free action parameter vector. The system of equations can be solved by linearizing the first-order necessary conditions for optimality:

$$\begin{bmatrix} \frac{\delta^2 H(\mathbf{x}(t),\mathbf{u}(\mathbf{p},\mathbf{x},t),t_I,t_F,\lambda)}{\delta \mathbf{p}^2} & \left[\frac{\delta \Delta \mathbf{x_T}(\mathbf{x}(t),\mathbf{u}(\mathbf{p},\mathbf{x},t),t_I,t_F)}{\delta \mathbf{p}}\right]^T \\ \frac{\delta \Delta \mathbf{x_T}(\mathbf{x}(t),\mathbf{u}(\mathbf{p},\mathbf{x},t),t_I,t_F)}{\delta \mathbf{p}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \Delta \mathbf{p} \\ \Delta \lambda \end{bmatrix} = \begin{bmatrix} \frac{\delta H(\mathbf{x}(t),\mathbf{u}(\mathbf{p},\mathbf{x},t),t_I,t_F,\lambda)}{\delta \mathbf{p}} \\ \Delta \mathbf{x_T}(\mathbf{x}(t),\mathbf{u}(\mathbf{p},\mathbf{x},t),t_I,t_F) \end{bmatrix} \tag{4.45}$$

The initial guess for the action parameters and the Lagrange multipliers are refined with each iteration until the state constraints are satisfied and partial derivative of the Hamiltonian with respect to the action parameters approaches zero. The per-cycle computational complexity of this approach is significantly more that that of either the constrained model-predictive trajectory generation or unconstrained optimization model-predictive trajectory generation algorithms as the Hessian of the Hamiltonian requires a minimum of n^2+1 action simulations of parameterized actions to estimate all of the second partial derivatives. A very efficient implementation of the algorithm stores the various action simulations used to estimate the partial derivatives in the Hessian of the Hamiltonian for the Hamiltonian and constraint Jacobians.

An outline of an implementation of the constrained optimization model-predictive trajectory generation algorithm is shown in Figure 4.18. Following the other algorithms presented in this chapter, the constrained optimization model-predictive trajectory generation is an iterative technique that modifies a set of parameter values in addition to a set of Lagrange multiplier values based on partial derivatives of constraint error and cost. The individual processes for estimating the Jacobian and Hessian of the constraint error and Hamiltonian follow those presented in the previous techniques.

4.3.1 EXPERIMENTS

MOBILE ROBOT INSTRUMENT PLACEMENT IN A CLUTTERED OBSTACLE FIELD

An important application of constrained optimization model-predictive trajectory generation involves mobility and instrument placement in cluttered obstacle fields. Figure 4.19 shows an example similar to a previous experiment described in Section 4.2.1 where a mobile robot must generate a minimum-cost action in an environment represented by a costmap. The difference in this experiment is that the mobile robot must reach a particular terminal state to perform some task. The trajectory determined by the constrained optimization model-predictive trajectory generation algorithm is shown in Figure 4.19a, which shows the vehicle achieving the target terminal state while navigating around the high-cost region in the environment. To compare the result with an alternative method, the trajectory

CONSTRAINEDOPTIMZATIONMPTG($\mathbf{x_I}$,$\mathbf{x_T}$,$\mathbf{u}(\mathbf{p},\mathbf{x},t)$,$\mathbf{f_{PMM}}(\mathbf{x}(t),\mathbf{u}(\mathbf{p},\mathbf{x},t),t)$)

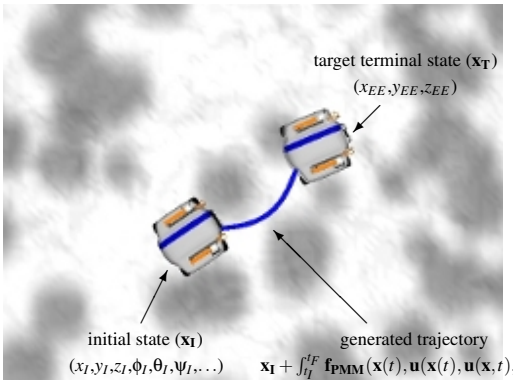
INPUT: $\mathbf{x_I}$ (initial state), $\mathbf{x_T}$ (target terminal state), $\mathbf{u}(\mathbf{p},\mathbf{x},t)$ (parameterized action), $\mathbf{f_{PMM}}(\mathbf{x}(t),\mathbf{u}(\mathbf{p},\mathbf{x},t),t)$ (predictive motion model)
OUTPUT: $\mathbf{p}$ (action parameters)

$\mathbf{p} \leftarrow$ INITIALIZE($\mathbf{x_I}$,$\mathbf{x_T}$) ;
while $\mathbf{x_T} \neq \mathbf{x}(t_I) + \int_{t_I}^{t_F} \mathbf{f_{PMM}}(\mathbf{x}(t),\mathbf{u}(\mathbf{p},\mathbf{x},t),t)\,dt$ & $\frac{\delta(H(\mathbf{x}(t),\mathbf{u}(\mathbf{p},\mathbf{x},t),t_I,t_F,\lambda))}{\delta \mathbf{p}} \neq 0$ **do**
  $\Delta\mathbf{x_T} \leftarrow \mathbf{x_T} - \left[\mathbf{x}(t_I) + \int_{t_I}^{t_F} \mathbf{f_{PMM}}(\mathbf{x}(t),\mathbf{u}(\mathbf{p},\mathbf{x},t),t)\,dt\right]$;
  $\frac{\delta^2(H(\mathbf{x}(t),\mathbf{u}(\mathbf{p},\mathbf{x},t),t_I,t_F,\lambda))}{\delta \mathbf{p}^2} \leftarrow$ ESTIMATEHAMILONITANHESSIAN($\mathbf{x_I}$,$\mathbf{x_T}$,$\mathbf{u}(\mathbf{p},\mathbf{x},t)$,$\mathbf{f_{PMM}}(\mathbf{x}(t),\mathbf{u}(\mathbf{p},\mathbf{x},t),t)$,$\mathbf{p}$) ;
  $\frac{\delta(H(\mathbf{x}(t),\mathbf{u}(\mathbf{p},\mathbf{x},t),t_I,t_F,\lambda))}{\delta \mathbf{p}} \leftarrow$ ESTIMATEHAMILTOIANJACOBIAN($\mathbf{x_I}$,$\mathbf{x_T}$,$\mathbf{u}(\mathbf{p},\mathbf{x},t)$,$\mathbf{f_{PMM}}(\mathbf{x}(t),\mathbf{u}(\mathbf{p},\mathbf{x},t),t)$,$\mathbf{p}$) ;
  $\frac{\delta(\Delta\mathbf{x_T}(\mathbf{x}(t),\mathbf{u}(\mathbf{p},\mathbf{x},t),t_I,t_F))}{\delta \mathbf{p}} \leftarrow$ ESTIMATECONSTRAINTJACOBIAN($\mathbf{x_I}$,$\mathbf{x_T}$,$\mathbf{u}(\mathbf{p},\mathbf{x},t)$,$\mathbf{f_{PMM}}(\mathbf{x}(t),\mathbf{u}(\mathbf{p},\mathbf{x},t),t)$,$\mathbf{p}$) ;
  $$\begin{bmatrix} \Delta\mathbf{p} \\ \Delta\lambda \end{bmatrix} \leftarrow - \begin{bmatrix} \frac{\delta^2 H(\mathbf{x}(t),\mathbf{u}(\mathbf{p},\mathbf{x},t),t_I,t_F,\lambda)}{\delta \mathbf{p}^2} & \left[\frac{\delta\Delta\mathbf{x_T}(\mathbf{x}(t),\mathbf{u}(\mathbf{p},\mathbf{x},t),t_I,t_F)}{\delta \mathbf{p}}\right]^T \\ \frac{\delta\Delta\mathbf{x_T}(\mathbf{x}(t),\mathbf{u}(\mathbf{p},\mathbf{x},t),t_I,t_F)}{\delta \mathbf{p}} & \mathbf{0} \end{bmatrix}^{-1} \begin{bmatrix} \frac{\delta H(\mathbf{x}(t),\mathbf{u}(\mathbf{p},\mathbf{x},t),t_I,t_F,\lambda)}{\delta \mathbf{p}} \\ \Delta\mathbf{x_T}(\mathbf{x}(t),\mathbf{u}(\mathbf{p},\mathbf{x},t),t_I,t_F) \end{bmatrix}$$;
  $\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}$;
  $\lambda \leftarrow \lambda + \Delta\lambda$;
end
return $\mathbf{p}$;

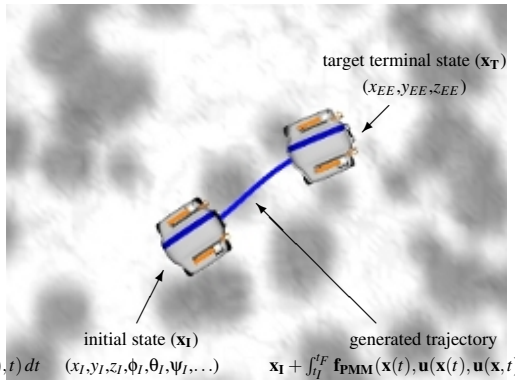
Figure 4.18: Constrained Optimization Model-Predictive Trajectory Generation algorithm.

generated by the alternative constrained optimization model-predictive trajectory generation technique in Section 4.1 is shown in Figure 4.19b, which drives straight through the high-cost region. The resulting cost of the trajectory generated with the constrained optimization model-predictive trajectory generation (26.310) was 77.4% less that the cost of the trajectory determined by the constrained model-predictive trajectory generator (116.655), despite the same action parameterization and initialization.

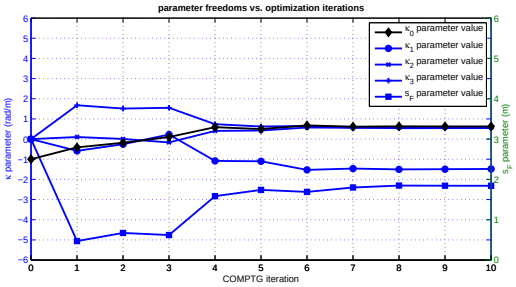
The primitive function for the curvature controller used a third-order spline as described in Equations 4.15 through 4.19. All knot points in the spline (κ_0, κ_1, κ_2, κ_3) were allowed to relax in addition to the length of the trajectory (s_F). The variation of the five action parameter values over the optimization is shown in Figure 4.19c. The parameter value history shows that an initial large parameter modification followed by a slower descent into the locally optimal parameter values. First, the cost dramatically decreases from the initial value (Figure 4.19d) while increasing the terminal boundary state constraint error (Figure 4.19e). Once the gradient of the cost function decreases, the contribution of the boundary state constraint satisfaction is more significant but continues to minimize the trajectory cost. The constrained model-predictive trajectory generation algorithm produces a trajectory with significantly higher cost because there is no aspect of the algorithm that attempts to minimize a penalty function. The relative efficiency of the constrained optimization model-predictive trajectory generation algorithm comes from reducing the problem scope from general optimal control to parameter search.



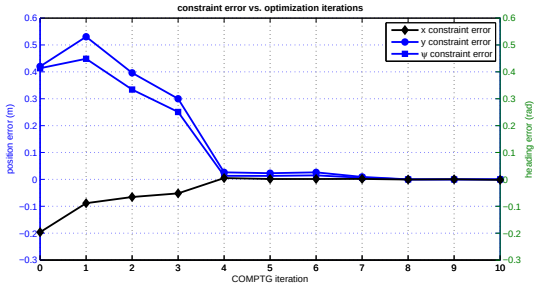
(a) trajectory generated using constrained optimization model-predictive trajectory generation, with a cost of 26.310



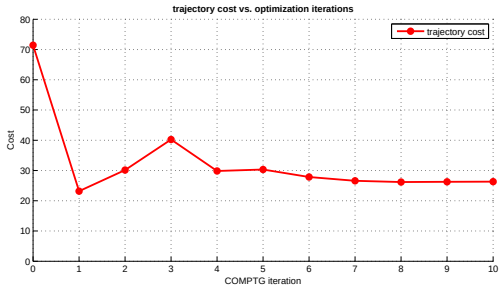
(b) trajectory generated using constrained model-predictive trajectory generation, with a cost of 116.655



(c) parameter freedom history during constrained optimization model-predictive trajectory generation



(d) constraint error history during constrained optimization model-predictive trajectory generation



(e) trajectory cost history during constrained optimization model-predictive trajectory generation

Figure 4.19: Constrained optimization model-predictive trajectory generation in an obstacle field. A solution is determined using the constraint optimization model-predictive trajectory generation technique in 4.19a that satisfies the boundary state constraints while minimizing a cost function along the path. The solution is different than the one generated using the constrained model-predictive trajectory generation technique in Figure 4.19b because the constrained optimization variation has a capacity to reduce the path cost.

4.4 IMPLEMENTATION

The model-predictive trajectory generation techniques presented in this chapter generally decompose into two parts: model-independent and model-dependent methods. Model-independent components provide the framework for the constraint satisfaction and/or utility minimization by modifying a vector of action parameters. They provide functionality including predictive motion model integration, parameterized control handling, and solutions methods for the three model-predictive trajectory generation techniques described in this Chapter. Model-dependent components describe the configuration, control, initialization, and predictive mobility model for a particular platform of interest.

An effective implementation of a model-predictive trajectory generator that has been used throughout the simulation and field experiments described in this chapter is built around general interfaces between vehicle-dependent and vehicle-independent parts. Figure 4.20 shows an effective model-predictive trajectory generation architecture that encompasses the constrained, unconstrained optimization, and constrained optimization model-predictive trajectory generation methods that is applied to a variety of different platforms by implementing vehicle-model specific components.

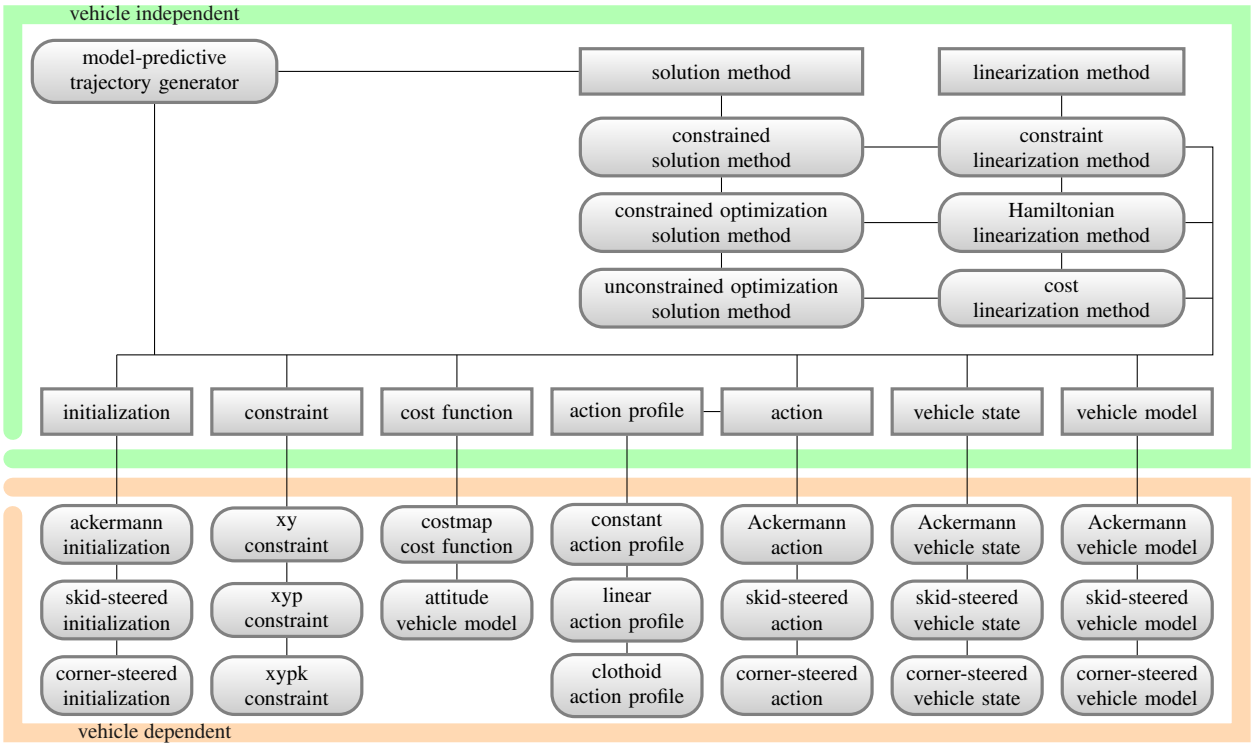


Figure 4.20: The architecture of an implementation of the model-predictive trajectory generation algorithm. This implementation is divided into vehicle-dependent and vehicle-independent components to simplify the adaptation onto multiple vehicle platforms.

4.5 SUMMARY

The presented model-predictive trajectory generation techniques have proven to be an effective method for generating parameterized actions that satisfy continuum boundary state constraints and minimize utility functionals while considering general environmental interaction and vehicle mobility. The methods have been shown to determine actions that predictively compensate for terrain shape and impaired mobility, optimize chassis configuration, generate efficient and expressive search spaces by sampling in the state space, and generate actions to navigate through cluttered obstacle fields. Particular implementations of the technique are highly application dependent, as the choice of action parameterization is a function of the mobility system, the environment, and the types of state constraints imposed. The constrained model-predictive trajectory generation technique in Section 4.1 is particularly suited to generate primitive actions in control sets that define motion planning graphs as an environment is generally not explicitly defined. The clearest application of unconstrained optimization model-predictive trajectory generation from Section 4.2 is in mobile robot navigation, where a minimum-cost trajectory is desired and only the initial state and planning horizon are typically defined. The last, most powerful, and most computationally expensive variation of the model-predictive trajectory generation algorithms described in Section 4.3 is constrained optimization model-predictive trajectory generation, which simultaneously satisfies boundary state constraints and minimizes a cost functional. While useful for small-scale instrument placement tasks, sequential search techniques are typically better suited to the task of long range motion planning as they are not susceptible to local minima.

An architecture for separating the vehicle dependent and vehicle independent portions of the algorithm has been presented and effectively demonstrated on a number of diverse mobile robot systems. Variants of the model-predictive trajectory generation algorithm have been used to predictively compensate for impaired mobility and terrain shape for planetary rovers, generate state-space sampling local motion planning search spaces for off-road and on-road mobile robots, and generate coordinated motion and manipulation actions.

CHAPTER 5

ADAPTIVE MODEL PREDICTIVE SEARCH SPACES

Whereas trajectory generators are generally concerned with determining locally optimal continuum actions, motion planners seek obstacle-free or globally minimum-cost paths through an environment. The primary function of a motion planner is to search the path continuum for an acceptable solution. For mobile robots with non-trivial mobility systems, approximations of the continuum are necessary as contemporary algorithms cannot search the entire action space efficiently. This approximation transforms the continuous representation to a discrete one which sacrifices optimality for runtime performance and reduced computational complexity.

Selection of the mapping between the path continuum and discretized graphs is one of the most challenging problems in mobile robot motion planning. Custom representations must be made for particular platforms, applications, and environments. Dense sampling of the continuum can be necessary to generate motion plans in cluttered obstacle fields. Control sets must be designed to efficiently search the feasible action space for a particular platform. Even with similar vehicles, mobility models degrade over time and heterogeneous terrain can render a particular representation inefficient or ineffective.

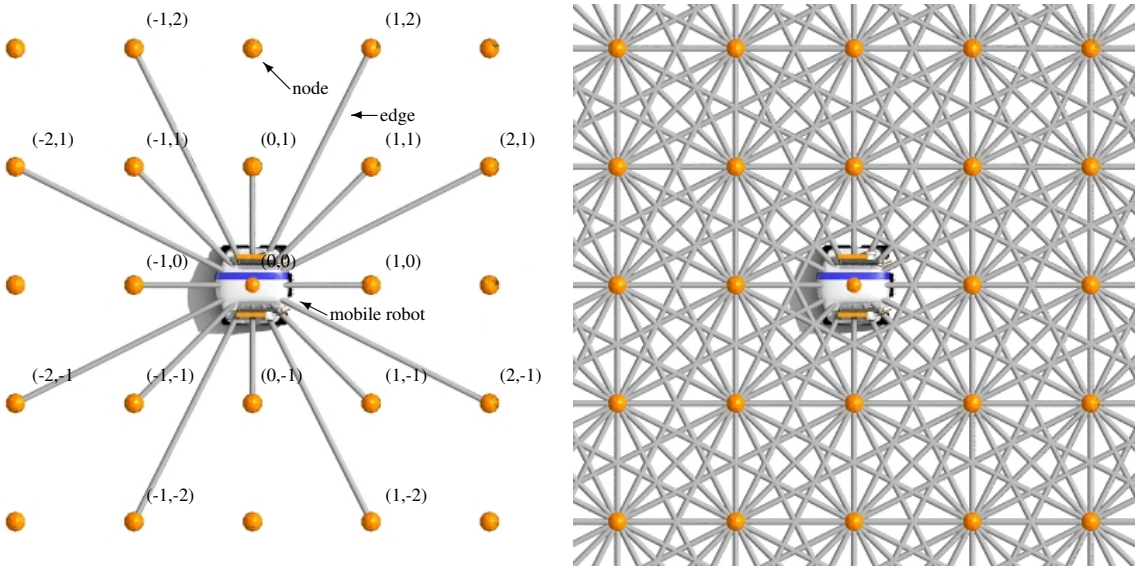
Related problems are the efficiency, density, and expressiveness of the search space. Dense graphs can be costly to search with a non-perfect heuristic and are wasteful in open, homogeneous environments. Conversely, coarse graphs can find a solution quickly, but it may be highly suboptimal if the approximation of all motions is not expressive enough. In a cluttered obstacle field, if a particular node or edge does not exist in the search space, the resulting motion plan may not be able to pass through narrow corridors. A potentially better search space is one that can adapt its structure to the environment while preserving the underlying topological map.

This section proposes several techniques for adaptive model-predictive search spaces for mobile robot motion planning in complex environments. Built upon the concept of state lattices (Pivtoraiko et al., 2009), informed state lattices are recombinant motion planning graphs whose edges are informed using the model-predictive trajectory generator presented in Chapter 4. Adaptive state lattices extend this by deforming the search space to the environment during graph construction. This technique exploits the gradient of a cost function representing all edges from a node to locally optimize the state mapping equation that transforms discretized nodes into continuum states. Feasibility is maintained by modifying the edges with the model-predictive trajectory generation techniques presented in Chapter 4. Local relaxation of the state mapping equations enable adaptive state lattices to produce much higher quality motion plans with coarse representations, which can be more efficient to search than dense, fixed representations in complex environments. The last technique presented, the progressively adaptive state lattice, optimizes the motion planning graph iteratively, enabling a faster replanning rate.

5.1 STATE LATTICES

In the context of motion planning, a state lattice is a regular sampling of state space where discrete nodes correspond to continuous positions, orientations, velocities, curvatures, and/or configurations of a vehicle. State lattices are typically defined by two objects: a state mapping equation and a control set. The state mapping equation governs the correspondence between continuous states in the real world and discrete nodes in the motion planning graph. It is used to transform the initial state and the target terminal state to the respective start and goal nodes in the discretized graph. Repeated expansions of the control set in a regular sampling of state space often result in a recombinant search space that can be very efficiently searched.

A control set represents the enumerated set of transitions (in the form of edges) from a particular node to other nodes in the graph. Repeated expansions of the control set from reachable nodes generate the discretized motion planning search space. Figure 5.1 shows a sixteen connected control set that enables the mobile robot to transition to any real-valued states represented by a discretized node in the two-dimensional graph. While searching the two-dimensional graph is efficient, the resulting path is continuous in position only, and is discontinuous in heading and path curvature. Resulting motion plans from this search space are infeasible for many mobile robots systems, specifically those with differential constraints that preclude turn-in-place actions.



(a) sixteen-connected control set emanating from the node (0,0,0)

(b) search space generated by sixteen-connected motion template

Figure 5.1: A motion planning search space produced by repeating a sixteen-connected control set at each node in the graph. In this example, the resulting path is continuous in two-dimensional position but discontinuous in path heading and curvature.

5.1.1 FEASIBLE CONTROL SET GENERATION

Feasible control set generation is the challenge of determining the subset of all possible actions that most effectively represents the continuum of actions that satisfy the predictive mobility model for a vehicle. There are numerous significant challenges in this problem. First, it is a fundamental trade-off between discretization, outdegree, feasibility, and efficiency. A finely discretized, highly expressive control set may provide an effective approximation of the continuum, but may consume too many computational resources (processor cycles and memory) to run efficiently. A coarse discretization with few, sparsely connected edges can quickly produce a suboptimal, inadequate path that fails to effectively move the robot through the environment. It is an application dependent problem that demands understanding of vehicle mobility, as the state-space discretization for a multi-ton field robot may be too coarse for a small mobile manipulator.

The challenges are even more significant for mobile robots in heterogeneous worlds. A coarse representation of the path continuum may be appropriate in some local environments whereas some locales will demand varying degrees of expressiveness. The ideal control set is one that operates efficiently for all possible obstacle densities. Control set design is often a complex, iterative process necessary for well performing motion planning search spaces in complex environments.

One effective method is to use input-space sampling techniques (Figure 5.2) to guide the search to produce feasible control sets as in Ferguson et al. (2008). In this approach, initial states are determined by inverting the state mapping equation ($\mathbf{f_{SME}}$) for all discretized nodes ($\mathbf{S_I}$). Candidate terminal states ($\mathbf{x_F}$) are determined by evaluating a sampled space of actions ($\mathbf{U_I}$) through the predictive motion model for each initial state. Each terminal state is transformed back into the discretized representation through the state mapping equation and added to a set of possible terminal nodes for the particular initial node ($\mathbf{S_T}(\mathbf{s_I})$). For each boundary state pair represented in $\mathbf{S_T}(\mathbf{s_I})$, the corresponding terminal state for the discretized node is determined and an action satisfying the boundary state constraints is generated using an inverse trajectory generator, such as the one presented in Chapter 4. If a feasible action can be found, it is added to the control set as a function of the initial discretized node. While the discretization of the resulting control set cannot be modified, the motion template outdegree can be effectively controlled by throttling the density of the input-space sampling.

Using this algorithm, control sets which adhere to the vehicle's predictive motion model can be determined (Figure 5.3). The parameterized representation of actions enables efficient storage of the inputs necessary to execute the motion. Only the parameters used to define the functions are necessary to reconstruct the action and, with the initial state and the predictive motion model, the subsequent trajectory. In this example, a representation of a control set for a three-dimensional motion planning graph representing variable position and heading is shown. The resulting search space is now continuous in position and heading and respects the predictive motion model.

A regular sampling of state space, a static representation of vehicle mobility, the assumption of flat terrain, and uniform terramechanical properties enables efficient evaluation and storage of control sets during search. These restrictions allow for two significant performance improvements, including precomputing edge swaths and a free-space heuristic (Pivtoraiko et al., 2009). Precomputing the edge swaths by determining each cell in a costmap that the vehicle passes through saves numerous expensive vehicle simulations during control set expansion. Similarly, a lookup table

FEASIBLECONTROLSETGENERATION($\mathbf{S_I}$,$\mathbf{U_s}$,$\mathbf{f_{PMM}}(\mathbf{x}(t),\mathbf{u}(\mathbf{x},t),t)$)

INPUT: $\mathbf{S_I}$ (sampled input nodes), $\mathbf{U_s}$ (sampled actions), $\mathbf{f_{PMM}}(\mathbf{x}(t),\mathbf{u}(\mathbf{x},t),t)$ (predictive motion model)
OUTPUT: $\mathbf{S_T}(\mathbf{s_I})$ (child node set), $\mathbf{U}(\mathbf{s_I})$ (control set)

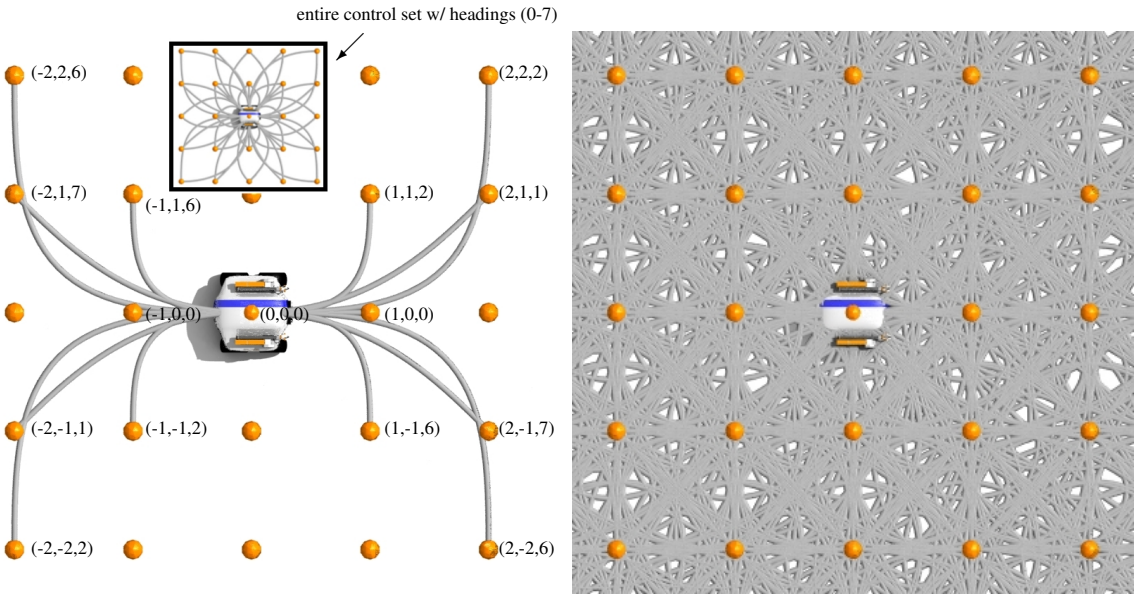
1 **foreach** $\mathbf{s_I} \in \mathbf{S_I}$ **do**
2 $\mathbf{x_I} \leftarrow \mathbf{f_{SME}}(\mathbf{s_I})$
3 **foreach** $\mathbf{u_s}(\mathbf{x},\mathbf{t}) \in \mathbf{U_s}$ **do**
4 $\mathbf{x_F} \leftarrow \mathbf{x_I} + \int_{t_I}^{t_F} \mathbf{f_{PMM}}(\mathbf{x}(t),\mathbf{u_s}(\mathbf{x},t),t)\,dt$
5 $\mathbf{s_T} \leftarrow \mathbf{f_{SME}}^{-1}(\mathbf{x_F})$
6 **if** $\mathbf{s_T} \ni \mathbf{S_T}(\mathbf{s_I})$ **then**
7 $\mathbf{S_T}(\mathbf{s_I}) \leftarrow \mathbf{s_T}$
  end
  end
8 **foreach** $\mathbf{s_T} \in \mathbf{S_T}$ **do**
9 $\mathbf{x_T} \leftarrow \mathbf{f_{SME}}(\mathbf{s_T})$
10 CONSTRAINEDMPTG($\mathbf{x_I},\mathbf{x_T},\mathbf{u}(\mathbf{p},\mathbf{x},t)$)
11 $\mathbf{U}(\mathbf{s_I}) \leftarrow \mathbf{u}(\mathbf{p},\mathbf{x},t)$
  end
 end
12 **return** $\mathbf{S_T}(\mathbf{s_I}),\mathbf{U}(\mathbf{s_I})$

Figure 5.2: Feasible control set generation for a state lattice using the constrained model-predictive trajectory generation algorithm.

that returns the minimum length path in open environments for any query in the state lattice (Knepper and Kelly, 2006) is a perfect heuristic that greatly improves graph search performance.

This static description of control sets suffers from two disadvantages. First, the vehicle cannot deviate from the fixed control set uniformly repeated throughout the environment. Even if a slight deviation from the control set trajectory would greatly decrease the cost of the resulting path, no technique exists to efficiently and effectively modify the static control set. Second, edges in the motion planning graph are still constructed on the assumption of fixed vehicle mobility and flat terrain. If the vehicle attitude or configuration is an important metric used to determine the minimum-cost motion plan, each edge may need to be simulated and regenerated efficiently in order to inform the search space about the true mobility costs.

The remaining portions of this chapter address these two shortcomings in recombinant state lattice search spaces. Section 5.2 applies the techniques presented in Chapter 4 to efficiently regenerate the actions that represent the state transition equations in the control set. The efficiency of the technique comes from exploration of the action continuum to search in a parameterized sub-space of motion. Sections 5.3 and 5.4 present two variations of the same technique for local optimization of the state mapping equation to generate motion planning search spaces that deviate from the regular sampling in path space to conform to the environment.



(a) a feasible lattice control set emanating from nodes with position (0,0) and headings (0)

(b) search space generated by feasible lattice control set

Figure 5.3: A motion planning search space produced by repeating a feasible control set at each node in the graph. In this example, the resulting path is continuous in both two-dimensional position and path heading.

5.2 Informed State Lattices

The state lattices described in Section 5.1 work only on the assumption that the mobility and environment is the same as the one used to generate the control set. The previous experiments in Section 4.1 showed that terrain influences the shape and duration of the actions necessary to satisfy the boundary state constraints. Differences between the environment used to generate the control set and the observed terrain affects the connectivity of the resulting state lattice. Likewise, even slight changes in vehicle mobility (wheel slip, impaired mobility, etc . . .) can result in dramatic changes for edge shapes and actions in the control set. Reasoning about edge traversability has typically been addressed by engineering environmental cost functions so that it will approximate the extra effort to execute the difficult or dangerous maneuvers in an effective manner. Typical implementation of this approach are however flawed because engineered cost functions often reason about cost on a two-dimensional map. In uneven terrain, orientation and chassis configuration play significant roles in the true cost of a trajectory through the environment. Evaluating the cost of a precomputed control set edge can be inherently flawed as the actual motion required to reach the target terminal state may deviate from the precomputed action. A control set that does not account for predictive motion model effects can generate solutions that are misguided, suboptimal, or even dangerous. This section describes the application of model-predictive trajectory generation techniques in state lattices to generate informed, feasible connections that better

reason about the interaction with the environment.

5.2.1 EDGE ADAPTATION

The model-predictive trajectory generation algorithms described in Chapter 4 can be used to correct the action that satisfies the state transition equation for any internal or external modifications of the predictive mobility model. Two types of changes to the control set are global and local control set changes. Global control set disturbances are changes in the predictive motion model that are repeatable at all other equivalent states in the motion planning graph. For example, if a vehicle mobility model changes due to a degraded mobility system (i.e. broken drive wheel), that disturbance is repeatable at all other positions and orientations where the control set would be expanded. Alternatively, local changes in the control set are ones which are dependent on the global state. Varying terrain shape, terramechanical models, and environmental effects are examples where local modifications of the control set are necessary to generate truly feasible state lattices.

The global control set disturbances can be addressed by regenerating all actions in the control set by repeating lines 8 through 11 the FEASIBLECONTROLSETGENERATION algorithm presented in Figure 5.2. Local control set modifications are most efficiently performed during sequential search of a motion planning graph. A variation of the informed state lattice technique in A* search is shown in Figure 5.4. By modifying the actions to satisfy the new state transition equations, a disconnected state lattice is corrected to achieve feasibility.

5.2.2 EXPERIMENTS

To test the effects of this algorithm, two motion plans were generated for a mobile robot using an informed and naive (fixed) state lattice with a cost function that penalized integrated attitude in a lunar-like environment. In this particular example, a planetary rover attempts to drive into a crater through a hilly and undulating environment. The predictive motion model used for this experiment compensates only for terrain shape in the vehicle simulation and does not include a wheel slip model. The cost of each action was determined by forward simulating the motion determined by the control set action with a cost function that penalized attitude over the trajectory. The specific cost functional integrated over the trajectory was a weighted sum of distance traveled and attitude:

$$\mathcal{L}(\mathbf{x}(t),\mathbf{u}(t),t) = 1 + w_{ATTITUDE}(\phi^4 + \theta^4) \tag{5.1}$$

The cost function design ensured that a positive penalty would be added for deviating from the nominally level configuration. Figure 5.5a through 5.5c shows a motion plan generated using an informed state lattice that regenerates the actions in the control set due to terrain shape. The resulting motion plan approaches the outer rim of the crater at an angle to reduce exposure to attitude and drives straight towards the goal after cresting the hill on the more gently sloping crater interior. Figures 5.5d through 5.5f show the solution generated by the naive state lattice, which differs in the crater rim approach. Rather than traversing the entire rim of the crater at an angle, the motion plan for the uninformed state lattice takes a more direct route up the face of the crater. This is because it does not reason about the

ISL-A*($\mathbf{x_I}$,$\mathbf{x_T}$)

INPUT: $\mathbf{x_I}$ (initial state), $\mathbf{x_T}(\mathbf{s})$ (target terminal state)
OUTPUT: $\mathbf{x}(t)$ (trajectory), $\mathbf{u}(\mathbf{p},\mathbf{x},t)$ (parameterized action sequence)
DATA: $\mathbf{U}(\mathbf{s_p},\mathbf{s_c})$ (control set), $\mathbf{f_{SME}}$ (state mapping equation), $\mathbf{f_{PMM}}$ (predictive motion model)

1 $\mathbf{s_{START}} \leftarrow \mathbf{f_{SME}}(\mathbf{x_I})$, $\mathbf{s_{GOAL}} \leftarrow \mathbf{f_{SME}}(\mathbf{x_T})$
2 OPEN $\leftarrow \emptyset$, CLOSED $\leftarrow \emptyset$
3 insert $\mathbf{s_{START}}$ into OPEN
4 **while** OPEN $\neq \emptyset$ **do**
5 remove $\mathbf{s_{TOP}}$ with minimum f from OPEN
6 **if** $\mathbf{s_{TOP}} = \mathbf{s_{GOAL}}$ **then**
7 **return** $\mathbf{s_{TOP}}$
  end
8 **foreach** $\mathbf{s_c} \in \mathbf{S_c}(\mathbf{s_{TOP}})$ **do**
9 $\mathbf{x_I} \leftarrow \mathbf{f_{SME}}(\mathbf{s_{TOP}})$
10 $\mathbf{x_T} \leftarrow \mathbf{f_{SME}}(\mathbf{s_c})$
11 CONSTRAINEDMPTG($\mathbf{x_I}, \mathbf{x_T}, \mathbf{U}(\mathbf{s_{TOP}}, \mathbf{s_c}), \mathbf{f_{PMM}}$)
  end
12 EXPANDCONTROLSET($\mathbf{s_{TOP}}, \mathbf{U}(\mathbf{s_{TOP}}, \mathbf{s_c})$)
13 insert $\mathbf{s_{TOP}}$ into CLOSED
end

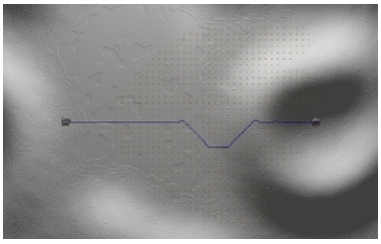
Figure 5.4: A* graph search with informed state lattices (ISL-A*). In this variation of state lattice graph construction, every action in the expanded control set is regenerated to compensate for local changes in mobility.

extra action length required to drive up the face of the crater and assumes that it is actually a shorter, cheaper path to drive.

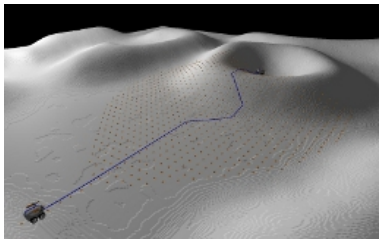
Figures 5.5g through 5.5i exhibit the experienced roll, pitch, and integrated cost of the resulting execution of the generated trajectory from the naive and informed state lattices. The naive state lattice produces a shorter trajectory than the informed state lattice (42.08m and 43.72m respectively) but experiences a higher contribution of the integrated attitude (31.65) than the informed state lattice (29.84). The result is that the informed state lattice produced a slightly lower cost solution (73.56 vs 73.73) because it could exploit more information about the actual edge costs in the environment than its naive state lattice counterpart.

5.3 ADAPTIVE STATE LATTICES

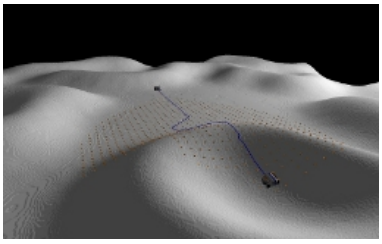
The main contribution of this thesis addresses the problems of optimality and relative optimality in motion planning search spaces for mobile robots. A state lattice provides a regular distribution of state space for sequential search to determine a safe action through an environment. If the search space is inadequate for a particular motion planning problem, an entirely different representation of the path continuum may be necessary to generate an alternative



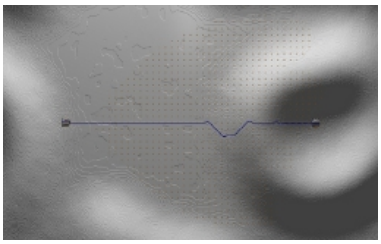
(a) informed state lattice solution (overhead)



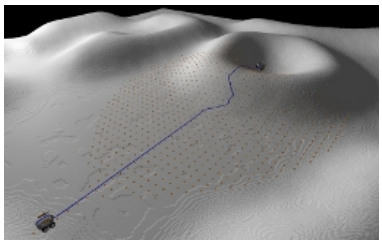
(b) informed state lattice solution (from initial state)



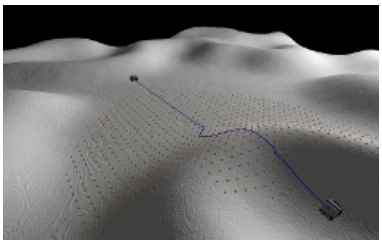
(c) informed state lattice solution (from target terminal state)



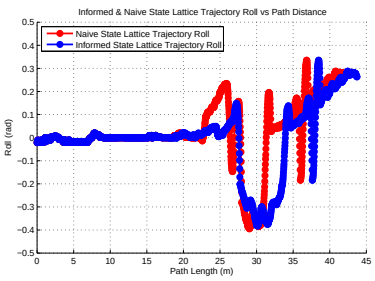
(d) naive state lattice solution (overhead)



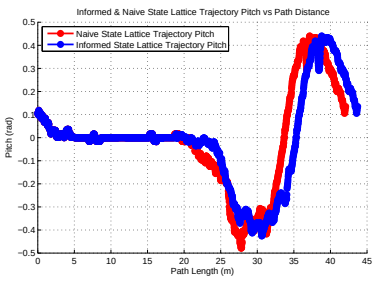
(e) naive state lattice solution (from initial state)



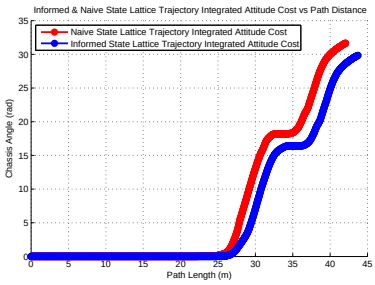
(f) naive state lattice solution (from target terminal state)



(g) informed and naive state lattice roll vs distance traveled



(h) informed and naive state lattice pitch vs distance traveled



(i) informed and naive state lattice integrated cost vs distance traveled

Figure 5.5: Entering a crater with a motion plan with informed and naive state lattices. In this example, edges in the informed state lattice are terrain-adaptive while edges in the naive state lattice are simply executions of the original actions in the control sets. The informed state lattice produces an answer that is longer but with less exposure to risk by minimizing vehicle attitude. The naive state lattice produces a suboptimal answer because it could not reason about the extra mobility costs associated with driving up the crater rim.

trajectory.

This section proposes a technique that optimizes the regular distribution of state space by locally relaxing the state mapping equation between discretized nodes and continuous states. Adaptation of the search space is based on minimizing the aggregate cost of edges through the search space, rather than the cost of the state represented by the discretized node itself. This thesis hypothesizes that local control set relaxation better represents desirable regions of the path continuum and often determines lower cost (faster, safer, and cheaper) paths through complex environments. Graph topology and feasibility are preserved during search space relaxation by regenerating actions in the modified control sets using the model-predictive trajectory generation techniques presented in Chapter 4.

5.3.1 LOCAL STATE MAPPING EQUATION OPTIMIZATION

The values of the continuous state represented by a discretized node influence the quality of the resulting control set. A technique that locally optimizes the state mapping equation between the continuous and discrete representations should improve the global optimality of continuous paths in the search space as the local state transition equation (edge) costs decrease. The use of a local state mapping equation deviates from the notion that the mapping between the discretized node and continuous state is uniformly repeated throughout the search space.

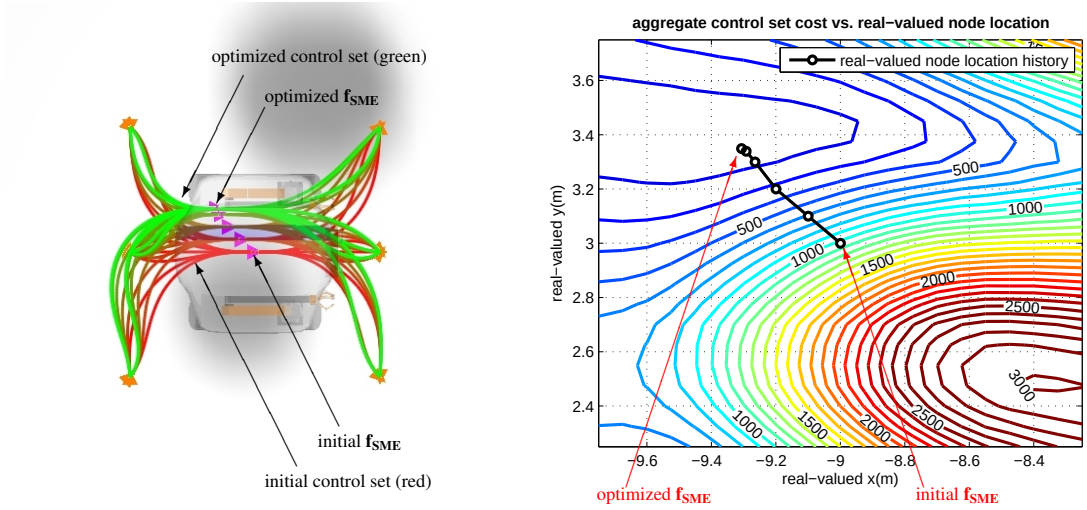
One particularly useful metric for local state mapping equation optimization is the aggregate control set cost. Moving the continuous state represented by a discretized node off a high-cost region will often decrease the associated cost of all control set edges. Using the state location itself for optimization is sub-optimal in several ways. First, trajectories determined by searching in the motion planning graph are independent of the cost of an associated state; they are function of the cost of concatenated edges used to compose the paths. Local optimization of a state that increases the associated cost of all edges connecting that state would decrease the local optimality of the motion planning graph. Second, modifying the state potentially violates feasibility of the connective edges in the control set. Lastly, small perturbations in state used to estimate partial derivatives of cost may be flat in large swaths of uniform cost. Conversely, costs evaluated as a path integral based on trajectories with small boundary state constraint perturbations often provide better guidance to local search, even if part of the trajectory is enveloped by a uniform penalty.

Consider the situation depicted in Figure 5.6. In this example, a control set composed of seven forward and seven backward actions connect the continuous states represented by the current node to fourteen child nodes. The environment consists of two high penalty regions representing unfavorable positions for the vehicle to pass through, one of which occurs on the boundary of the parent node. A cost function (J_{AG}) for state mapping equation optimization is defined as the aggregate cost of all edges between a parent node and its k children:

$$J_{AG}(\mathbf{s_p}, \mathbf{s_c}, \mathbf{U}(\mathbf{s_p}, \mathbf{s_c})) = \sum_{k=1}^{n} \left[\int_{t_{I,k}}^{t_{F,k}} \Phi(\mathbf{f_{SME}}(\mathbf{s_p}), t_I, \mathbf{f_{SME}}(\mathbf{s_c}_k), t_F) + \int_{t_{I,k}}^{t_{F,k}} \mathcal{L}(\mathbf{x}(t), \mathbf{U}(\mathbf{s_p}, \mathbf{s_c}_k), t)\, dt \right] \tag{5.2}$$

With this definition of cost, the state mapping equation can be relaxed into an locally optimal configuration for the environment. Figure 5.6a shows the original control set expansion in red and subsequent steps in the state mapping equation optimization transitioning to green. The actual continuous state represented by the parent node with the

iteratively modified state mapping equation is shown as a pink triangle pointing in the direction of state heading. The fourteen child nodes are labeled as orange triangles, which remain fixed throughout the local state mapping equation optimization process. In order to plot a sampling of the aggregate control set edge cost, only the two-dimensional position was modified in this experiment. The history of the state generated by the state mapping equation as a function of the optimization step is shown in Figure 5.6b overlaid on a sampling of the aggregate control set edge cost. Starting from the regular sampling of state space, the local state mapping equation relaxes in a locally optimal configuration between the two high-cost regions where the aggregate edge costs are nearly minimum.



(a) local optimization of the state transition equation

(b) aggregate cost vs. state transition equation values

Figure 5.6: State mapping equation optimization for a control set with fourteen child nodes. In this example, the state mapping equation for the current node is modified based on the aggregate cost of the control set state transition equations. The aggregate cost of the control set based on the optimized state mapping equation is decreased by 84% (159.83 vs 1031.67) after five optimization cycles.

A representation of the cost as a function of state mapping equation variations is shown in Figure 5.6b, which was determined by sampling the aggregate path cost at uniformly distributed sampling of modified position mapping. The technique efficiently follows cost function gradients to a locally optimal configuration using simple gradient descent techniques. The STATEMAPPINGEQUATIONOPTIMIZATION algorithm is outlined in Figure 5.7. Estimates of the partial derivative with respect to the state mapping equation are determined by forward or central differences of the aggregate control set cost. The continuous state represented by the state mapping equation is perturbed and the aggregate control set cost is recomputed. For a control set with b edges and c controllable state mapping equation dimensions, a minimum of $b(c+1)$ trajectories must be generated and evaluated in each step of the state mapping equation optimization.

STATEMAPPINGEQUATIONOPTIMIZATION($\mathbf{s_p}$,$\mathbf{S_c}(\mathbf{s_p})$,$\mathbf{U}(\mathbf{s_p},\mathbf{s_c})$,$\mathbf{f_{SME}}(s)$)

INPUT: $\mathbf{s_p}$ (parent node), $\mathbf{S_c}(\mathbf{s_p})$ (child node set), $\mathbf{U}(\mathbf{s_p},\mathbf{s_c})$ (control set), $\mathbf{f_{SME}}$ (state mapping equation)
OUTPUT: $\mathbf{f_{SME}}(\mathbf{s_p})$ (modified state mapping equation)

1 **while** $\frac{\delta J_{AG}}{\delta \mathbf{f_{SME}}(\mathbf{s_p})} \neq \mathbf{0}$ **do**
2 $\quad \mathbf{x_I} \leftarrow \mathbf{f_{SME}}(\mathbf{s_p})$
3 $\quad$ **foreach** $\mathbf{s_c} \in \mathbf{S_c}$ **do**
4 $\quad\quad \mathbf{X_T} \leftarrow \mathbf{f_{SME}}^{-1}(\mathbf{s_c})$
$\quad$ **end**
5 $\quad$ EstimateJacobianOfAggregateControlSetCost($\mathbf{x_I}, \mathbf{X_T}, \mathbf{U}(\mathbf{p},\mathbf{x},t)$)
6 $\quad \Delta\mathbf{f_{SME}}(\mathbf{s_p}) \leftarrow -\alpha \frac{\delta J_{AG}}{\delta \mathbf{f_{SME}}(\mathbf{s_p})}$
7 $\quad \mathbf{f_{SME}}(\mathbf{s_p}) \leftarrow \mathbf{f_{SME}}(\mathbf{s_p}) + \Delta\mathbf{f_{SME}}(\mathbf{s_p})$
end
8 **return** $\mathbf{f_{SME}}(\mathbf{s_p})$

ESTIMATEJACOBIANOFAGGREGATECONTROLSETCOST($\mathbf{x_I}$,$\mathbf{X_T}$,$\mathbf{U}(\mathbf{p},\mathbf{x},t)$)

INPUT: $\mathbf{x_I}$ (parent state), $\mathbf{x_T}$ (child state set), $\mathbf{U}(\mathbf{x},\mathbf{u},t)$ (control set)
OUTPUT: $\frac{\delta J_{AG}}{\delta \mathbf{f_{SME}}(\mathbf{s_p})}$ (aggregate control set cost Jacobian)

1 $c_{controlset,original} \leftarrow$ ESTIMATEAGGREGATECONTROLSETCOST($\mathbf{x_I}$,$\mathbf{X_T}$,$\mathbf{U}(\mathbf{p},\mathbf{x},t)$)
2 **foreach** $x \in \mathbf{x_I}$ **do**
3 $\quad x \leftarrow x + h$
4 $\quad c_{controlset,perturbed} \leftarrow$ ESTIMATEAGGREGATECONTROLSETCOST($\mathbf{x_I}$,$\mathbf{X_T}$,$\mathbf{U}(\mathbf{p},\mathbf{x},t)$)
5 $\quad x \leftarrow x - h$
6 $\quad \frac{\delta J_{AG}}{\delta \mathbf{f_{SME}}(\mathbf{s_p})} \leftarrow \frac{c_{controlset,perturbed} - c_{controlset,original}}{h}$
end
7 **return** $\frac{\delta J_{AG}}{\delta \mathbf{f_{SME}}(\mathbf{s_p})}$

ESTIMATEAGGREGATECONTROLSETCOST($\mathbf{x_I}$,$\mathbf{X_T}$,$\mathbf{U}(\mathbf{p},\mathbf{x},t)$)

INPUT: $\mathbf{x_I}$ (parent state), $\mathbf{X_T}$ (child state set), $\mathbf{U}(\mathbf{p},\mathbf{x},t)$ (control set)
OUTPUT: $c_{controlset}$ (control set cost)
DATA: $\mathbf{f_{PMM}}$ (predictive motion model)

1 $c_{controlset} \leftarrow 0$
2 **foreach** $\mathbf{x_T} \in \mathbf{X_T}$ **do**
3 $\quad$ CONSTRAINEDMPTG($\mathbf{x_I}, \mathbf{x_T}, \mathbf{u}(\mathbf{p},\mathbf{x},t), \mathbf{f_{PMM}}$)
4 $\quad c_{controlset} \leftarrow c_{controlset} + \int_{t_I}^{t_F} \Phi(\mathbf{x_I}, t_I, \mathbf{x_T}, t_F) + \int_{t_I}^{t_F} \mathcal{L}(\mathbf{x}(t), \mathbf{u}(\mathbf{p},\mathbf{x},t), t)\, dt$
end
5 **return** $c_{controlset}$

Figure 5.7: State Mapping Equation Optimization.

5.3.2 CONTROL SET ADAPTATION

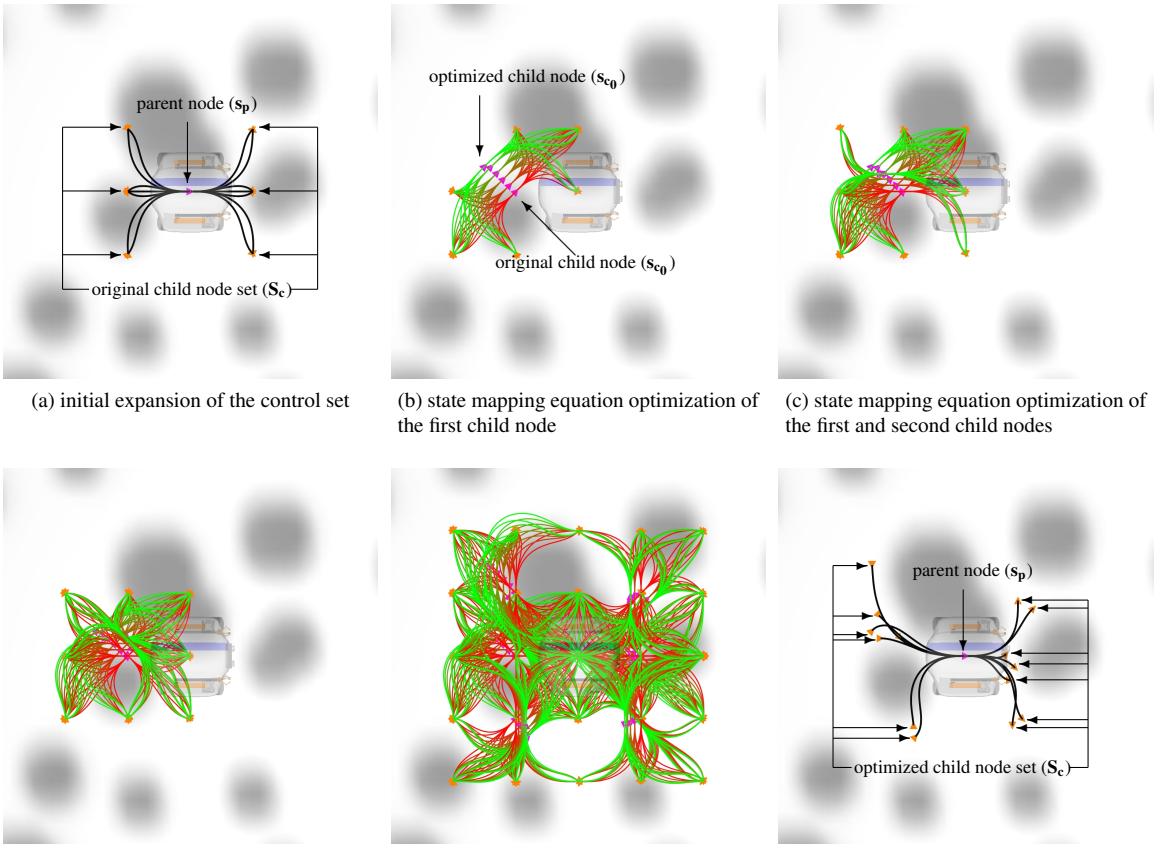
Sequential search processes determine the search direction based on the edge costs of the descendant nodes. State mapping equation optimization of nodes after expansion in sequential search (after the node has been popped from the top of the OPEN list but before it has been pushed onto the CLOSED list) is suboptimal and does not guarantee that the lowest cost node has been expanded. While this approach may generate lower cost trajectories than motion plans generated with a fixed state lattice, this approach is greedy because it attempts to improve only the minimum cost nodes in the graph. If an edge transitioning to a descendant node on the OPEN list is expensive, it may never be adapted to the environment, even if it leads to a significantly better solution.

A better alternative is to apply state mapping equation optimization to all descendant nodes as they are expanded to provide better estimates about the actual cost of traversing an edge. Search in this motion planning graph is inherently correct as edge costs are accurate when the are placed on the OPEN list. Figure 5.8 shows control set adaptation in a three-dimensional motion planning graph (x,y,ψ) that modifies the state mapping equation of the resulting children nodes:

The original control set for the current node (Figure 5.8a) consists of fourteen child nodes which vary in position and heading. Control set adaptation is applied to this control set as several of the edges pass through and terminate in high-cost regions. The control set adaptation example in Figure 5.8 is shown as the sequential optimization of individual child nodes, although in practice this process can also occur simultaneously. Figure 5.8b shows the state mapping equation optimization for the first child nodes. The optimized control set for the first child node maneuvers around the high-cost region to the lower-left of the state represented by the current node and modifies the shape of the edge from the current to this particular child node in the process. Optimizing the entire connectivity is important because relaxing the single edge may result in a sub-optimal configuration for the rest of the control set. This technique is repeated for all children in the search space (Figures 5.8b through 5.8e) until the state mapping equation for all children has been completed. Figure 5.8f shows the resulting modified control set from the current node. Compared to the initial control set in Figure 5.8a, the relaxed control set conforms to the environment and effectively maneuvers around obstacles. In the adapted control set, several of the real-valued states represented by the child nodes now lie in the narrow minimum-cost valley between the high cost regions. The aggregate control set cost was reduced by 29% (1256.47 vs 1781.75), which does not include the improved edge costs for all of the descendant node control sets.

Two variations of graph search algorithms for adaptive state lattices are shown in Figure 5.9. The first approach, Greedy Adaptive State Lattice A* search (GASL-A*), optimizes only nodes that have been expanded and put onto the closed list. The second approach, Adaptive State Lattice A* search (ASL-A*), optimizes the state mapping equation for all nodes before they are added to the open list. Optimizing the state mapping equation for all nodes on the OPEN list make the graph more informed about the actual cost to traverse edges between candidate nodes. The only significant difference between these two algorithms is when the STATEMAPPINGEQUATIONOPTIMIZATION routine is called.

To test the importance of the differences between the two algorithms, Figure 5.10 shows a comparison of motion plans produced with fixed, greedy adaptive, and adaptive state lattices in two environments. The first environment, shown in Figures 5.10a, 5.10c, and 5.10e, was specifically designed to highlight the differences between the fixed,



(a) initial expansion of the control set

(b) state mapping equation optimization of the first child node

(c) state mapping equation optimization of the first and second child nodes

(d) state mapping equation optimization of the first, second, and third child nodes

(e) state mapping equation optimization of k child nodes

(f) the resulting modified control set after state mapping equation optimization of the children nodes

Figure 5.8: Adaptation of children nodes during a control set expansion. An initial control set is expanded in 5.8a consisting of 14 children nodes. The minimum-cost state transition equation is computed from this set to determine the best route for search space expansion. The state mapping equation is optimized for each of the 14 children nodes before expansion as the cost associated with each edge changes. Despite not modifying the state mapping equation for the parent node, the aggregate cost of the resulting control set decreases by 29% (1256.47 vs 1781.75).

GASL-A*($\mathbf{x_I}$,$\mathbf{x_T}$)

INPUT: $\mathbf{x_I}$ (initial state), $\mathbf{x_T}(\mathbf{s})$ (target terminal state)
OUTPUT: $\mathbf{x}(t)$ (trajectory), $\mathbf{u}(\mathbf{p},\mathbf{x},t)$ (parameterized action sequence)
DATA: $\mathbf{U}(\mathbf{s_p},\mathbf{s_c})$ (control set), $\mathbf{f_{SME}}$ (state mapping equation)

$\mathbf{s_{START}} \leftarrow \mathbf{f_{SME}}(\mathbf{x_I})$, $\mathbf{s_{GOAL}} \leftarrow \mathbf{f_{SME}}(\mathbf{x_T})$
OPEN $\leftarrow \emptyset$, CLOSED $\leftarrow \emptyset$
insert $\mathbf{s_{START}}$ into OPEN
while OPEN $\neq \emptyset$ **do**
    remove $\mathbf{s_{TOP}}$ with minimum f from OPEN
    if $\mathbf{s_{TOP}} = \mathbf{s_{GOAL}}$ **then**
        return $\mathbf{s_{TOP}}$
    end
    STATEMAPPINGEQUATIONOPTIMIZATION($\mathbf{s_{TOP}}$,$\mathbf{S_c}(\mathbf{s_{TOP}})$,$\mathbf{U}(\mathbf{s_{TOP}},\mathbf{S_c}(\mathbf{s_{TOP}}))$,$\mathbf{f_{SME}}(s)$)
    EXPANDCONTROLSET($\mathbf{s_{TOP}}$,$\mathbf{U}(\mathbf{s_{TOP}},\mathbf{s_c})$)
    insert $\mathbf{s_{TOP}}$ into CLOSED
end

ASL-A*($\mathbf{x_I}$,$\mathbf{x_T}$)

INPUT: $\mathbf{x_I}$ (initial state), $\mathbf{x_T}(\mathbf{s})$ (target terminal state)
OUTPUT: $\mathbf{x}(t)$ (trajectory), $\mathbf{u}(\mathbf{p},\mathbf{x},t)$ (parameterized action sequence)
DATA: $\mathbf{U}(\mathbf{s_p},\mathbf{s_c})$ (control set), $\mathbf{f_{SME}}$ (state mapping equation)

$\mathbf{s_{START}} \leftarrow \mathbf{f_{SME}}(\mathbf{x_I})$, $\mathbf{s_{GOAL}} \leftarrow \mathbf{f_{SME}}(\mathbf{x_T})$
OPEN $\leftarrow \emptyset$, CLOSED $\leftarrow \emptyset$
insert $\mathbf{s_{START}}$ into OPEN
while OPEN $\neq \emptyset$ **do**
    remove $\mathbf{s_{TOP}}$ with minimum f from OPEN
    if $\mathbf{s_{TOP}} = \mathbf{s_{GOAL}}$ **then**
        return $\mathbf{s_{TOP}}$
    end
    foreach $\mathbf{s_c} \in \mathbf{S_c}(\mathbf{s_{TOP}})$ **do**
        STATEMAPPINGEQUATIONOPTIMIZATION($\mathbf{s_c}$,$\mathbf{S_c}(\mathbf{s_c})$,$\mathbf{U}(\mathbf{s_c},\mathbf{S_c}(\mathbf{s_c}))$,$\mathbf{f_{SME}}(s)$)
    end
    EXPANDCONTROLSET($\mathbf{s_{TOP}}$,$\mathbf{U}(\mathbf{s_{TOP}},\mathbf{s_c})$)
    insert $\mathbf{s_{TOP}}$ into CLOSED
end

Figure 5.9: Two variations of graph search with Adaptive State Lattices. The first approach, Greedy (Post-Sort) Adaptive State Lattice A* search (GASL-A*), optimizes only nodes that have been expanded and put onto the closed list. The second approach, Adaptive (Pre-Sort) State Lattice A* search (ASL-A*), optimizes the state mapping equation for all nodes before they are added to the open list. Optimizing the state mapping equation for all nodes on the OPEN list make the graph more informed about the actual cost to traverse edges between candidate nodes.

greedy adaptive, and (fully) adaptive state lattice search. In this example, a vehicle must move from an initial state to a target goal state in an environment where the vehicle must pass through a narrow gap or take a much longer route around the obstacle. The fixed state lattice is not dense enough to pass through the beginning of narrow gap and generates a motion plan that goes around the perimeter of the obstacle. The greedy adaptive state lattice searches slightly deeper into the gap, but since all of the frontier edges pass in or near the high-cost regions, their children are never expanded during search. Instead the motion plan wraps around the boundary of the high-cost region similarly to the fixed state lattice solution. The adaptive state lattice solution, which optimizes all children nodes during control set expansion, is able to pass through the gap and produce a cheaper and more direction solution through the environment, even though all of the examples use state lattices with the same original control set and graph topology.

To test the differences between the three approaches in a more natural environment, motion plans were generated and compared using a randomly generated cost map. Figures 5.10b, 5.10d, and 5.10f show the search spaces and resulting solutions for the fixed, greedy adaptive, and adaptive state lattices respectively. As expected, the fixed state lattice produces the most expensive solution, the greedy adaptive state lattice produces a better motion plan, and the adaptive state lattice produces the best answer. The differences between the greedy adaptive and adaptive state lattice solutions again come from not optimizing all of the children nodes in the graph. The adaptive state lattice is able to pass through a narrow gap bounded by several high-cost regions that the greedy adaptive state lattice could not, producing a lower cost solution.

Optimizing the descendant nodes is important for control set adaptation in order to determine the best direction to search. The examples showed that in situations where the best solution is desired, a fully adaptive state lattice is preferred over its greedy adaptive variant.

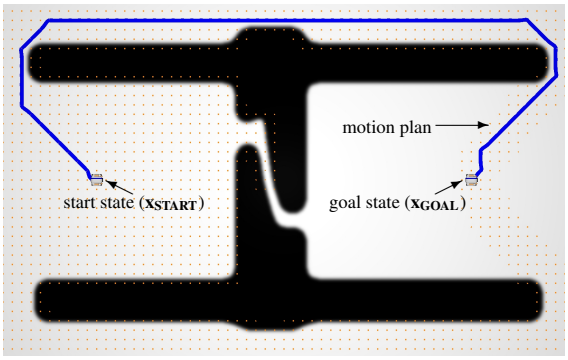
5.3.3 EXPERIMENTS

Two experiments were designed to evaluate the optimality and relative optimality benefits of the proposed adaptive state lattice algorithms. First, an experimental study compares the quality, runtime, and memory requirements of the fixed and adaptive state lattices to evaluate the differences between the fixed and adaptive state lattice solutions. A second experiment extends this work for chassis optimization for mobile robots operating in rough terrain, where the configuration state lattice dimension is relaxed to the environment.

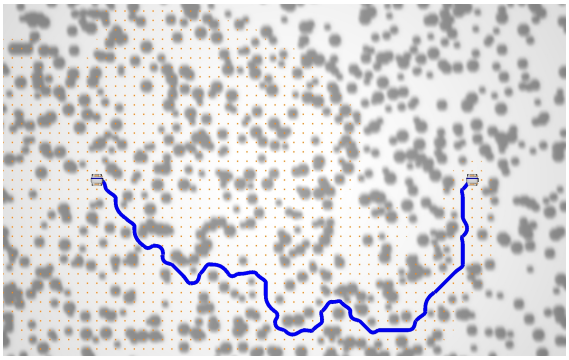
FIXED AND ADAPTIVE STATE LATTICES IN INCREASINGLY COMPLEX ENVIRONMENTS

EXPERIMENTAL DESIGN

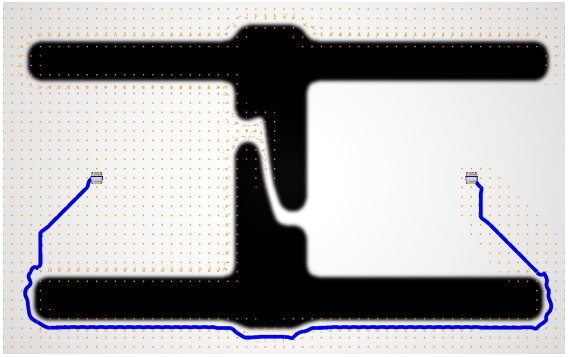
This state lattice comparison experiment is designed as a series of motion planning problems in twenty-one increasingly complex environments. For each technique, eleven control sets with varying outdegree for each of the four regularly sampled state discretizations in Table 5.1 are searched for a minimum-cost solution using heuristic search. Each environment, discretization, and control set are used with a fixed state lattice, a single-step adaptive state lattice, and a five-step adaptive state lattice. The single-step adaptive state lattice can execute at most one step of state mapping equation optimization whereas the five-step bounded adaptive state lattice technique can iterate the state mapping



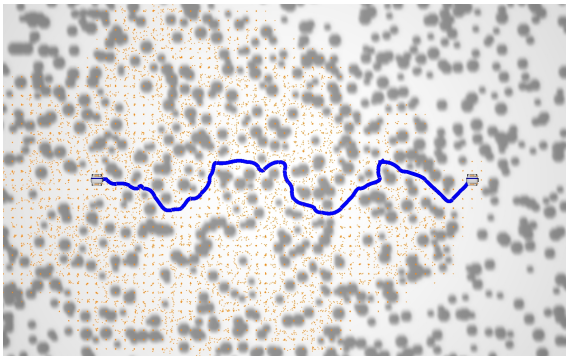
(a) fixed state lattice motion plan, which resulted in a path cost of 94.936 and took 12.191 seconds to compute



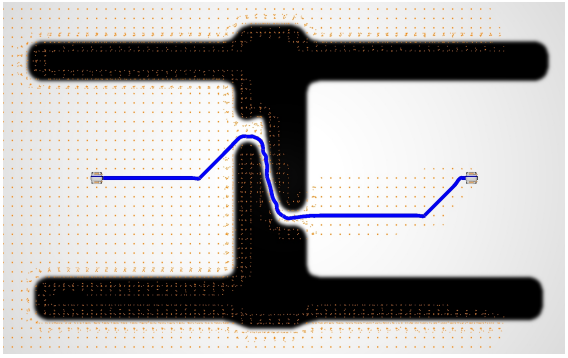
(b) motion plan generated with a fixed state lattice, which resulted in a path cost of 81.634



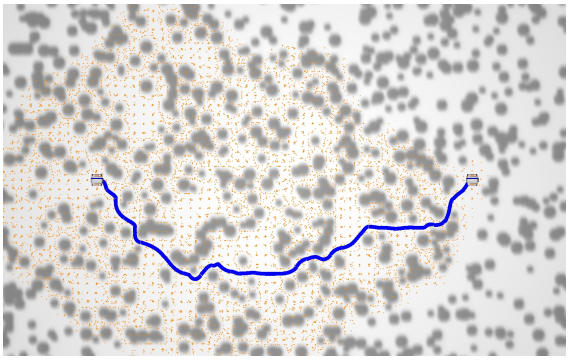
(c) greedy adaptive state lattice motion plan, which resulted in a path cost of 94.523 and took 63.677 seconds to compute



(d) motion plan generated with a greedy adaptive state lattice, which resulted in a path cost of 66.545



(e) adaptive state lattice motion plan, which resulted in a path cost of 85.000 and took 75.941 seconds to compute



(f) motion plan generated with a adaptive state lattice, which resulted in a path cost of 62.211

Figure 5.10: Comparison of solutions generated by fixed, greedy adaptive, and fully adaptive state lattices. In 5.10a, 5.10c, and 5.10e, motion plans are generated using the three approaches with the same control set. The fixed and greedy adaptive state lattice are unable to pass through the narrow gap obstacles because unoptimized expanded nodes in that region intersect with areas of high cost. The same experiment is repeated in a more natural environment in 5.10b, 5.10d, and 5.10f, which the adaptive state lattice again outperforms the fixed and greedy adaptive state lattices.

equation optimization up to five times (as shown in Figure 5.8b). A total of 2,772 path plans were generated and compared to estimate the optimality and relative optimality of the presented search space construction methods.

Search Space Discretization Label	Position Discretization (m)	Heading Discretization (rad)	Potential States in a $100m \times 100m$ costmap
S1	1.0	$\pi/4$	81,608
S2	1.0	$\pi/8$	163,216
S3	0.5	$\pi/4$	323,208
S4	0.5	$\pi/8$	646,416

Table 5.1: Position and heading discretization for the four evaluated search spaces.

The metrics for performance of the fixed and adaptive search spaces are the resulting path cost and the computational resources required (CPU time and memory) to generate the plan. In this experiment, path cost is represented as a combined function of trajectory distance and risk. Risk is determined for a trajectory as the line integral of cost in the costmap.

The test platform used for these experiments used a single core from an 2.4 GHz Intel Core 2 Quad processor with 4 GB of RAM. The implementation of the state lattice motion planner and model-predictive trajectory generator operated on a single thread. As both the fixed and adaptive motion template expansion could benefit equally by multi-threading (by splitting the workload of simulating and estimating the cost of all edges in the motion template), multi-threading the application is not necessary to provide a proper performance comparison.

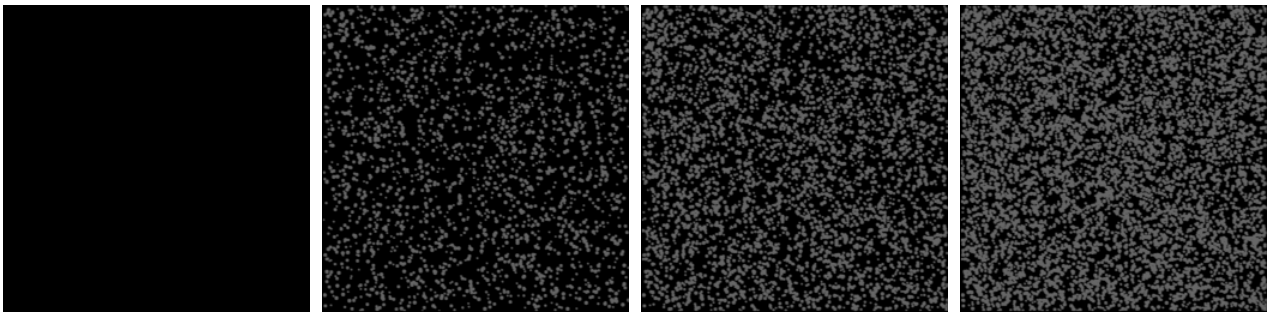
The experiments used a sequence of 21 cost maps with size 1024×1024 with a $10cm$ resolution and varying risk density to compare the performance of the proposed approaches in increasingly complex environments. The label of each costmap ("0" through "20") represents an increasing level of complexity and obstacle density in the environment. Four examples of the costmaps used in the experiment are shown in Figure 5.11. These costmaps were produced by subsequently adding randomized regions of high costs between 0 (no risk) and 255 (high risk) to the previous cost map. To prevent the situation where the vehicle starts or ends on a high-cost area, a region around those states were kept free during the randomized map construction process. Only one query (s_{START} to s_{GOAL}) was used for each combination of costmap, control set, and state lattice construction technique.

RESULTS

The single-step and multi-step adaptive state lattices were shown to improve the motion plan optimality in nearly all environments and the relative optimality in particularly complex environments. This section first illustrates several detailed examples of motion plans generated by the fixed and adaptive state lattice techniques. That discussion follows with analysis of the optimality and relative optimality of solutions generated throughout the experiment.

SELECTED PATH COMPARISONS

Figure 5.12 show the planned paths and a representation of the underlying search space for three sampling densities of motion templates in an obstacle field using fixed and multi-step adaptive state lattice algorithms. In these plots, the blue line represents the generated path while the orange triangles represent the real-valued location of nodes on the



(a) cost map "0" (b) cost map "5" (c) cost map "10" (d) cost map "15"

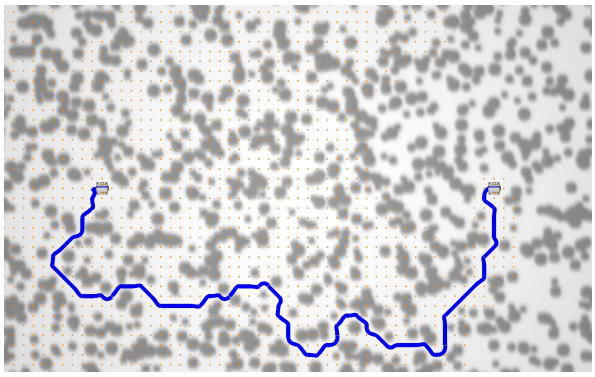
Figure 5.11: A sampling of cost maps used during the adaptive state lattice experiments. The colors are inverted from the other motion planning example figures, where here dark colors indicate "free" or "low risk" regions while light color represent "high risk" areas.

open and closed lists. Notice that in the fixed search space, the location of these nodes is uniform and regular in position and heading. In the adaptive state lattice, the real-valued location of these nodes are allowed to adjust and conform to the environment through the state mapping equation optimization. In regions in or near high cost regions, the location of these nodes adjust so that the aggregate cost of the edges passing through these nodes is minimized. The nodes actually cluster in the narrow gaps between high-cost regions. This results in a denser sampling of the path continuum in regions where, as previously shown in Figure 5.10, the inclusion of a single edge can dramatically influence the optimality of the resulting motion plan. The regular sampling is not affected in open regions of the adaptive search spaces as local optimization of the state mapping equation does not decrease the aggregate edge cost of the control set.

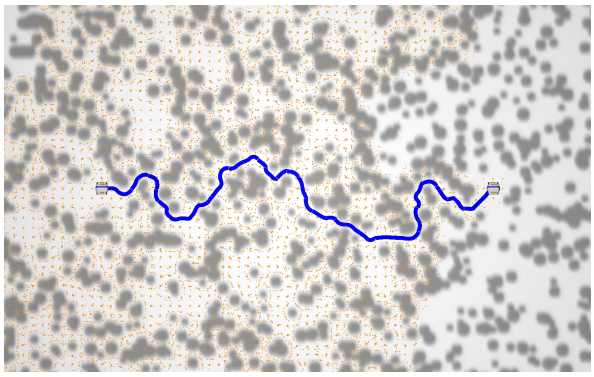
In Figures 5.12a, 5.12c, and 5.12e, the paths are generated using a fixed motion template and/or discretization of increasing density. The coarsest sampling fixed state lattice is not expressive enough to navigate directly to the goal and is forced to take a longer, circuitous route to reach the goal. The medium sampling fixed state lattice produces a slightly better answer because of the increased expressiveness of the motion template, but still plans a somewhat inefficient route. The dense sampling fixed state lattice takes a relatively direct route to the goal, but compared to the coarse sampling representation, the inefficient search takes significantly longer to generate the solution.

The same control sets are applied with the multi-step adaptive in Figures 5.12b, 5.12d, and 5.12f. For each of the three control set representations, the adaptive state lattice produces a lower-cost solution than any of the fixed state lattice examples. The most interesting comparison exists between the coarse adaptive state lattice and the dense fixed state lattice. The coarse adaptive state lattice produces an answer that is 3.8% lower cost (97.335 vs 101.179), takes 18.4% less time to compute (95.047 vs 116.466), and uses less memory to represent the search space. Since the coarse adaptive state lattice generated a better solution faster than the dense fixed state lattice, it achieves a higher degree of relative optimality.

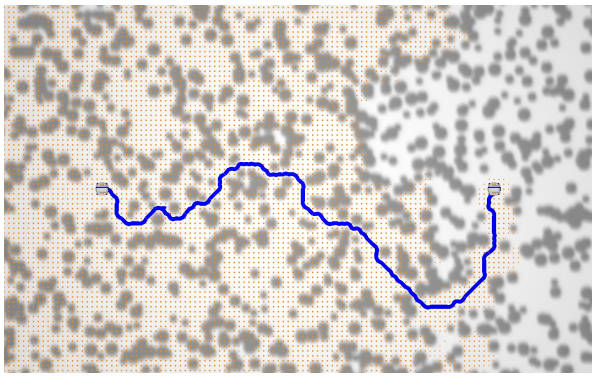
There are two other interesting features in example shown in Figure 5.12. First, the adaptive state lattice solutions



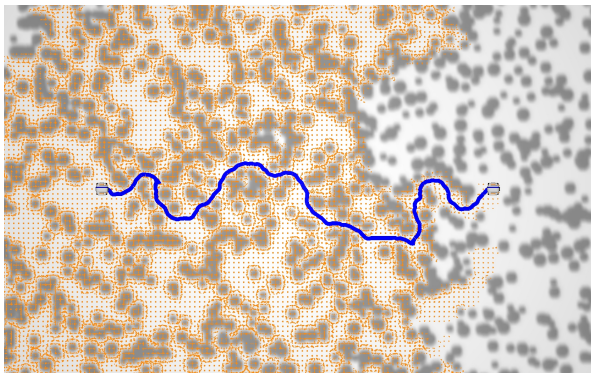
(a) solution for a low sampling fixed motion template, which took 5.413 seconds and resulted in a path cost of 135.754



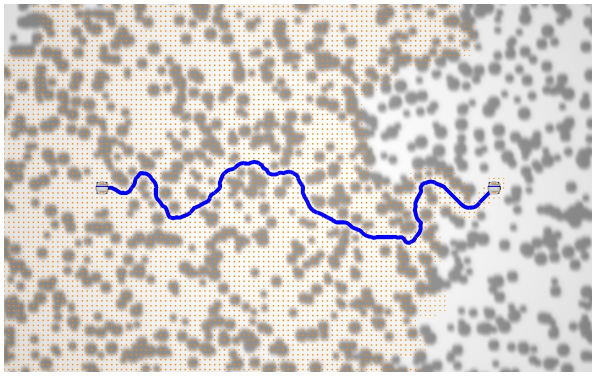
(b) solution for a low sampling adaptive motion template, which took 95.047 seconds and resulted in a path cost of 97.335



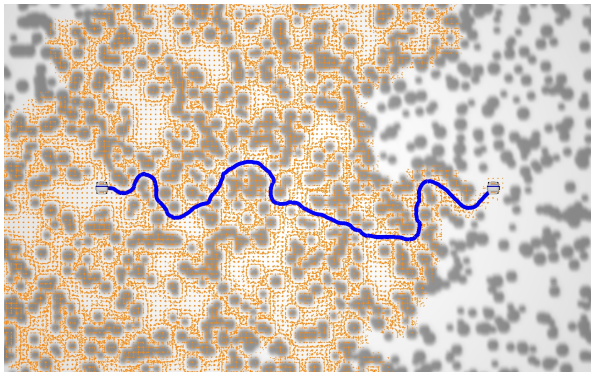
(c) solution for a medium sampling fixed motion template, which took 44.121 seconds and resulted in a path cost of 130.915



(d) solution for a medium sampling adaptive motion template, which took 269.244 seconds and resulted in a path cost of 103.604



(e) solution for a high sampling fixed motion template, which took 116.466 seconds and resulted in a path cost of 101.179



(f) solution for a high sampling adaptive motion template, which took 1066.260 seconds and resulted in a path cost of 85.845

Figure 5.12: Planned paths using fixed and adaptive state lattices of varying sampling density. Specifically note that the low sampling, adaptive state lattice produced a better solution faster than the high-sampling fixed motion template.

Test Label	Outdegree	Search Space Discretization Label	Fixed/Adaptive	Runtime (sec)	Path Cost
a	14	S1	Fixed	5.413	135.754
b	14	S1	Adaptive	95.047	97.335
c	11	S4	Fixed	44.121	130.915
d	11	S4	Adaptive	269.244	103.604
e	29	S4	Fixed	116.466	101.179
f	29	S4	Adaptive	1066.260	85.845

Table 5.2: Parameters, runtime, and path costs for six example planned paths shown in Figure 5.12.

of varying control set density all belong to nearly the same homotopy class. The finest discretization of the adaptive state lattice slightly deviates from this by navigating around a single obstacle differently. The second important feature is that the coarse and medium density solutions deformed into the higher quality paths produced by the adaptive search spaces without passing over obstacles or high-cost regions. Relaxation in the form of local state mapping equation optimization during search space construction enables high quality motion plans to be produced without modifying the graph topology. Trajectory deformation before or during search is typically better than trajectory relaxation after search.

PATH COST VERSUS SAMPLING

Figure 5.13 shows the path cost versus the motion template outdegree for the fixed and adaptive state lattices with varying sampling in six of the twenty-one increasingly complex environments. Each of the search space construction techniques (fixed state lattice, single-step adaptive state lattice, and the five-step adaptive state lattice) varied in both discretization and outdegree for these experiments. Four lines with a single color and varying markers are used to represent the different actions produced with particular sampling resolutions (S1, S2, S3, and S4). Typically the densest representation (S4) in fixed and adaptive search spaces produced the best solutions while the coarsest discretizations (S1) generated the least optimal answers for a particular method. Another constant trend is that increasing the position resolution (S3) provides a more significant improvement in path cost than increasing the heading resolution (S2).

The paths produced by the fixed and adaptive search spaces are identical in the open environment represented in Figure 5.13a. In this example, the control set adaptation does not change the local state mapping equation in any of the motion template expansions as the minimum-cost configuration has already been achieved. Figure 5.13b shows that the best quality of solutions generated by the fixed state lattice are about equal with the coarsest discretizations of the multi-step adaptive search space. The densest multi-step adaptive search spaces produce the best quality of solutions in this example, a trend that continues through Figures 5.13c through 5.13f.

As the complexity of the costmap increases, the difference between the solution quality of the fixed, single-step adaptive, and multi-step adaptive state lattices becomes more significant. In the most complex environment tested, the coarsest single-step adaptive state lattice performs roughly as well as the densest fixed state lattice for a particular outdegree.

These graphs demonstrate that the optimality of the resulting motion plan can typically be improved by applying search space adaptation in the graph search. The adaptive motion template outperformed its fixed motion template counterpart in nearly all situations without exploiting knowledge of the target terminal state location. The complexity

of the environment and the expressiveness of the search space all play a role in the difference in the quality of the solution produced by each technique.

PATH COST VERSUS RUNTIME

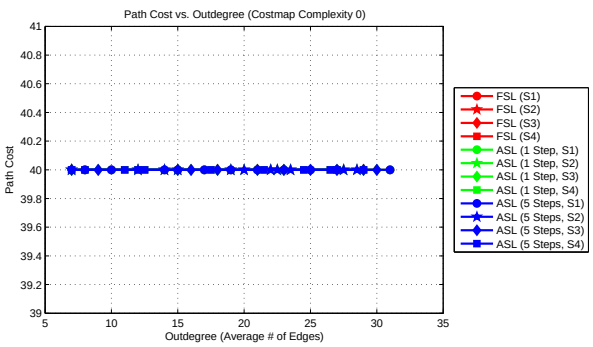
There are many mobile robot applications where minimizing the risk, regardless of the computational burden required, is ideal. Platforms which move slowly through the environment, which are difficult (if not impossible) to repair or recover in the event of failure, and those that incur large costs for motion execution, are all situations where it is better to spend the extra time to produce the best path through the environment. Conversely, there are many situations where producing a path slowly is not desired or is potentially dangerous to the platform, such as urban or field mobile robot navigation. In these applications, it also becomes important to use the technique that provides the best solution given a fixed amount of runtime. For the same set of experiments discussed in Section 5.3.3, Figure 5.14 shows the cost of the generated motion plans plotted against runtime for six environment complexities to measure the relative optimality of the techniques.

Relative optimality can be approximated in Figures 5.14a through 5.14f by tracing a vertical line through any point in the plot. The lowest intersection with the projected line provides the best solution generated given a fixed amount of time. In the open environment represented in Figure 5.14a, both techniques produce a path with the same cost, although the adaptive state lattice produces the answer more slowly than the fixed search space. This is not surprising as the adaptive state lattice inspects the gradient of the aggregate cost of the control set (even though it is uniformly zero in this environment), whereas the fixed search space does not. In Figures 5.14b and 5.14c, the dense fixed control set outperforms the single-step and multi-step adaptive state lattice for a fixed amount of runtime. Although the adaptive state lattices produced answers with a higher degree of optimality in the costmaps represented between Figures 5.14a through 5.14c, the fixed state lattice exhibited a higher degree of relative optimality since it was able to produce better answers more quickly for particular values of outdegree.

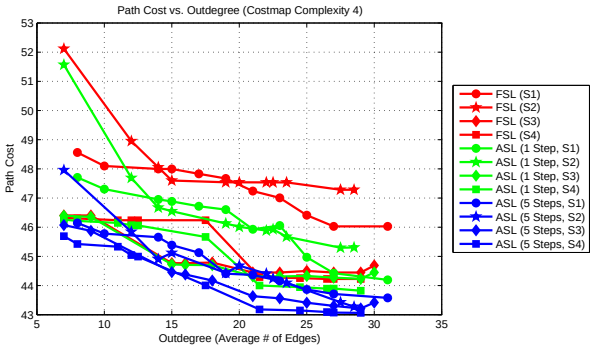
As the environmental complexity continues to increase in Figures 5.14d through 5.14f, the adaptive state lattices begin to outperform the fixed state lattice representations on the basis of relative optimality. For particular coarse resolutions of the single and multi-step adaptive state lattices, paths with lower cost are generated more quickly than in dense fixed search spaces. The added cost of searching an increased number of edges and nodes becomes more significant as environmental complexity increases and the heuristic becomes less accurate. A coarse search space that can more efficiently represent important portions of the path continuum with the fewest number of nodes and edges possible is far more efficient to search and balances out the added cost of state mapping equation optimization.

MEMORY VERSUS SAMPLING

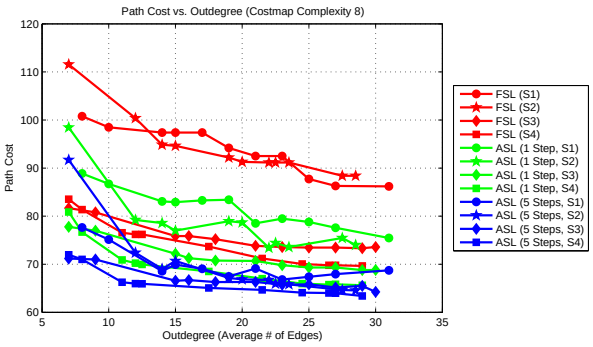
The last important measure of performance is the memory required to represent the search space by each technique. The results thus far have demonstrated that increasing the density and outdegree of the search space is an effective technique for producing lower-cost motion plans using fixed state lattices in complex environments. The penalty for increasing the search space resolution is the memory required to store the motion planning graphs. The memory



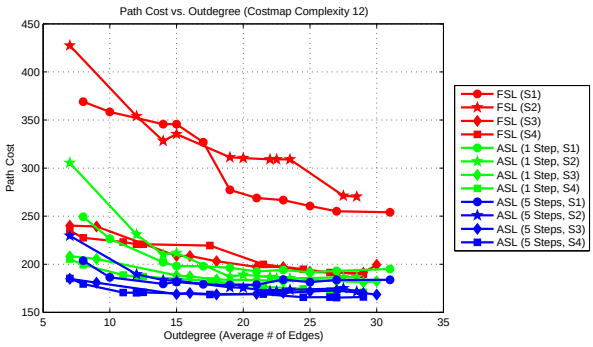
(a) fixed and adaptive state lattice comparison for costmap "0"



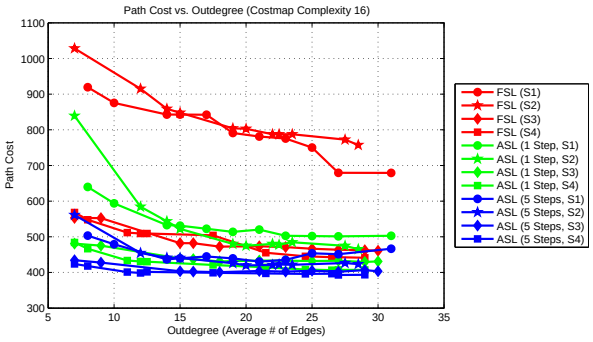
(b) fixed and adaptive state lattice comparison for costmap "4"



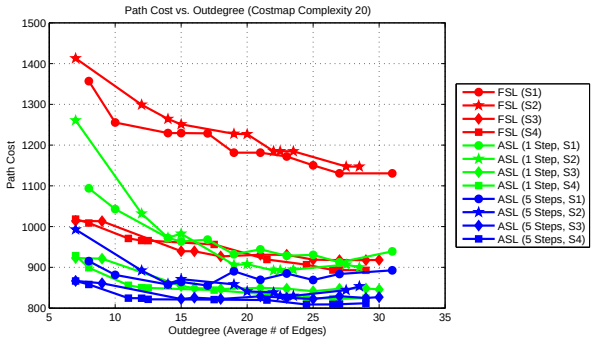
(c) fixed and adaptive state lattice comparison for costmap "8"



(d) fixed and adaptive state lattice comparison for costmap "12"

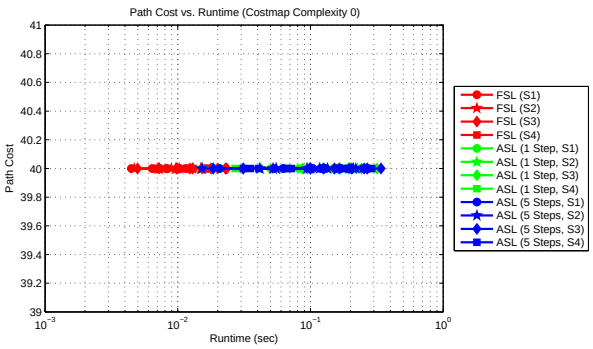


(e) fixed and adaptive state lattice comparison for costmap "16"

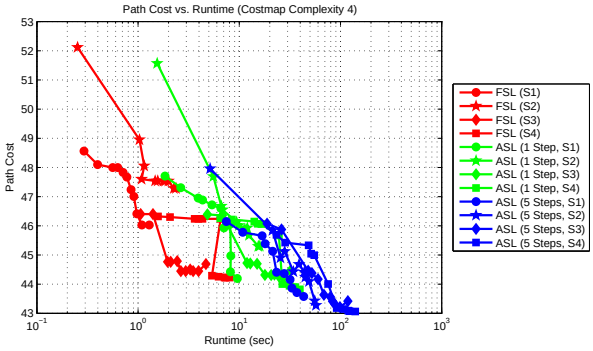


(f) fixed and adaptive state lattice comparison for costmap "20"

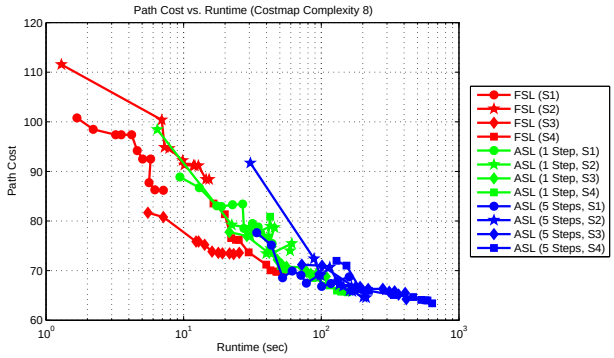
Figure 5.13: Path cost vs. outdegree for fixed and adaptive state lattices over varying risk densities. In nearly all of the examples shown, the paths produced by varying degrees of state lattice optimization were lower overall cost than those of the fixed state lattices. As the complexity of the environment increases, the difference between the quality of solutions becomes more significant as the dominant factor shifts from minimum path length to minimizing overall cost.



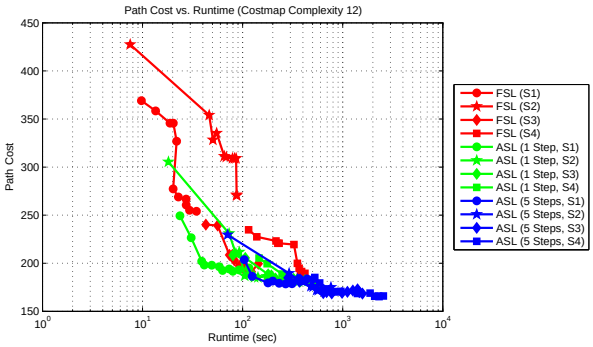
(a) fixed and adaptive state lattice comparison for costmap "0"



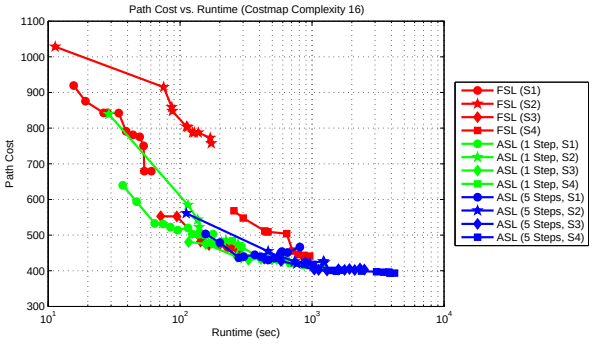
(b) fixed and adaptive state lattice comparison for costmap "4"



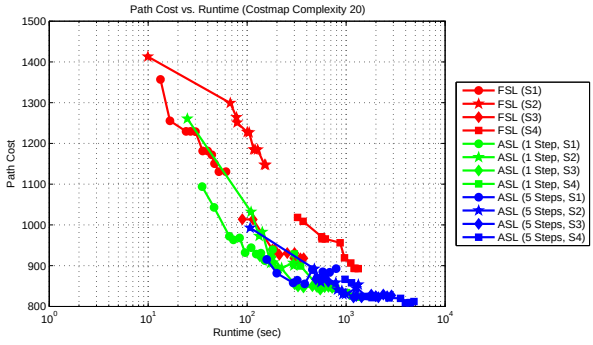
(c) fixed and adaptive state lattice comparison for costmap "8"



(d) fixed and adaptive state lattice comparison for costmap "12"



(e) fixed and adaptive state lattice comparison for costmap "16"



(f) fixed and adaptive state lattice comparison for costmap "20"

Figure 5.14: Path cost vs. runtime for fixed and adaptive state lattices over varying risk densities. As expected in simple, open environments the fixed search space produces answers faster than the adaptive search space. As the environment becomes more complex, it can be faster to use a low-outdegree, single-step adaptive search space than to use a high-outdegree fixed search space for a particular level of optimality. Likewise, for a fixed amount of runtime, a low-outdegree, single-step adaptive state lattice produces a better answer in less time than a denser, fixed search space.

footprint required to represent a discretized search space increases roughly exponentially with the resolution. The largest consumer of memory in the graph search are the open and closed lists, so the sum of their size is used as a metric to measure memory performance. Figure 5.15 shows the size of the OPEN and CLOSED lists as a function of the outdegree for the four search space discretizations (S1, S2, S3, and S4) in six of the twenty-one cost maps:

With the exception of the open environment represented in Figure 5.15a, several interesting trends emerge from this data. First, the size of the OPEN and CLOSED lists increases roughly exponentially with sampling discretization as expected. In each costmap, the OPEN and CLOSED lists in the S1 discretization use approximately one-eighth of the OPEN and CLOSED lists in the S4 discretization. Likewise, the OPEN and CLOSED lists in the S2 and S3 discretization occupy roughly one-quarter and one-half of the OPEN and CLOSED list space from the S4 discretization respectively. Doubling the position resolution increases the size of the open list twice as much as doubling the heading resolution because both the x and y discretization are halved.

The memory footprint of a system plays a part in the concept of relative optimality. Even if a dense search space generates an reasonable answer quickly (as shown in Figures 5.13b and 5.13c), if a significant amount of memory is consumed, it may not be the best search space for a particular application. Adaptive state lattice techniques can also reduce the memory required to generate a motion plan by better focusing the graph search. In the context of mobile robots, if the energy is available, a process can always take longer to generate a solution. If a system runs out of physical memory, which is often shared with other on board systems, search would terminate and not return a solution. Adaptive state lattices enables search to more efficiently represent the important portions of the path continuum, which may be particularly suited for particular applications with limited hardware resources.

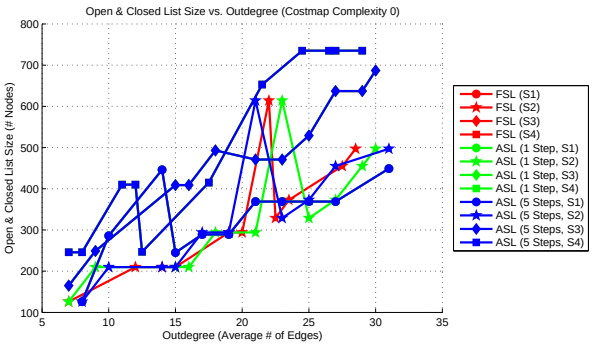
ADAPTIVE CHASSIS OPTIMIZATION STATE LATTICES

This concept can be extended into the problem of determining motion and chassis configuration for a planetary rover operating in rough terrain. This problem is related to the one described in Section 4.1.4, where a mobile robot must reason about the freedoms of the chassis configuration in order to move effectively through the environment.

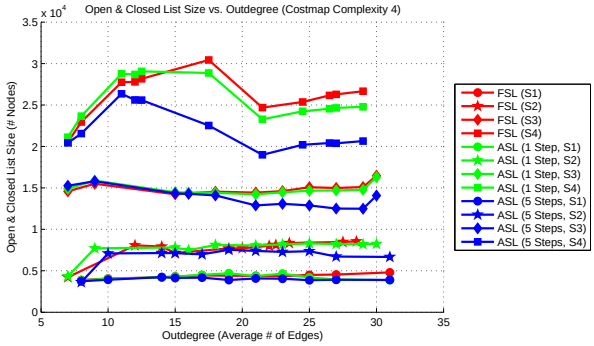
The challenge with the previously described approach is that the action parameterization must be determined beforehand. In situations where the initial action parameterization is too expressive, many locally optimal action parameters can be found that are far from the global optimum. Conversely, the continuum solution may be able to be represented if the action parameterization is too restrictive. Combining graph search with local chassis optimization is a potentially better method for determining trajectories where the action parameterization is poorly defined or when the solution space contains numerous local optima.

EXPERIMENTAL DESIGN

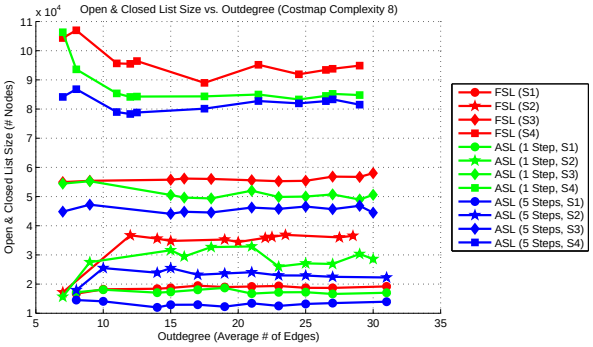
To test another application of the adaptive state lattice, a control set was designed for a simulated mobile robot based on the Scarab planetary rover (Bartlett et al., 2007). Nodes in the control represented two-dimensional position, orientation, and the left and right chassis configuration angles.



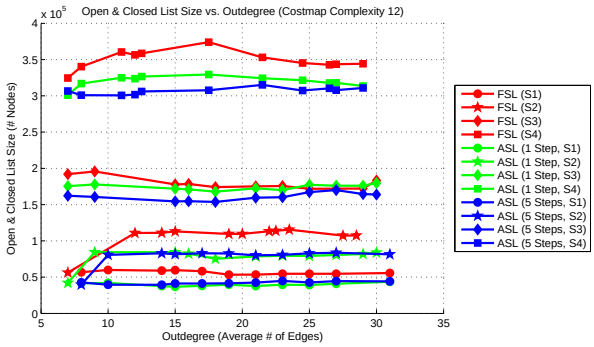
(a) fixed and adaptive state lattice comparison for costmap "0"



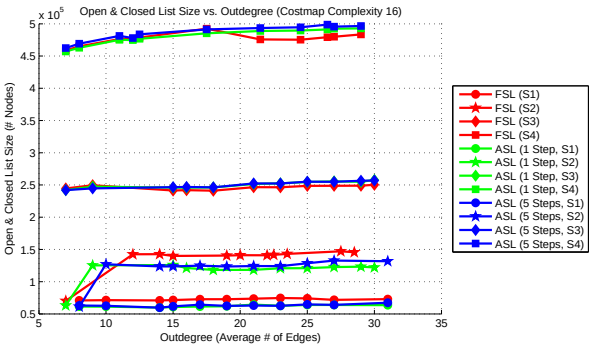
(b) fixed and adaptive state lattice comparison for costmap "4"



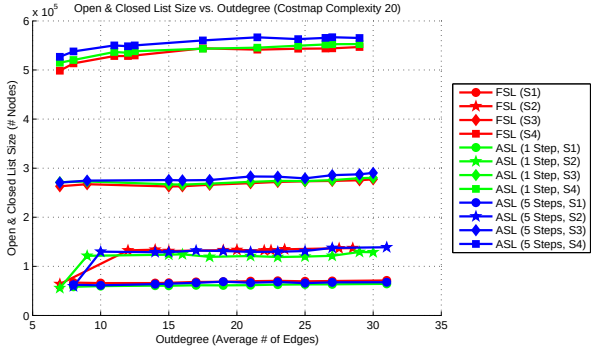
(c) fixed and adaptive state lattice comparison for costmap "8"



(d) fixed and adaptive state lattice comparison for costmap "12"



(e) fixed and adaptive state lattice comparison for costmap "16"



(f) fixed and adaptive state lattice comparison for costmap "20"

Figure 5.15: Memory versus outdegree for fixed and adaptive state lattices over varying risk densities. As expected, memory usage increases roughly exponentially with sampling density.

$$(x, y, \psi, \vartheta_{left}, \vartheta_{right}) \in SE(2) \times [0, 2\pi] \times [0, 2\pi] \tag{5.3}$$

Connectivity in such a graph can be provided by the previously described model-predictive trajectory generation technique, since it can determine feasible actions that efficiently connect boundary states in rough terrain. The elevation (z) and attitude (ϕ,θ), and passive chassis angle (β) are determined using a suspension model based on minimizing the distance between the terrain and wheel contact points. A cost function related to platform stability (Furlong et al., 2009) is used to evaluate the risk associated with states along each control set edge:

$$\mathcal{L}(\mathbf{x}(t), \mathbf{u}(t), t) = 1 + w_\phi \phi^2 \tag{5.4}$$

Only the chassis configuration freedoms (ϑ_{left},ϑ_{right}) were allowed to relax in this experiment to isolate the benefits of exploiting the actively articulating chassis. Both the fixed and adaptive chassis state lattices used the same initial control set, which only considered a nominal configuration for the actively reconfigurable chassis ($\vartheta_{left} = \vartheta_{right} = \frac{\pi}{4}$). The graph search used a cost function squared roll weight (w_ϕ) of 10.

RESULTS

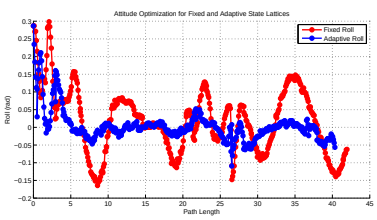
Figure 5.16 shows the results of the experiment with fixed and adaptive state lattices with chassis optimization. The roll experienced by the two resulting motion plans is depicted in 5.16a. The solution produced by the adaptive lattice plan reduces the overall roll experienced by the vehicle and produces a shorter path. Figures 5.16b and 5.16c show the histories of the left and right chassis angle commands as a function of path distance. In order to use the model-predictive trajectory generation technique from Chapter 4, the actions would have to be parameterized with many degrees of freedom and initialized in the region of convergence of this solution.

Views of the paths generated for the fixed and adaptive search spaces are shown in Figures 5.16d through 5.16g. The adaptive state lattice searches fewer nodes (3979 control set expansions) than the fixed state lattice (8268 control set expansions) as a more direct, optimal path is found. The adaptive and fixed state lattices produced a solution with a respective path costs of 41.5 and 45.7 and runtimes of 45.7sec and 114.1sec.

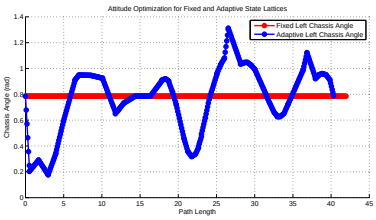
There are two important points in the results of this experiment. First, the adaptive state lattice with chassis optimization reduces the overall roll experienced compared to the original graph. The two motion planning graphs are topologically identical, but the state mapping equation in the adaptive state lattice points to different real-world states. Second, the chassis optimization actually influences edge shapes in the generated informed search space. Simply post-processing the path or applying feedback control when following the path will not produce the same answer as the adaptive state lattice.

5.4 PROGRESSIVELY ADAPTIVE STATE LATTICES

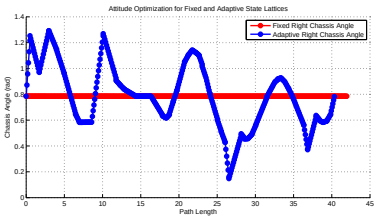
There are two situations where the adaptive state lattice technique may not be the most effective solution for mobile robot motion planning search space generation. First, the increased runtime of the technique in relatively open en-



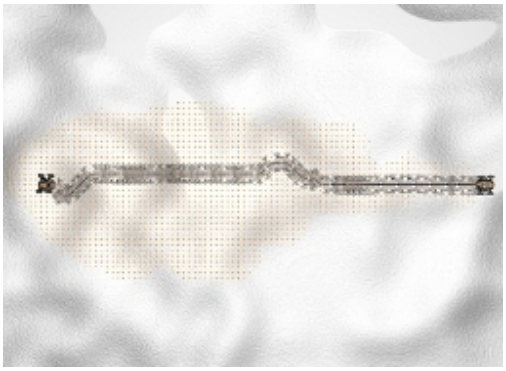
(a) roll vs distance traveled



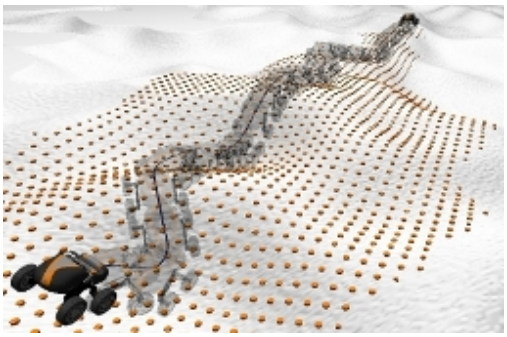
(b) left chassis angle command vs distance traveled



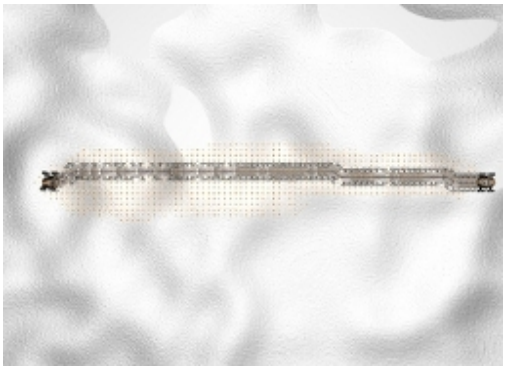
(c) right chassis angle command vs distance traveled



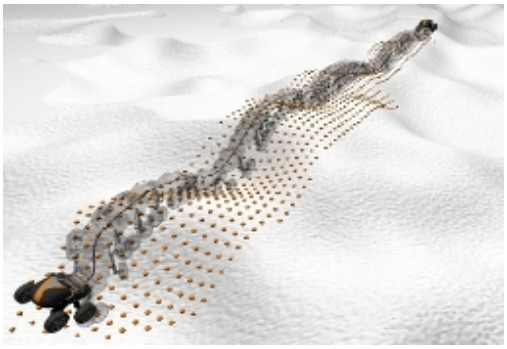
(d) overhead view of trajectory generated by fixed state lattice in the dimensions of chassis configuration angle



(e) perspective view of trajectory generated by fixed state lattice in the dimensions of chassis configuration angle



(f) overhead view of trajectory generated by adaptive state lattice that relaxes in the dimensions of chassis configuration angle



(g) perspective view of trajectory generated by adaptive state lattice in the dimensions of chassis configuration angle

Figure 5.16: Comparison of generated trajectories with fixed and adaptive state lattices that penalize integrated attitude and distance. Figures 5.16a, 5.16b, and 5.16c shows the roll, left chassis angle command, and right chassis angle command as a function of path distance respectively for the fixed and adaptive state configuration optimization lattices. In this example, the fixed state lattice cannot deviate from the standard chassis configuration while the adaptive state lattice locally relaxes in those dimensions of the graph. The adaptive state lattice produces an answer 9.2% cheaper than the fixed state lattice while searching 51.8% fewer nodes.

PASL-A*($\mathbf{x_I}$,$\mathbf{x_T}$)

INPUT: $\mathbf{x_I}$ (initial state), $\mathbf{x_T}(\mathbf{s})$ (target terminal state)
OUTPUT: $\mathbf{x}(t)$ (trajectory), $\mathbf{u}(\mathbf{p},\mathbf{x},t)$ (parameterized action sequence)
DATA: $\mathbf{U}(\mathbf{s_p},\mathbf{s_c})$ (control set), $\mathbf{f_{SME}}$ (state mapping equation), $\mathbf{U_{PREV}}(\mathbf{s_p},\mathbf{s_c})$ (previously adapted control sets)

1 $\mathbf{s_{START}} \leftarrow \mathbf{f_{SME}}(\mathbf{x_I})$, $\mathbf{s_{GOAL}} \leftarrow \mathbf{f_{SME}}(\mathbf{x_T})$
2 OPEN $\leftarrow \emptyset$, CLOSED $\leftarrow \emptyset$
3 insert $\mathbf{s_{START}}$ into OPEN
4 **while** OPEN $\neq \emptyset$ **do**
5 remove $\mathbf{s_{TOP}}$ with minimum f from OPEN
6 **if** $\mathbf{s_{TOP}} = \mathbf{s_{GOAL}}$ **then**
7 **return** $\mathbf{s_{TOP}}$
end
8 **foreach** $\mathbf{s_c} \in \mathbf{S_c}(\mathbf{s_{TOP}})$ **do**
9 **if** $\mathbf{s_c} \in \mathbf{U_{PREV}}(\mathbf{s_p},\mathbf{s_c})$ **then**
10 $\mathbf{s_c} \leftarrow \mathbf{U_{PREV}}(\mathbf{s_p},\mathbf{s_c})$
end
11 STATEMAPPINGEQUATIONOPTIMIZATION($\mathbf{s_c}$,$\mathbf{S_c}(\mathbf{s_c})$,$\mathbf{U}(\mathbf{s_c},\mathbf{S_c}(\mathbf{s_c}))$,$\mathbf{f_{SME}}(s)$)
12 $\mathbf{U_{PREV}}(\mathbf{s_p},\mathbf{s_c}) \leftarrow \mathbf{s_c}$
end
13 EXPANDCONTROLSET($\mathbf{s_{TOP}}$,$\mathbf{U}(\mathbf{s_{TOP}},\mathbf{s_c})$)
14 insert $\mathbf{s_{TOP}}$ into CLOSED
end

Figure 5.17: A* search in a progressively adaptive state lattice (PASL-A*).

vironments may exceed the desired replanning requirements of the system. The second related situation occurs in dynamic environments. In operations where the system receives high-rate perceptual information updates or when a vehicle must react quickly to maintain safely a faster replanning rate could be more desirable. The results presented in the previous section demonstrated the performance improvement of a single-step of the adaptive state lattice could dramatically improve the quality of solutions generated.

This section proposes the Progressively Adaptive State Lattice (PASL) algorithm, which utilizes the local state transition equations from the previous motion plan to seed the optimization of the next cycle. Successive motion plans iteratively improve the quality of the search space while maintaining identical topologies, resulting in a constantly refining path that can quickly react to recently observed perceptual information. An overview of the progressively adaptive state lattice combined with a A* graph search is shown in Figure 5.17.

While very similar to the adaptive state lattice, one important difference occurs between lines 7 and 10. First, every expanded node that would normally be subject to control set adaption is first checked against a data structure to see if it has been optimized in a previous motion plan. After normal control set adaptation, the newly modified child nodes are entered or replaced in the previously expanded control set data structure. As successive motion planning cycles

expand the same region of nodes several times, several cycles of the progressively adaptive state lattice algorithm with single-step control set adaptation has roughly the same effect as a single cycle of an adaptive state lattice with a multi-step control set adaptation. Similar to the concept of anytime search algorithms (Ferguson and Stentz, 2005), progressively adapting the state lattice is beneficial when a faster replanning rate is desired and originally sub-optimal motion plans are acceptable.

5.4.1 EXPERIMENTS

Two experiments are designed to demonstrate the benefits of the progressively adaptive state lattices. First, a motion plan is iteratively refined from a constant initial state to show how the search space adapts to the environment over time. Second, a vehicle follows a portion of the reference trajectory before replanning. This scenario better represents vehicle implementation as the mobile robot will execute suboptimal actions initially and will iteratively refine the solution.

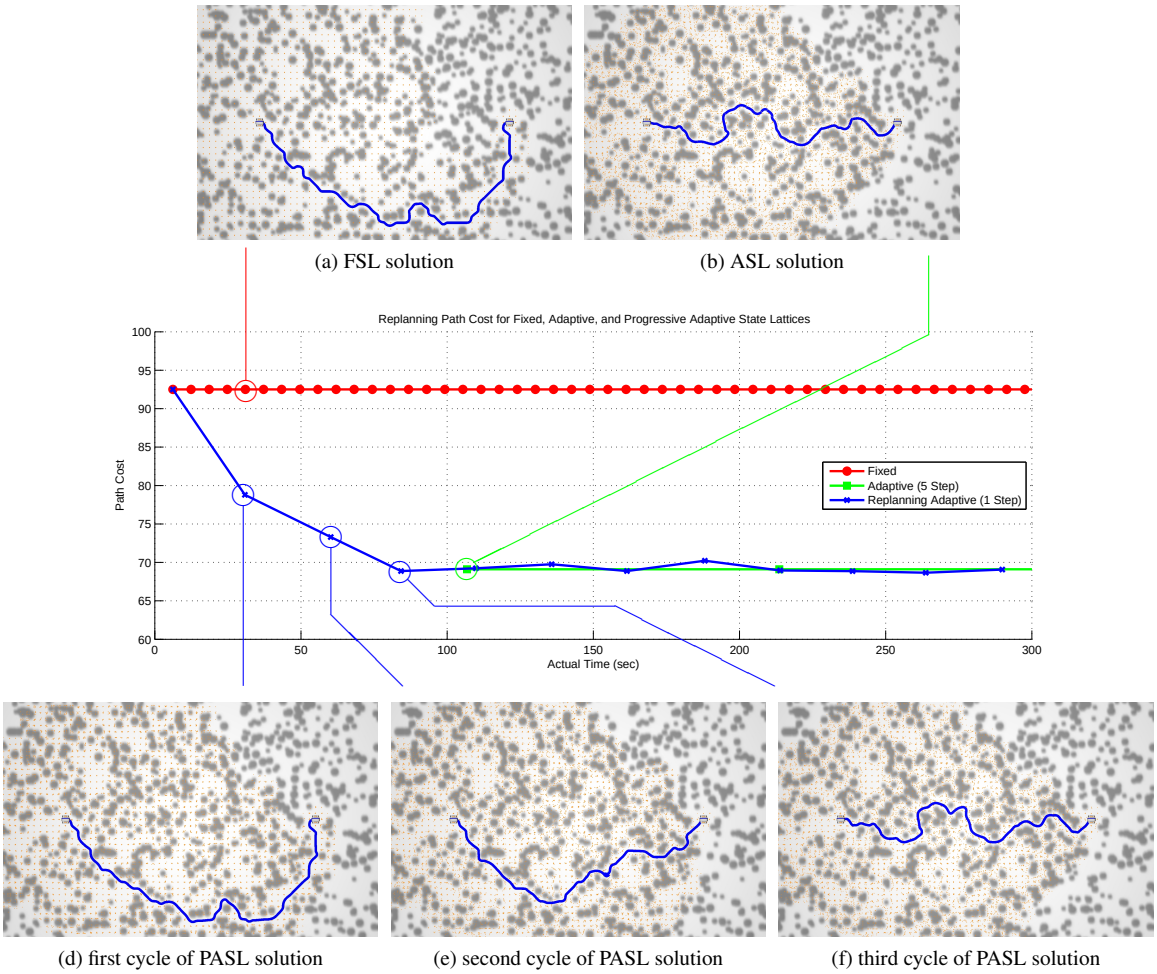
ITERATIVELY IMPROVED MOTION PLANS FROM A CONSTANT INITIAL STATE

To demonstrate the iterative improvement capability of the progressively adaptive state lattice technique, a motion plan is generated as fast as one can be produced by a fixed state lattice, a multi-step adaptive state lattice, and a progressively adaptive state lattice. This particular progressively adaptive state lattice example executes a single step of the search space optimization while exploiting the state mapping equation optimization from previous planning cycles.

Figure 5.18 shows the quality of the resulting motion plans as a function of time for the three different techniques. The fixed state lattice exhibits a significantly faster update rate but produces a higher cost solution than the adaptive state lattice variations. The multi-step adaptive state lattice generates a high quality solution but takes the longest to produce the trajectory through the obstacle field. The progressively adaptive state lattice represents a comprimise between the two techniques, replanning at a faster rate than the adaptive state lattice but eventually producing a solution of the same quality as the multi-step adaptive state lattice. In this example, the progressively adaptive state lattice produces a path whose cost is near the multi-step adaptive state lattice after just three replanning cycles.

Another interesting feature of the example shown in Figure 5.18 is that the update rate of the progressively adaptive state lattice increases as a function of time. After state mapping equations for nodes in the progressively adaptive state lattice reach their locally optimal solution, optimization is not performed on these nodes, thereby increasing the speed at which nodes are expanded in the search space.

The utility of iteratively improving the quality of the motion plan for a real-world application is in the event that the original fixed state lattice plan is too risky to traverse. Rather than the autonomy system calling for an intervention, the progressively adaptive state lattice can operate for a few cycles to iteratively improve the quality of the solution. If the progressively adaptive state lattice would have determined a solution of an acceptable quality, it will have prevented an intervention due to failure of the motion planning system to determine a quality path through the environment.



(a) FSL solution (b) ASL solution

(d) first cycle of PASL solution (e) second cycle of PASL solution (f) third cycle of PASL solution

Figure 5.18: Motion plan quality as a function of time for a fixed state lattice, a multi-step adaptive state lattice, and a progressively adaptive state lattice. The fixed state lattice produces an answer quickly, but cannot improve over time. Conversely, the multi-step adaptive state lattice takes a long time to produce a near-continuum solution. The progressively adaptive state lattice improves the quality of the resulting motion plan iteratively at a faster update rate than the multi-step adaptive state lattice by exploiting the state mapping equation optimization for all nodes from the previous search.

ITERATIVELY IMPROVE MOTION PLANS FROM A VARIED INITIAL STATE

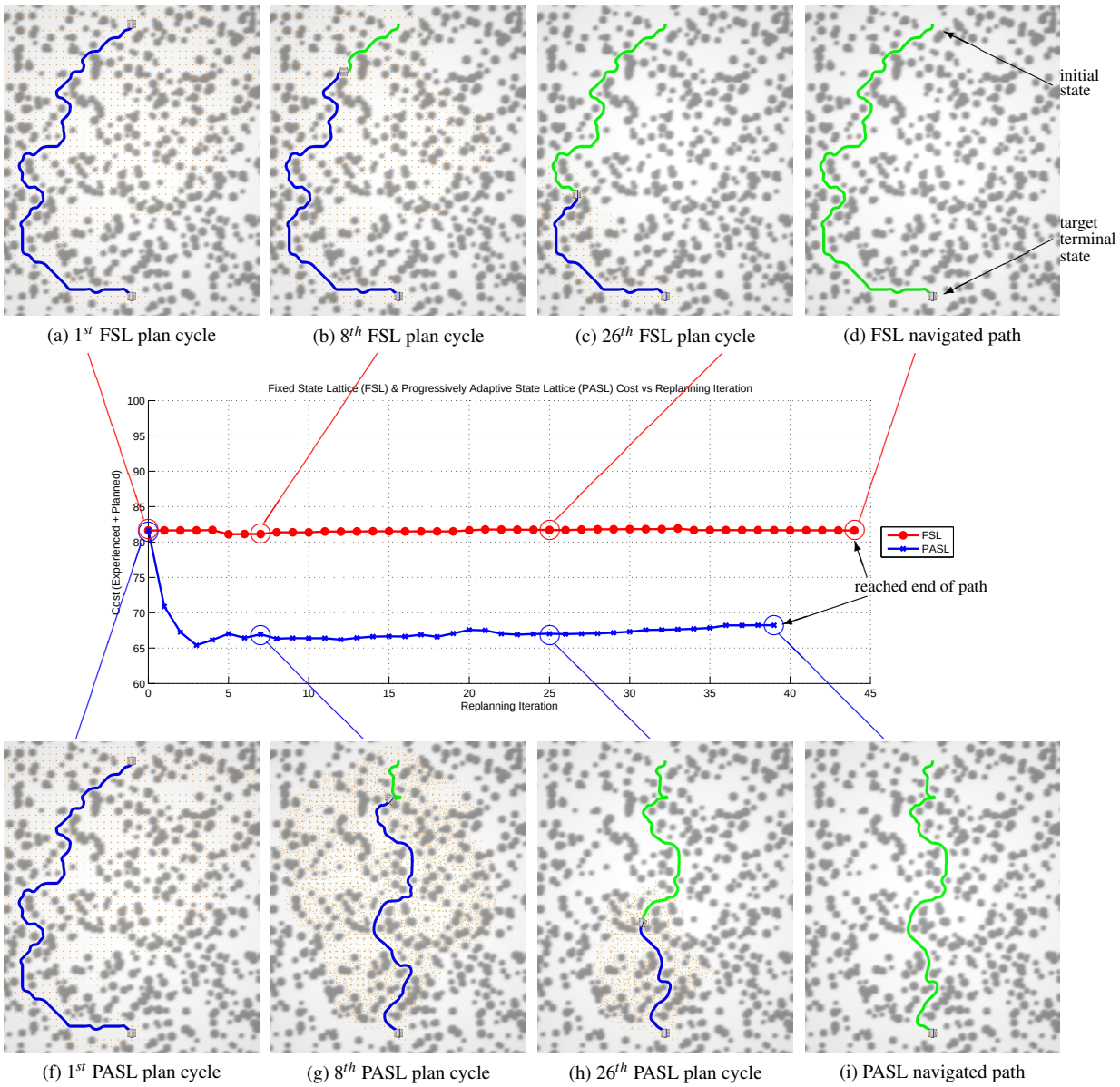
A second experiment with progressively adaptive state lattice involves exploiting iteratively improved motion plans from a varying initial state. Whereas the previous experiment showed that the quality of the motion plan can improve from a static vehicle configuration, this test involves showing how the search space can be iteratively improved while the vehicle is in motion. In this experiment, a vehicle traverses the next edge from the generated motion plan before it determines a new trajectory through the environment. The first cycle of the progressively adaptive state lattice uses the fixed state lattice solution in order to determine a reasonable first action quickly.

Figure 5.19 shows the result of planning and executing a motion plan using a fixed state lattice and a progressively adaptive state lattice. Initially, both motion plans follow the motion plan produced by the original (unoptimized) state lattice. During the first replanning cycle of the progressively adaptive state lattice, local state mapping equation optimization reorganizes the edges to produce a lower cost solution than the fixed state lattice. The graph shows plots of the predicted cost of traversal as the sum of the experienced cost and the predicted cost to reach the goal over replanning cycles. In this example, the mobile robot following the trajectory produced by the progressively adaptive state experienced 16.43% lower path cost (68.22 vs 81.63) than the one following the fixed state lattice trajectory, despite the topologically identical state lattices.

5.5 SUMMARY

State lattices are useful structures for mobile robot motion planning because they are feasible by design and sample a controllable distribution of state space. Informed state lattices take this one step further by correcting for the model assumptions in the control set construction using the model-predictive trajectory generation technique presented in Chapter 4. Efficiency is achieved by searching in the low-dimensional parameter space of actions in the model-predictive trajectory generator. Regenerating the connectivity when provided this new information enables motion planners to generate more informed motion plans that more accurately represent the true cost of executed motions.

Adaptive state lattices demonstrated that a search space can be optimized without knowledge of the terminal state and without modifying the topological structure of the motion planning graph. The process locally optimizes the state mapping equation for a particular node in the graph by minimizing the aggregate cost of all connected edges. This enables the search space to relax around obstacle and high-risk regions to achieve locally optimal connectivity in the motion planning graph. While the resulting motion plans are more computationally expensive on a per-expansion basis, simulation experiments showed that the adaptive state lattice was more efficient than searching denser state lattices in complex environments, producing both better answers in a fixed amount of time and faster solutions for a given level of desired optimality. An extension related to configuration optimization for a planetary rover with an articulating chassis is shown, where a five-dimensional motion planning graph is searched with locally optimized chassis configuration angles in the state lattice. The adaptive state lattice variation for articulation configuration produced a solution with lower overall exposure to attitude and risk in a rough terrain environment. Relaxation as a post-processing step is ineffective for generating near-globally optimal motion plans in complex environments. By relaxing during search using adaptive state lattice techniques, generated solutions can transcend homotopy classes in



(a) 1^{st} FSL plan cycle (b) 8^{th} FSL plan cycle (c) 26^{th} FSL plan cycle (d) FSL navigated path

(f) 1^{st} PASL plan cycle (g) 8^{th} PASL plan cycle (h) 26^{th} PASL plan cycle (i) PASL navigated path

Figure 5.19: Motion plan quality as a function of time for a fixed and progressively adaptive state lattice with varying initial state. Simulations of two mobile robots are shown, where two platforms follow a fixed state lattice and a progressively adaptive state lattice respectively. After several replanning cycles, the progressively adaptive state lattice converges to a local optimum, providing better guidance through the cluttered obstacle field.

the environment, nearing the globally optimal solution.

The progressively adaptive state lattice method iteratively improves the quality of the search space by exploiting the state mapping equation optimization information from the previous cycle. Analogous to the concept of anytime search (Ferguson and Stentz, 2005), this technique progressively optimizes the local state mapping equations of nodes in the search space to produce a lower cost solution. This technique provides the benefits of the adaptive state lattice with a substantially faster update rate as only a single state mapping equation cycle is executed at each expanded node.

Figure 5.20 summarizes some of the advantages and disadvantages of the proposed adaptive state lattice approach. Coarse fixed state lattices typically search quickly but can provide sub-optimal solutions. Dense fixed state lattices provide better answers, but blindly adding actions and states into the search space without regard for the environment slows down search and can significantly increase the memory footprint of the motion planning on the system. Adaptive state lattice techniques using coarse and dense state lattice representations were shown to produce the best solutions in complex environments, although they are more expensive on a per-control set expansion basis. For many applications, a coarse, adaptive state lattice may provide the best balance between optimality, runtime, and memory footprint.

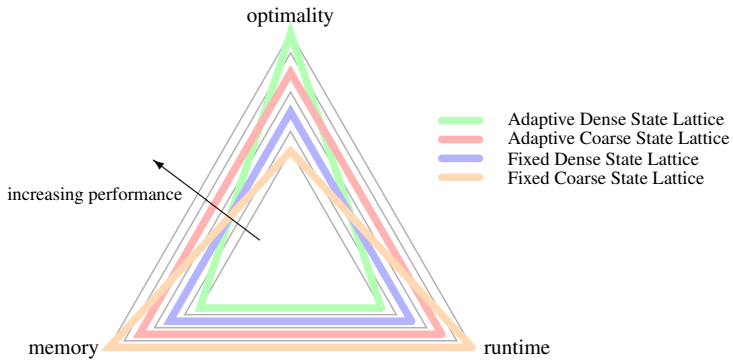


Figure 5.20: A qualitative comparison of the performance of the adaptive and fixed state lattices in complex environments. Two trends emerge from this representation of the experiment from Section 5.3. First, adaptive state lattice techniques produce higher quality solutions compared to fixed state lattices of the same discretization despite being topologically identical. Second, coarse search space representations provide solutions faster than dense search spaces while consuming significantly less resources. The most balanced solution for many mobile robot applications that value optimality, speed, and computational resources would be an adaptive coarse representation of the path continuum.

CHAPTER 6

INTRICATE PATH NAVIGATION AND CONTROL

The ability to generate intricate paths through complex environments cannot be fully exploited unless there is a capacity to follow them. Consider the problem shown in Figure 6.1. In this example the vehicle attempts to follow a reference trajectory produced by a regional motion planner subject to a large path disturbance. The mobile robot navigation problem is the continuum search for an action that reacquires the reference trajectory while minimizing risk to the platform under computational and temporal constraints.

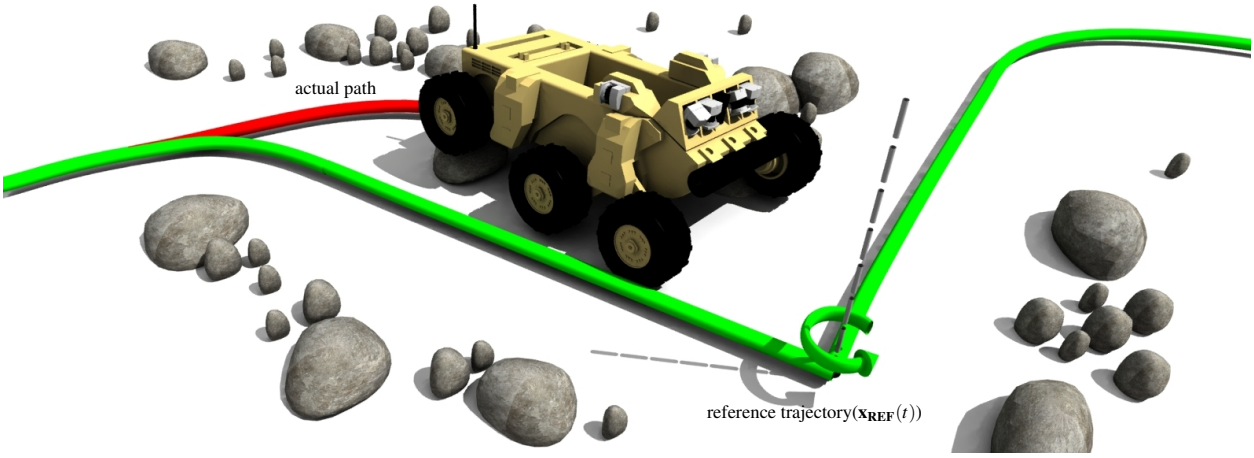


Figure 6.1: A difficult problem in path tracking, navigation, and control. Here, a vehicle is attempting to track a trajectory with geometric singularities subject to a significant path disturbances.

The temporal constraints are related to the rate at which a mobile robot must react to new information to maintain safety. The magnitude of the constraint is related to the vehicle's speed, the environment's obstacle density, and the platform's perceptual horizon. A minimum replanning rate is typically defined for a platform, where a mobile robot navigator must return an action. Since computational constraints limit the number of actions that a vehicle can evaluate, a mobile robot navigator must determine methods that efficiently sample the action space.

As discussed in Chapter 3, typical approaches to reference trajectory tracking and obstacle avoidance involve sampling in the path space, the action space, and the state space. Path space sampling techniques generate an action based on reacquiring a forward point on the path. Action space sampling methods search a set of actions determined

by sampling in the space of low-order inputs. State-space sampling techniques also search an action set based on uniform or biased sampling in the local state space of the vehicle, but those actions are defined only by the boundary state constraints.

Consider shown in Figure 6.2. Pure pursuit (Figure 6.2a) generates an action that reacquires a forward point on the path, cutting the cusp and pointing the vehicle in the wrong direction. Input-space sampling techniques (Figure 6.2b) can not accurately predict the actual path of the vehicle unless it samples in the space of arcs, splines, and turn-in-place actions sequences. Likewise, the state-space sampling solution (Figure 6.2c), previously described in Section 4.1 will not produce actions similar to the reference trajectory unless they are guided by the structure of the reference trajectory. A commonly applied technique is to reduce the action horizon to the point of the geometric singularity or discontinuity (Figures 6.2d through 6.2f). The drawback of this approach is that the period for which vehicle safety is evaluated is reduced.

Whereas all of these techniques exploit at most states from the reference trajectory, a potentially better technique is to exploit the actions used to determine a local portion of the reference trajectory to parameterize the action space. This chapter describes a receding horizon model-predictive control (RHMPC) technique for model-predictive navigation and control of mobile robot systems.

6.1 Receding Horizon Model-Predictive Control

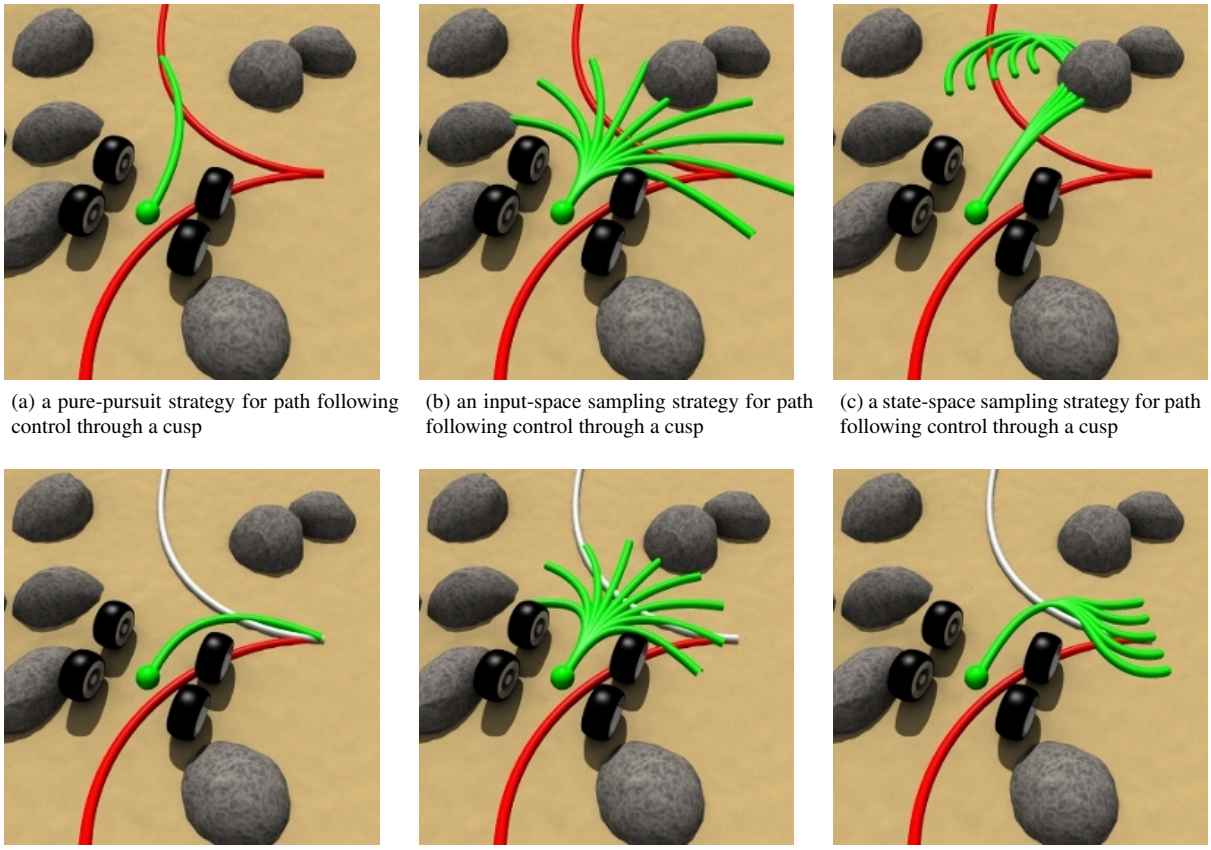
The mobile robot navigation problem is formulated as an optimal control problem. Just as with unconstrained optimization model-predictive trajectory generation, the problem becomes determining the actions that minimize a utility functional subject to a set of constraints:

$$\begin{aligned} \text{minimize:} \quad & J(\mathbf{x},\mathbf{u},t) \\ \text{subject to:} \quad & \dot{\mathbf{x}} = \mathbf{f}_{\mathbf{PMM}}(\mathbf{x},\mathbf{u},t) \\ & \mathbf{x}(t_I) = \mathbf{x}_{\mathbf{I}} \\ & \mathbf{u}(\mathbf{x},t) \in \mathbf{U}(\mathbf{x},t), \quad t \in [t_I, t_F] \end{aligned} \tag{6.1}$$

The difference between this formulation and unconstrained optimization model-predictive trajectory generation is the constraint that the actions must be defined for some period of time. It is necessary to plan far enough ahead to reduce path following error and ensure that the vehicle can safely maneuver around obstacles. The initial state constraint and the predictive motion model satisfaction require that the action is feasible in the formulation. The challenge becomes determining the space of actions to search and the construction of the optimization utility function minimized.

6.1.1 Action Parameterization

Reducing the continuum of actions into a manageable search space is one of the most difficult problems in mobile robot navigation. This is further complicated when the reference trajectory is composed of complex primitives including



(a) a pure-pursuit strategy for path following control through a cusp

(b) an input-space sampling strategy for path following control through a cusp

(c) a state-space sampling strategy for path following control through a cusp

(d) a pure-pursuit strategy for path following control through a cusp

(e) an input-space sampling strategy for path following control through a cusp

(f) a state-space sampling strategy for path following control through a cusp

Figure 6.2: Alternative path following controller techniques maneuvering through a cusp. Each of the three widely applied path following control techniques fail to generate a path that accurately follows the reference trajectory without sacrificing lookahead horizon.

turn-in-place and cusp actions that result in paths with geometric singularities and discontinuities. Whereas path position and orientation may be discontinuous at these points, the states and actions in the reference trajectory define a perfectly feasible motion. A requirement of this approach is that the sequence of actions that define the reference trajectory must be known in addition to the states along the path. Since many current motion planning algorithms are built with knowledge of the inputs necessary to execute the commands this requirement is not as strict as it appears. Formally, the reference trajectory is defined as a set of states ($\mathbf{x_n}(t)$) and actions ($\mathbf{u}(\mathbf{p_n}, \mathbf{x}, t)$):

$$\mathbf{x_{REF}}(t) = \begin{bmatrix} \mathbf{u}(\mathbf{p_0},\mathbf{x},t) & \mathbf{u}(\mathbf{p_1},\mathbf{x},t) & \dots & \mathbf{u}(\mathbf{p_n},\mathbf{x},t) \\ \mathbf{x_0}(t) & \mathbf{x_1}(t) & \dots & \mathbf{x_n}(t) \end{bmatrix}^T \tag{6.2}$$

The receding horizon model-predictive control technique parameterizes the action space of the navigator by utilizing the structure of the local reference trajectory. Exploiting the parameterization of the reference trajectory enables the controller to better follow and match the shape of the reference trajectory without resorting to sampling in the action space. Input space sampling is inefficient if not ineffective for intricate paths because of the dimensionality of the action space necessary to search.

An overview of the receding horizon model-predictive control technique is shown in Figure 6.3. In this scenario, a mobile robot initially driving in reverse attempts to track a reference trajectory through a cusp. The mobile robot is currently off the path and needs to generate an action that will execute the desired cusp and maintain vehicle safety. After the closest state on the reference trajectory is determined (Figure 6.3a), the reference trajectory is divided into the individual trajectories ($\mathbf{x_n}(t)$) and parameterized actions ($\mathbf{u}(\mathbf{p_n},\mathbf{x},t)$) (Figure 6.3b). Next, the reference trajectory is searched to determine a sequence where the constraint on the action horizon ($[t_I,t_F]$) is satisfied (Figure 6.3c).

The initial guess for the parameterized actions ($\mathbf{u_{RHMPC}}$) is defined by the sequence of parameterized actions between the closest point on the path and the predefined control horizon (Figure 6.3d). In this example, the free parameters of the receding horizon model-predictive controller ($\mathbf{p_{RHMPC}}$) are defined by a concatenation of the free parameters in the control inputs:

$$\mathbf{p_{RHMPC}} = \begin{bmatrix} \mathbf{p_2}^T & \mathbf{p_3}^T & \mathbf{p_4}^T \end{bmatrix}^T \tag{6.3}$$

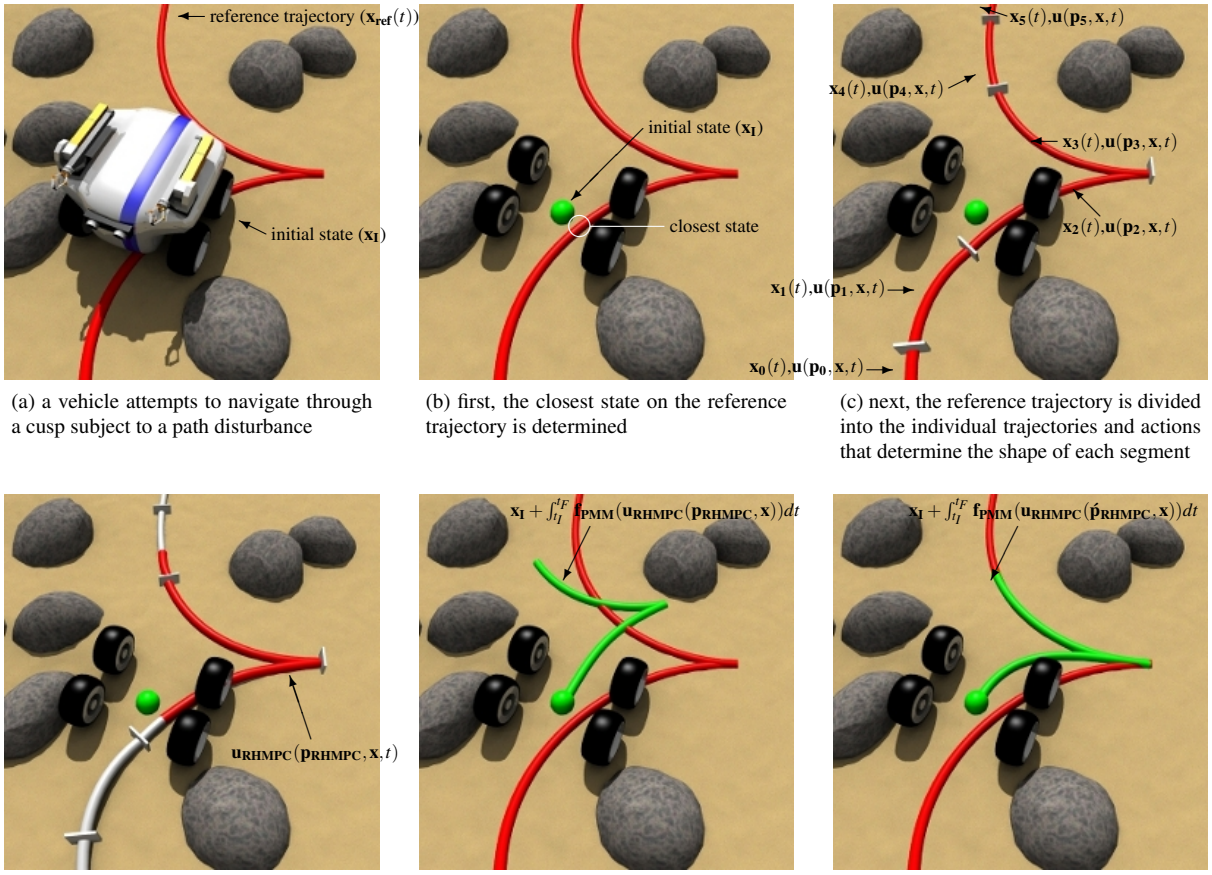
The action defined by parameters $\mathbf{p_2}$ define a reverse motion while the parameters $\mathbf{p_3}$ and $\mathbf{p_4}$ represent forward actions. Treating the sequence as a single parameterized action enables it to be handled just like any other in unconstrained optimization model-predictive trajectory generation.

6.1.2 Path Deviation Optimal Control

Once the action parameterization has been determined, the next step is to modify the parameters to compensate for disturbances, approximations, and errors in the motion model. This technique seeks to minimize a cost function ($\mathbf{J}(\mathbf{x},\mathbf{u},t)$) by modifying a set of actions:

$$J(\mathbf{x},\mathbf{u},t) = \int_{t_I}^{t_F} \mathcal{L}_{CROSSTRACK}(\mathbf{x}(t),t),t)dt \tag{6.4}$$

In order to follow a trajectory, a cost function is designed to penalize the area between the resulting action and the states in the reference trajectory. An initial penalty is determined by forward simulating the actions determined from the reference trajectory segmentation (Figure 6.3e) and the penalty is optimized by perturbing the inputs in the same manner as unconstrained optimization model-predictive trajectory generation (Figure 6.3f). It is important here to note that the receding horizon model-predictive control technique provides a more elegant solution than alternative



(a) a vehicle attempts to navigate through a cusp subject to a path disturbance

(b) first, the closest state on the reference trajectory is determined

(c) next, the reference trajectory is divided into the individual trajectories and actions that determine the shape of each segment

(d) a horizon-limited portion of the reference trajectory is used to parameterized the action for unconstrained optimization model-predictive trajectory generation

(e) the first receding horizon model-predictive control action is evaluated by passing it through the predictive motion model

(f) a corrective receding horizon model-predictive control action is determined by minimize the integrated deviation from the reference trajectory

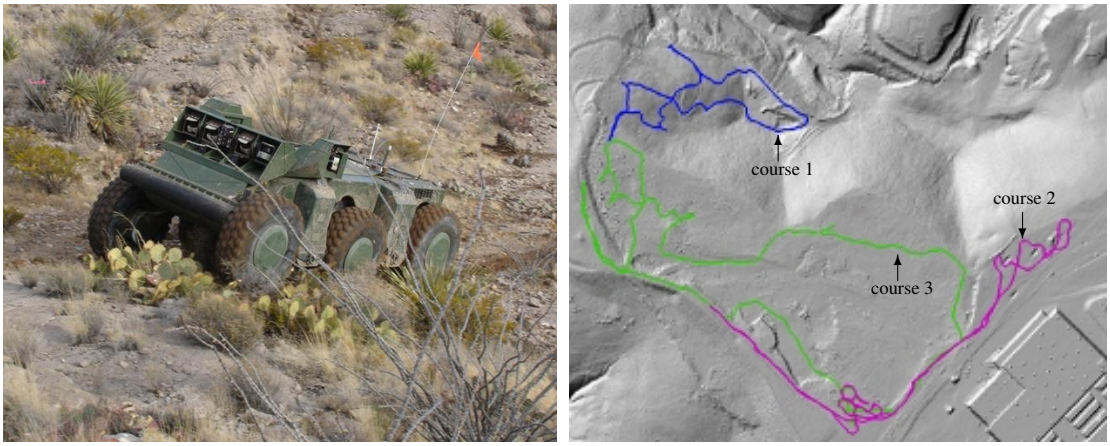
Figure 6.3: The overview of the receding horizon model-predictive controller (RHMPC) is presented. First, the parameterization of the RHMPC primitive is found by selecting a portion of the target path. Then, the action is forward simulated and subsequently optimized from the current state to account for disturbances in the system.

sampling techniques shown in Figure 6.2 without sacrificing the action horizon.

6.2 EXPERIMENTS

6.2.1 INTRICATE PATH FOLLOWING IN COMPLEX ENVIRONMENTS

To test the performance of the receding horizon model-predictive controller in a real-world environment, an experiment was designed and executed that compared the performance of the presented trajectory follower to one that directly executed the original trajectory. The Crusher mobile robot (Figure 6.4a) was used for this field experiment. This mobile robot was particularly suited for this experiment since the six-wheeled skid-steered mobility system is particularly challenging to model in rough terrain. The field experiments, consisting of three multi-kilometer courses (Figure 6.4b) defined by sequences of way-points, was conducted at a test site in Pittsburgh, Pennsylvania with variable off-road terrain.



(a) Crusher, a six-wheeled skid-steered mobile robot

(b) The field experiment courses

Figure 6.4: The mobile robot and test environment for the receding horizon model-predictive controller field experiments. The experiments consisted of three multi-kilometer runs through variable terrain.

IMPLEMENTATION

Each system used the same implementation of A* on a fixed state lattice for the regional motion planner that searched feasible actions designed specifically for Crusher's predictive motion model. The edges in the search space were provided by a control set consisting of forward, reverse, and turn-in-place actions with lengths varying between 3m and 9m. Actions in the search space consisted of constant commanded linear velocities and either second-order splines of curvature command as a function of distance or constant angular velocity commands as a function of heading. This implementation used a 60m $\times$ 60m vehicle centric cost map. Updates to the reference trajectory were provided by the regional motion planner at 2Hz.

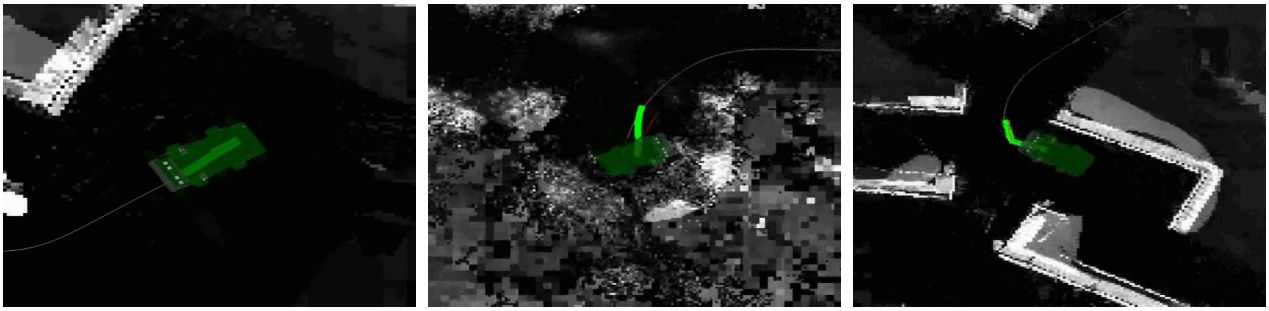
The reference trajectories produced consisted of a sequence of independent profile parameters. This low-bandwidth representation was beneficial because a sequence of expressive actions over tens of meters could be described with a relatively few numbers of parameters.

EXPERIMENTAL DESIGN

Integrated path cost is the main metric used to quantify success for the field experiments, which is related to the risk, mobility, and traversability for a vehicles configuration in the environment. While inherently unit-less and scaleless, it provides the best metric for comparing performance between the two systems because it is the quantity optimized by both the trajectory follower and motion planner.

RESULTS

Figure 6.5 shows several selected examples from the field experiments where the receding horizon model-predictive control was used to navigate intricate paths in varied, natural terrain. Three different situations are shown involving geometric singularities and discontinuities including cusps and turn-in-place actions.



(a) planning a control with constant horizon through a cusp action

(b) planning a control that reasons about a turn-in-place action and compensates for a path following disturbance

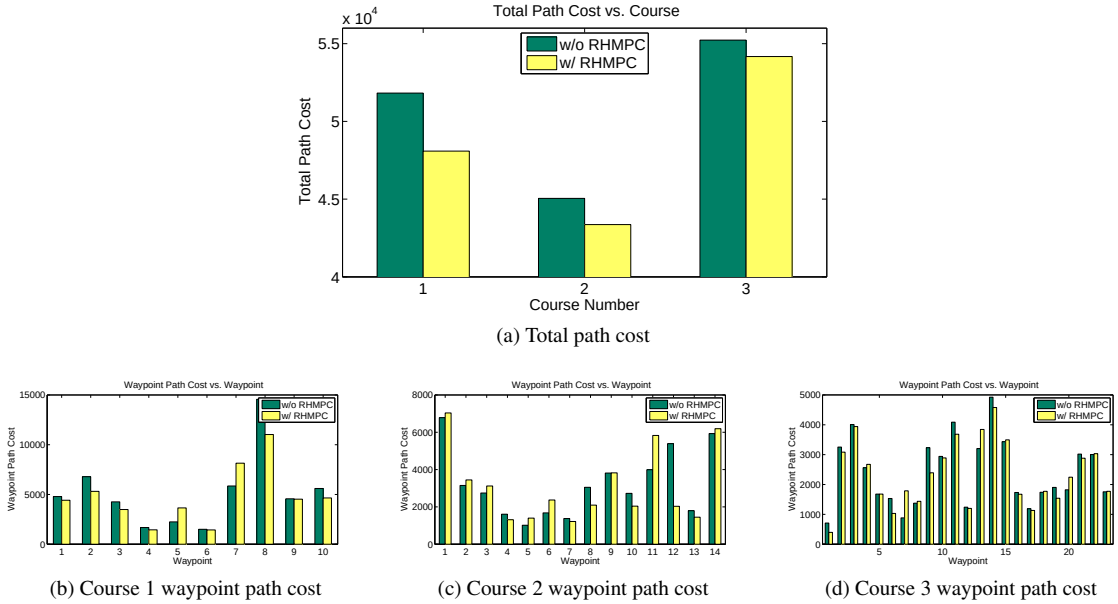
(c) planning a control with a constant horizon through a future turn-in-place action

Figure 6.5: Selected examples of RHMPC from multi-kilometer field experiments.

In Figure 6.5a, a corrective control is determined that maintains a constant horizon through a cusp action in the reference trajectory. The actions were determined by relaxing a sequence of parameterized actions that included forwards and reverse actions up to a distance-based horizon. Figure 6.5b shows the receding horizon model-predictive control determined for following of a trajectory with an initial turn-in-place action with path following disturbance. The current vehicle state is off the reference trajectory and determines an corrective control that extends the length of the angular velocity command and bends the straight segment to reacquire the path in a feasible manner. The last example shown in Figure 6.5c involves planning through a future turn-in-place action between two nominally straight

segments. This examples shows the flexibility of the technique, where the turn-in-place action does not necessarily need to start or end at a specific point in the receding horizon model-predictive control.

Figure 6.6 shows the results from the three field experiment runs comparing the performance of the system using the receding horizon model-predictive controller and the system that applied a regional motion planner-based trajectory follower.



(a) Total path cost

(b) Course 1 waypoint path cost

(c) Course 2 waypoint path cost

(d) Course 3 waypoint path cost

Figure 6.6: The way-point and total path cost for a series of comparison runs on three multi-kilometer courses. The receding horizon model-predictive controller reduced the accumulated cost experienced by the robot.

On average, the system using the receding horizon model-predictive controller outperformed or achieved similar levels of performance than the alternative system. For portions of the course where disturbances relative to the predicted motion are uncommon or the local cost gradients were small very little improvement would be expected. It is important to note that while the trajectory follower did not outperform the alternative system between every way-point, it did improve the overall performance of the system by up to 7.2%. The variability in the results is expected because of the nature of outdoor mobile robot field experiments where any number of small changes can cause the robot to select a significantly different route.

6.3 SUMMARY

This section described a method for navigation of intricate paths in complex environments. The presented receding horizon model-predictive control technique is a navigator because it maintains the ability to deviate from the target

path in the presence of obstacles while nominally acting as a trajectory follower. The main contribution of the receding horizon model-predictive controller is the ability to maintain a constant lookahead, in distance or time, independent of the complexity of the target path. This is particularly important when the reference trajectory is composed of intricate behaviors, such as cusps and turn-in-place actions, where traditional techniques can fail to follow the path. The receding horizon model-predictive controller exploits the structure of the sequential search motion plan to initialize the unconstrained optimization variation of the model-predictive trajectory generator to search in the local continuum for an action which maintains vehicle safety while tracking the target path. Parameterization of the input enables corrective trajectories to be generated efficiently enough for use in real-time path following as demonstrated by the simulation and field experiments.

CHAPTER 7

CONCLUSIONS

This thesis presented several techniques that enable mobile robots to more effectively plan and navigate complex environments. The model-predictive trajectory generator delivered the capability to determine actions that satisfy continuum boundary state constraints subject to arbitrary predictive motion models. Adaptive model-predictive search spaces enabled motion planning search graphs to exploit environmental information. The receding horizon model-predictive controller exploited global guidance information to effectively initialize and parameterize actions that were relaxed to follow paths and avoid obstacles. This section presents short summaries of the three main research topics in the dissertation, reviews the contributions of the thesis, and discusses future directions for the work.

7.1 SUMMARY

MODEL-PREDICTIVE TRAJECTORY GENERATION

The presented model-predictive trajectory generation algorithm has proven to be an effective means for generating parameterized control inputs to generate paths between arbitrary boundary states subject to highly generic mobility models. Formulating the problem as parametric optimal control enables constrained optimization to be used to efficiently generate inputs that satisfy state constraint and/or minimize a utility functional. Implementation of the method requires choice of parameterized control inputs to define the space of inputs, a predictive motion model to approximate vehicle motion, and an initialization function to determine where the parameters in the constrained optimization start from. An architecture for separating the vehicle-dependent and vehicle-independent portions of the algorithm was shown that has been effectively deployed on a number of diverse mobility systems. Several simulation and field experiments showed that trajectory generation can determine solutions where constraint values, dynamic constraints, and environmental information varies significantly. The model-predictive trajectory generation algorithm can also be used to provide the underlying connectivity in a tiered optimization framework where something which marginally effects motion needs to be optimized along the shape of the path. The field experiments demonstrated the effectiveness of the algorithm in generating actions and informed search spaces that are more efficient than alternative input-space sampling techniques.

ADAPTIVE MODEL-PREDICTIVE SEARCH SPACES

The ability to adapt search spaces to the environment is a powerful tool for mobile robots navigating in complex environments. The ability to reason about and modify a search space without changing the topology of a recombinant graph enables efficient search without changing the density of the sampling. The results presented in this section demonstrate that an adaptive state lattice with lower sampling can generate motion plans that exhibit higher levels of completeness and optimality than some graphs with denser sampling. It was also shown that in these types of environments, relaxation as a post-processing step was ineffective at generating a path plan that could compete with the adaptive state lattice because it did not exist in the radius of convergence to the global minimum. Relaxation could still be applied to the motion plan generated by the adaptive state lattice, but integration of the relaxation process into the search space enables the generated motion plan to pass over boundaries that define homotopically distinct trajectories. The progressively adaptive state lattice is a continual optimization of a search space that provides a faster update rate by trading off initial path sub-optimality.

RECEDING HORIZON MODEL-PREDICTIVE CONTROL

The receding horizon model-predictive controller was developed to navigate intricate paths generated by motion planners in complex environments. The novelty in the technique lies in exploiting the structure of the regional motion plan by parameterizing the action space from a portion of the reference trajectory. The implementation of the technique is a minor extension of the unconstrained optimization model-predictive trajectory generation technique, where actions are actually structured as sequences of lower-order primitives. Since no partial derivatives are determined analytically, the optimization process can naturally handle sequences of actions with varying independent variables, including time, distance, or heading. This generality enables the technique to handle trajectory following with geometric singularities (turn-in-places) and discontinuities (cusps) naturally without relying on other operational modes or special case handlers.

7.2 CONTRIBUTIONS

This thesis makes original contributions in the areas of trajectory generation, motion planning, navigation and control for mobile robots operating in complex environments.

- **DEVELOPMENT OF THE FIRST ADAPTIVE MODEL-PREDICTIVE RECOMBINANT SEARCH SPACE.** The main contribution is the development of the first adaptable model-predictive recombinant search space. Adaptability comes in two forms in this work. First, this approach utilizes the model-predictive trajectory generation algorithm to regenerates edges that adapt to changes in mobility and new environmental information. Second, the algorithm locally optimizes the mapping between states and nodes in the search space to move around obstacles and high cost regions. The adaptive model-predictive search space technique improves completeness and optimality in motion planning search spaces by exploiting environmental cues to generate an efficient, informed

search space. This was shown to competently navigate challenging obstacle fields more efficiently than denser, more expressive search spaces.

- **DEVELOPMENT OF A GENERAL, REAL-TIME MODEL-PREDICTIVE TRAJECTORY GENERATION TECHNIQUE.** The model-predictive trajectory generation technique transforms optimal control into parameter search for actions that satisfy boundary state constraints and/or a minimize a utility function. The technique specifically shows how root finding and optimization, numerical linearization, and predictive motion models can be used to effectively and efficiently determine actions. Advances in solving this problem have enabled some of the other contributions mentioned here.

- **DEMONSTRATED APPLICATION AND MERITS OF MODEL-PREDICTIVE TRAJECTORY GENERATION ON A VARIETY OF MOBILE ROBOT PLATFORMS.** The generality of the model-predictive trajectory generation algorithm has been demonstrated on a number of diverse mobile robot systems. It has been used to generate coordinated motion and articulation actions for mobile manipulators and planetary rovers (Anderson et al., 2008; Furlong et al., 2009), place instruments for mobile robots with impaired mobility (Pivtoraiko et al., 2008), effectively navigate urban environments (Ferguson et al., 2008), and guide high-speed mobile robots in rough, outdoor terrain (Howard et al., 2008).

- **A RECEDING HORIZON MODEL-PREDICTIVE CONTROLLER FOR INTRICATE PATH NAVIGATION.** The unconstrained optimization model-predictive trajectory generation technique was combined with an improved way of sampling and parameterizing the action space to determine corrective actions that naturally handle geometric singularities and discontinuities. Experiments demonstrated that the technique effectively follows intricate paths in natural environments and improved the overall system performance compared to a regional motion planning-based navigator.

7.3 FUTURE WORK

There are many future directions for this research. At a high level, many extensions of the this work involves methods to make the contributions run faster (graduated fidelity, replanning, and node synthesis) and produce better search spaces (node decomposition, guided action space sampling, predictive motion model identification)

- **GRADUATED FIDELITY ADAPTIVE STATE LATTICES.** While the presented work demonstrated the benefits of relaxation of the entire state lattice, it did not explore the benefits of restricting the adaptation to higher priority regions of the search space, such as near the start and goal states.

- **INTEGRATING REPLANNING SEARCH ALGORITHMS WITH ADAPTIVE STATE LATTICES.** The next obvious extension to the adaptive state lattice work is to integrate this with a replanning algorithm (such as D*). By processing state transition equation updates just as changes in edge costs, replanning algorithms are a potentially more efficient way to regenerate the solution.

- **NODE SYNTHESIS AND DECOMPOSITION OF ADAPTIVE STATE LATTICES.** As nodes relax into locally optimal configurations, there are situations where multiple nodes share similar locations. Graphs with multiple nodes representing the same configuration are redundant and less efficient, it makes sense to combine control sets to represent a single node location. Similarly, there are situations where splitting control sets into different nodes can provide a more effective motion planning graph. Control sets whose edges represent the limits of feasibility are rigid to movement, despite the benefits provided to other nodes in the control set. Determining techniques which can reason about these limits and decompose the nodes into multiple may provide more effective motion planning graphs where kinematic constraints are significant.

- **GUIDED ACTION SPACE SAMPLING FOR RECEDING HORIZON MODEL-PREDICTIVE CONTROL** The receding horizon model-predictive control technique can be extended by combining the state-space sampling technique (searching in the state space near a horizon state) and parameterizing the input based on the regional motion plan. While the action parameterization technique replaces other initialization functions and may less efficiently estimate the parameters required to satisfy the boundary state constraints, the shape of the resulting search space may be more effective by maintaining a shape closer to the reference trajectory.

- **PREDICTIVE MOTION MODEL IDENTIFICATION.** The ability for all aspects of this work to exploit a highly general predictive motion model enables the application of system identification algorithms. For example, the receding horizon model-predictive controller can use a higher-fidelity, more accurate model to compensate for approximations in the regional motion planner. Alternatively, the modified predictive motion model could be integrated into an informed search space simply by regenerating the control set actions.

This thesis culminates at an exciting time for field robotics. Just in the past few years robots have autonomously traversed hundreds of miles across deserts, safely navigated urban environments, explored subterranean and underwater realms, and performed in-situ studies of the geologic history of Mars. In parallel to other work in systems, computing, perception, and learning, future advances in motion planning, navigation, and control will enable mobile robots to move more safely, quickly, and more confidently in challenging environments. This thesis demonstrates that when mobile robots reason about their environment rather than simply react to it, robots exhibit more intelligent actions.

GLOSSARY

ACKERMANN STEERED VEHICLE	A mobility platform that uses the direction of wheels to control the angular velocity (via curvature) of the vehicle.
COMPLETENESS	The degree to which a motion planning graph contains all solutions.
CONTINUUM OPTIMIZATION	A method for optimal control based on relaxing in continuous space.
CORNER-STEERED VEHICLES	A mobility platform that uses the direction of wheels to control the angular velocity of the vehicle.
CONTROL INPUTS	The signals that modify the state of the vehicle. In mobile robotics, control inputs often appear as desired position rates or linear and angular velocity commands.
CONTROL SET	A description of the motion primitives (actions and trajectories) used to define transitions between nodes in a state lattice. A control set is represented by the variable $\mathbf{U}(\mathbf{s_p}, \mathbf{s_c})$
CRUSHER	A six-wheeled skid-steered mobile robot built by the Carnegie Mellon University's National Robotics Engineering Center.
GLOBAL MOTION PLANNING	A long range motion planner meant to achieve mission objectives. Global motion planners often appear in mobile robot applications as sequential search methods. Also referred to as regional motion planning.
GRAPH SEARCH	See "Sequential Search"
LAGR	A four-wheeled skid-steered mobile robot built by the National Robotics Engineering Center as a testbed for developing machine learning. The arconym "LAGR" stands for "Learning Applied to Ground Robots".
LAGR-EOD	A modified LAGR mobile robot built by the Carnegie Mellon University's National Robotics Engineering Center outfitted with a multiple degree of freedom manipulator.
LOCAL MOTION PLANNING	A short range motion planner designed to keep the vehicle safe and moving towards the global motion planning goal. Local motion planners often appear in mobile robot applications as input sampling methods. Also referred to as navigators.
MOBILE MANIPULATOR	A mobile robot with degrees of freedom in the mobility platform and internal configuration of the system.
MOTION TEMPLATE	See "Control Set"
NAVIGATOR	See definition for Local Motion Planning.
OPTIMAL CONTROL	A problem defined as the minimization of a utility under satisfaction of state and input constraints.

OUTDEGREE	The number of actions in a control set, which can vary as a function of the current state.
PATH	A data structure that represents the vehicle history as a sequence of positions. A path is a subset of a trajectory and can be determined by integrating the predictive motion model with a particular action.
PREDICTIVE MOTION MODEL	A description of the motion of an object under the influence of external forces and inputs. Predictive motion models for mobile robots vary widely in fidelity based on the application from simple kinematic models to full dynamics simulations. The predictive motion model is represented by the variable $\mathbf{f_{PMM}}(\mathbf{x}(t), \mathbf{u}(t), t)$.
REGIONAL MOTION PLANNING	See definition for Global Motion Planning.
RELATIVE OPTIMALITY	A metric in motion planning used to describe the achievable quality of a motion plan given limited runtime or computational resources.
SAMPLING	The degree to which a motion template represents all feasible motions by a vehicle. A motion template with low sampling covers a small portion of possible actions, while one with high sampling covers many possible actions.
SCARAB	A four-wheeled skid-steered mobile robot built by the Carnegie Mellon University's Field Robotics Center with an actively articulating chassis.
SEQUENTIAL SEARCH	An method for optimal control based on searching a discrete graph.
SKID-STEERED VEHICLE	A mobility platform that uses differences in wheel velocities to control the angular velocity of the vehicle.
STATE LATTICE	A recombinant motion planning graph composed by a series of motion templates.
STATE MAPPING EQUATION	The mapping between discrete nodes in a graph and continuous states in the real world. The state transition equation is represented by the variable $\mathbf{f_{SME}}(\mathbf{x})$.
STATE TRANSITION EQUATION	The state change between discrete nodes in a graph. In mobile robot motion planning, this takes the form of integrating the predictive motion model subject to the initial state constraints and the action. The state transition equation is represented by the variable $\mathbf{f_{STE}}(\mathbf{s}, \mathbf{u}(\mathbf{x}, t))$.
TRAJECTORY	A data structure that represents the vehicle history as a sequence of states. A trajectory can be determined by integrating the predictive motion model with a particular action.
UTILITY	An expressive of penalty that can be defined as a function of the vehicle state. This cost can represent instantaneous energy consumption, wheel slip, loss of mobility, slope, path smoothness, obstacle proximity, risk, or any other measure of interest.

BIBLIOGRAPHY

F.B. Amar, Ph. Bidaud, and F.B. Ouezdou. On modeling and motion planning of planetary vehicles. In *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pages 1381–1386, 1993.

O. Amidi. Integrated mobile robot control. Technical Report CMU-RI-TR-90-17, Carnegie Mellon University, 1990.

D. Anderson, T.M. Howard, D. Apfelbaum, H. Herman, and A. Kelly. Coordinated control and range imaging for mobile manipulation. In *Proceedings of the 11th International Symposium on Experimental Robotics*, 2008.

F. Avnaim, J.D. Boissonnat, and B. Faverjon. A practical exact motion planning algorithm for polygonal objectsamidst polygonal obstacles. In *Proceedings of the 1988 IEEE International Conference on Robotics and Automation*, volume 3, pages 1656–1661, April 1988.

D. Baker. Exact solutions to some minimum-time problems and their behavior near inequality state constraints. *IEEE Transactions on Automatic Control*, 34(1):103–106, 1999.

J. Barraquand and J.C. Latobe. On non-holonomic mobile robots and optimal maneuvering. *Revue d'Intelligence Artificielle*, 3(2):77–103, 1989.

P.W. Bartlett, D. Wettergreen, and W. Whittaker. Design of the scarab rover for mobility and drilling in the lunar cold traps. In *Proceedings of the International Symposium on Artifical Intelligence, Robotics, and Automation in Space*, February 2007.

J. Berg, D. Ferguson, and J. Kuffner. Anytime path planning and replanning in dynamic environments. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 2366–2371, Gaithersburg and MD and USA, May 2006.

T. Berglund, H. Jonsson, and I. Soderkvist. An obstacle-avoiding minimum variation b-spline problem. In *Proceedings of the 2003 International Conference on Geometric Modeling and Graphics*, pages 156–161, Los Alamitos, CA, USA, July 2003.

J.J. Besiadecki, P.C. Leger, and M.W. Maimone. Tradeoffs between directed and autonomous driving on the mars exploration rovers. *International Journal of Robotics Research*, 26(91):91–104, January 2007.

J.T. Betts. Survey of numerical methods for trajectory optimization. *Journal of Guidance, Control, and Dynamics*, 21 (2):193–207, 1998.

D. Bonnafous, Sm. Lacroix, and Thierry Siméon. Motion generation for a rover on rough terrains. In *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 784–789, Maui and Hawaii and USA, October 2001.

R.W. Brockett. *Control Theory and Singular Riemann Geometry*. Springer-Verlag, 1981.

J. Canny, B. Donald, J. Reif, and P. Xaiver. On the complexity of kinodynamic planning. In *Proceedings of the 29th IEEE Symposium on Foundations of Computer Science*, pages 306–318, 1988.

G. Chen and T. Fraichard. A real-time navigation architecture for automated vehicles in urban environments. In *Proceedings of the 2007 IEEE Intelligent Vehicles Symposium*, pages 1223–1228, Instanbul, Turkey, June 2007.

M. Cherif. Motion planning for all-terrain vehicles: A physical modeling approach for coping with dynamic and contact interaction constraints. *IEEE Transactions on Robotics and Automation*, 15(2):202–218, 1999.

M. Cherif, C. Laugier, C. Milesi-Bellier, and B. Faverjon. Planning the motions of an all-terrain vehicle by using geometric and physical models. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2050–2057, San Diego, CA, USA, May 1994.

J. Chestnutt. *Navigation Planning for Legged Robots*. PhD thesis, Robotics Institute, Carnegie Mellon University, December 2007.

R.C. Coulter. Implementation of the pure pursuit path tracking algortihm. Technical Report CMU-RI-TR-92-01, Carnegie Mellon University, 1992.

L.B. Cremean, T.B. Foote, J.H. Gillula, G.H. Hunes, D. Kogan, K. Kriechbaum, J.C. Lamb, J. Leibs, L. Lindzey, C.E. Rasmussen, A.D. Stewart, J.W. Burdick, and R.M. Murray. Alice: An information-rich autonomous vehicle for high-speed desert navigation. *Journal of Field Robotics*, 9(23):777–810, 2006.

H. Delingette, M. Gerbert, and K. Ikeuchi. Trajectory generation with curvature constraint based on energy minimization. In *Proceedings of the 1991 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 206–211, 1991.

D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel. Path planning for autonomous driving in unknown environments. In *Proceedings of the 11th International Symposium on Experimental Robotics*, Athens, Greece, July 2008.

L.E. Dubins. On curves of minimal length with a constraint on average curvature and with a prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79:497–516, 1957.

N. Fairfield, G.A. Kantor, and D. Wettergreen. *Real-Time SLAM with Octree Evidence Grids for Exploration in Underwater Tunnels*, 2007.

N. Faiz, S. Agrawal, and R. Murray. Differentially flat systems with inequality constraints: an approach to real-time feasible trajectory generation. *Jouranl of Guidance, Control, and Dynamics*, 24(2):219–227, 2001.

D. Ferguson and A. Stentz. Field D*: An interpolation-based path planner and replanner. In *Proceedings of the International Symposium on Robotics Research*, San Francisco and CA and USA, October 2005.

D. Ferguson and A. Stentz. Using interpolation to improve path planning: The field d* algorithm. *Journal of Field Robotics*, 23(2'):79–101, 2006.

D. Ferguson, T.M. Howard, and M. Likhachev. Motion planning in urban environments. *Journal of Field Robotics*, 25 (11-12):939–960, 2008.

C. Fernandes, L. Gurvitz, and Z.X. Li. *Optimal Nonholnomic Motion Planning for a Falling Cat*, chapter 10. Kluwer Academic Publications, 1993.

D. Fox, W. Burgand, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine*, 4:22–33, March 1997.

E. Frazzoli, M.A. Dahleh, and E. Feron. Real-time motion planning for agile autonomous vehicles. In *Proceedings of the American Control Conference*, Arlington, VA, USA, June 2001.

P.M. Furlong, T.M. Howard, and D. Wettergreen. Model-predictive control for mobile robots with actively reconfigurable chassis. In *To appear in the Proceedings of the 7th Interntional Conference on Field and Service Robotics*, July 2009.

C. Green and A. Kelly. Towards optimal sampling in space of paths. In *Proceedings of the International Symposium on Robotics Research*, Hiroshima and Japan, November 2007.

H. Haddad, M. Khatib, S. Lacroix, and R. Chatila. Reactive navigation in outdoor environments using potential fields. In *Proceedings of the 1998 IEEE Conference on Robotics and Automation*, volume 2, pages 1232–1237, Leuven, Belgium, May 1998.

P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum-cost paths. *IEEE Transactions of Systems Science and Cypernetics*, 4(2), 1968.

T.M. Howard and A. Kelly. Trajectory generation on rough terrain considering actuator dynamics. In *Proceedings of the 5th International Conference on Field and Service Robots*, Port Douglas, Australia, July 2005a.

T.M. Howard and A. Kelly. Terrain-adaptive generation of optimal continuous trajectories for mobile robots. In *Proceedings of the 8th International Conference on Artificial Intelligence, Robotics, and Automation in Space*, Munich, Germany, September 2005b.

T.M. Howard and A. Kelly. Constrained optimization path following of wheeled robots in natural terrain. In *Proceedings of the 10th International Symposium on Experimental Robotics*, Rio de Janeiro, Brazil, July 2006.

T.M. Howard and A. Kelly. Optimal rough terrain trajectory generation for wheeled mobile robots. *International Journal of Robotics Research*, 26(2):141–166, February 2007.

T.M. Howard, C. Green, and A. Kelly. State space sampling of feasible motions for high performance mobile robot navigation in highly constrained environments. In *Proceedings of the 6th International Conference on Field and Service Robots*, Chamonix, France, July 2007.

T.M. Howard, C. Green, D. Ferguson, and A. Kelly. State space sampling of feasible motions for high performance mobile robot navigation in complex environments. *Journal of Field Robotics*, 25(6-7):325–345, 2008.

T.M. Howard, C. Green, and A. Kelly. Receding horizon model-predictive control for mobile robot navigation of intricate paths. In *To appear in the Proceedings of the 6th International Conference on Field and Service Robots*, 2009.

K. Iagnemma, F. Genot, and S. Dubowsky. Rapid physics-based rough-terrain rover planning with sensor and control uncertainty, 1999. URL `citeseer.ist.psu.edu/iagnemma99rapid.html`.

K. Iagnemma, A. Rzepniewski, S. Dubowsky, and P. Schenker. Control of robotic vehicles with actively articulating suspensions in rough terrain. *Autonomous Robots*, 14:5–16, 2003.

G. Ishigami, K. Nagatani, and K. Yoshida. Path planning for planetary exploration rovers and its evaluation based on wheel slip dynamics. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, pages 2361–2366, April 2007.

G Ishigami, K. Nagatani, and K. Yoshida. Slope traversal controls for planetary exploration rover on sandy terrain. *Journal of Field Robotics*, 26(3):264–286, 2009.

J.W. Jackson and P.E. Crouch. Curved path approaches and dynamic interpolation. *IEEE Aerospace and Electronic Systems Magazine*, 6(2):8–13, 1991.

P. Jacobs and J. Canny. Planning smooth paths for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2–7, 1989.

T. Kalmár-Nagy, R. D'Andrea, and P. Ganguly. Near-optimal dynamic trajectory generation and control of an omni-directional vehicle. *Robotics and Autonomous Systems*, 46:47–64, 2004.

Y. Kanayama and B.I. Hartman. Smooth local path planning for autonomous vehicles. In *Proceedings of the International Conference on Robotics and Automation*, volume 3, pages 1265–1270, 1989.

Y. Kanayama and N. Miyake. Trajectory generation for mobile robots. In *Proceedings of the International Symposium on Robotics Research*, pages 16–23, 1985.

L. Kavraki, P. Svestka, J.C. Latombe, and M.H. Overmars. Probablistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4), August 2006.

A. Kelly and B. Nagy. Reactive nonholonomic trajectory generation via parametric optimal control. *International Journal of Robotics Research*, 22(7-8):583–601, 2003.

A. Kelly and T. Stentz. Rough terrain autonomous mobility - part 2: An active vision and predictive control approach. *Autonomous Robots*, 5:163–198, 1998.

A. Kelly, T. Stentz, O. Amidi, M. Bode, D. Bradley, A. Diaz-Calderon, M. Happold, H. Herman, R. Mandelbaum, T. Pilarski, P. Rander, S. Thayer, N. Vallidis, and R. Warner. Toward reliable off road autonomous vehicles operating in challenging environments. *International Journal of Robotics Research*, 25(5-6):449–483, 2006.

S.K. Kim and D.M. Tilbury. Trajectory generation for a class of nonlinear systems with input and state constraints. In *Proceedings of the American Control Conference*, volume 6, pages 4908–4913, 2001.

W.S. Kim, I.A. Nesnas, M. Bajracharya, R. Madison, A. I. Ansar, R.D. Steele, J.J. Biesiadecki, and K.S. Ali. Targeted driving using visual tracking on mars: From research to flight. *Journal of Field Robotics*, 26(3):243–263, 2009.

R. Knepper and A. Kelly. High performance state lattice planning using heuristic look-up tables. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, October 2006.

S. Koenig and M. Likhachev. D* lite. In *In Proceedings of the AAAI Conference of Artificial Intelligence (AAAI)*, pages 476–483, 2002.

K. Komoriya and K. Tanie. Trajectory design and control of a wheel-type mobile robot using b-spline curve. In *Proceedings of the IEEE-RSJ International Conference on Intelligent Robots and Systems*, pages 398–405, 1989.

Y. Koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 1398–1404, Sacramento, California, April 1991.

Y. Kuwata, T. Schouwenaars, A. Richards, and J. How. Robust constrained receding horizon control for trajectory planning. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2005.

S. Lacroix, A. Mallet, D. Bonnafous, G. Bauzil, S. Fleury, M. Herrb, and R. Chatila. Autonomous rover navigation on unknown terrains functions and integration. *International Journal of Robotics Research*, 21(10-11):917–942, 2002.

A. Lacze, Y. Moscovitz, N. DeClaris, and K. Murphy. Path planning for autonomous vehicles driving over rough terrain. In *Proceedings of the 1998 IEEE ISIC/CIRA/ISAS Joint Conference*, pages 50–55, Gaithersburg and MD and USA, September 1998.

F. Lamiraux and J.P. Laumond. Reactive obstacle avoidance and trajectory optimization for nonholonomic systems: Two problems, one solution. In *Proceedings of the 2004 World Automation Congress*, volume 15, pages 473–478, June/July 2004.

J.P. Laumond. Nonholonomic motion planning via optimal control. In *Proceedings of the Workshop on Algorithmic Foundations of Robotics*, pages 227–238, 1995.

J.P. Laumond, M. Taix, and P. Jacobs. A motion planner for car-like robots based on a mixed global/local approach. In *IEEE International Workshop on Intelligent Robots and Systems*, 1990.

S. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

S. LaValle and J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5): 378–400, May 2001.

B.M. Leedy, J.S. Putney, C. Bauman, S. Cacciola, J.M. Webster, and C.F. Reinholtz. Virginia tech's twin contenders: A comparative study of reactive and deliberative navigation. *Journal of Field Robotics*, 23(9):709–727, 2006.

M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun. Anytime dynamic a*: An anytime, replanning algorithm. In *Proceedings of the International Conference on Automated Planning and Scheduling*, June 2005.

C.S. Lin, P.R. Chang, and J.Y.S. Luh. Formulation and optimization of cubic polynomial joint trajectories for industrial robots. *IEEE Transactions on Automatic Control*, 23:1066–1074, 1983.

M. Maimone, A. Johnson, Y. Cheng, R. Willson, and L. Matthies. Autonomous navigation results from the mars exploration rover (mer) mission. In *Proceedings of the International Symposium on Experimental Robotics*, 2004.

D.Q. Mayne, J.B. Rawlings, C.V. Rao, and P.O.M. Scokaert. Constrained model-predictive control: Stability and optimality. *Automatica*, 36:789–814, 2000.

N.A. Melchior, J. Kwak, and R. Simmons. Particle rrt for path planning in very rough terrain. In *Proceedings of the NASA Science Technology Conference 2007*, May 2007.

I. Miller, S. Lupashin, N. Zych, P. Moran, B. Schimpf, A. Nathan, and E. Garcia. Cornell university's 2005 darpa grand challenge entry. *Journal of Field Robotics*, 23(8):625–652, 2006.

A. C. Morris, D. Ferguson, Z. Omohundro, D. Bradley, D. Silver, C. Baker, S. Thayer, C. Whittaker, and W.L. Whittaker. Recent developments in subterranean robotics. *Journal of Field Robotics*, 23(1):35–57, 2006.

R. Murray and S. Sastry. Nonholonomic motion planning: steering using sinusoids. *IEEE Transactions on Automatic Control*, 38:700–716, 1993.

B. Nagy and A. Kelly. Trajectory generation for car-like robots using cubic curvature polynomials. In *Proceedings of the 3rd International Conference on Field and Service Robotics*, June 2001.

S. Nakamura, M. Faragalli, N. Mizukami, I. Nakatani, Y. Kunii, and T. Kubota. Wheeled robot with movable center of mass for traversing over rough terrain. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1228–1233, October-November 2007.

M. Pivtoraiko and A. Kelly. Efficient constrained path planning via search in state lattices. In *Proceedings of the 8th International Symposium on Artificial Intelligence, Robotics, and Automation in Space*, Munich, Germany, 2005.

M. Pivtoraiko, R. Knepper, and A. Kelly. Optimal, smooth, nonholonomic mobile robot motion planning in state lattices. Technical Report CMU-RI-TR-07-15, Carnegie Mellon, 2007.

M. Pivtoraiko, T.M. Howard, I. Nesnas, and A. Kelly. Field experiments in rover navigation via model-based trajectory generation and nonholonomic motion planning in state lattices. In *Proceedings of the 9th International Symposium on Artificial Intelligence, Robotics, and Automation in Space*, February 2008.

M. Pivtoraiko, R.A. Knepper, and A. Kelly. Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics*, 26(3):308–333, 2009.

K.S. Pratt, R. Murphy, S. Stover, and C. Griffin. Conops and autonomy recommendations for vtol small unmanned aerial system based on hurricane katrina operations. *Journal of Field Robotics*, 26(8):636–650, 2009.

N. Ratliff, M. Zucker, J. Bagnell, and S. Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. In *Proceedings of the IEEE International Conference on Robitics and Automation*, 2009.

J.A. Reeds and L.A. Schepp. Optimal paths for a car that goes both forward and backward. *Pacific Journal of Mathematics*, 145(2):367–393, 1990.

J. Reuter. Mobile robot trajectories with continuously differentiable curvature: an optimal control approach. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems*, volume 3, pages 38–43, Victoria, BC, Canada, October 1998.

P. Schenker, P. Pirjanian, T. Huntsberger, H. Aghazarian, E. Baumgartner, K. Iagnemma, A. Rzepniewski, S. Dubowsky, P.C. Leger, D. Apostolopoulos, and G.T McKee. Reconfigurable robots for all-terrain exploration. In *Proceedings of the SPIE Symposium on Sensor Fusion and Decentralized Control in Robotics Systems III*, volume 4196, September 2000.

Z. Shiller. Motion planning for mars rover. In *Proceedings of the First Workshop on Robot Motion and Control*, pages 257–262, 1999.

Z. Shiller and J.C. Chen. Optimal motion planning of autonomous vehicles in three-dimensional terrains. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 198–203, 1990.

Z. Shiller and Y.R. Gwo. Dynamic motion planning of autonomous vehicles. *IEEE Transactions on Robotics and Automation*, 7(2), April 1991.

S. Shimoda, Y. Kuroda, and K. Iagnemma. Potential field navigation of high speed unmanned ground vehicles on uneven terrain. In *Proceedings of the 2005 IEEE Conference on Robotics and Automation*, pages 2828– 2833, Barcelona and Spain, April 2005.

D.H. Shin and S. Singh. Path generation for robot vehicles using composite clothoid segments. Technical Report CMU-RI-TR-90-31, The Robotics Institute, Carnegie Mellon University, 1990.

T. Siméon and B. Dacre-Wright. A practical motion planner for all-terrain mobile robots. In *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1993.

R. Simmons, E. Krotkov, L. Chrisman, F. Cozman, R. Goodwin, M. Hebert, L. Katragadda, S. Koenig, G. Krishnaswamy, Y. Shinoda, W.L. Whittaker, and P. Klarer. Experience with rover navigation for lunar-like terrains. In *Proceedings of the 1995 IEEE Conference on Intelligent Robots and Systems*, volume 1, pages 441–446, August 1995.

S. Singh, R. Simmons, T. Smith, T. Stentz, V. Verma, A. Yahja, and K. Schwehr. Recent progress in local and global traversability for planetary rovers. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1194–1200, San Francisco and CA and USA, April 2000.

J. Snider. Automatic steering methods for autonomous automobile path tracking. Technical Report CMU-RI-TR-09-08, Carnegie Mellon University, 2009.

M. Spenko, Y. Kuroda, S. Dubowsky, and K. Iagnemma. Hazard avoidance for high-speed mobile robots in rough terrain. *Journal of Field Robotics*, 23(5):311–331, 2006.

A. Stentz. The d* algorithm for real-time planning of optimal traverses. Technical Report CMU-RI-TR-94-37, Carnegie Mellon University, 1994.

A. Stentz and M. Herbert. A complete navigation system for goal acquisition in unknown environments. *Autonomous Robotics*, 2(2), August 1995.

M. Tarokh and G. McDermott. Kinematics modeling and analyses of articulated rovers. Technical Report CS/10/2005, University of California - San Diego, 2005.

S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Goffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, and P. Stang. Stanley: The robot that won the darpa grand challenge. *Journal of Field Robotics*, 23(9):661–692, 2006.

D. Tilbury, J.P. Laumond, R. Murray, S. Sastry, , and G. Walsh. Steering car-like systems with trailers using sinusoids. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 3, pages 1993–1998, 1992.

P.G. Trepagnier, J. Nagel, P.M. Kinney, C. Koutsougeras, and M. Dooner. Kat-5: Robust systems for autonomous vehicle navigation in challenging and unknown terrain. *Journal of Field Robotics*, 23(8):509–526, 2006.

C. Urmson, C. Ragusa, D. Ray, J. Anhalt, D. Bartz, T. Galatali, A. Gutierrez, J. Johnston, S. Harbaugh, H. Kato, W. Messner, N. Miller, K. Peterson, B. Smith, J. Snider, S. Spiker, J. Ziglar, W.L. Whittaker, M. Clark, P. Koon, A. Mosher, and J. Struble. A robust approach to high-speed navigation for unrehearsed desert terrain. *Journal of Field Robotics*, 23(8):467–508, 2006.

C. Urmson, J. Anhalt, H. Bae, J. Bagnell, C. Baker, R.E. Bittner, T. Brown, M.N. Clark, M. Darms, D. Demitrish, J. Dolan, D. Duggins, D. Ferguson, T. Galatali, C.M. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T.M. Howard, S. Kolski, M. Likhachev, B. Litkouhi, A. Kelly, M. McNaughton, N. Miller, J. Nickolaou, K. Peterson, B. Pilnick,

R. Rajkumar, P. Rybski, V. Sadekar, B. Salesky, Y. Seo, S. Singh, J.M. Snider, J.C. Struble, A. Stentz, M. Taylor, W.L. Whittaker, Z. Wolkowicki, W. Zhang, and J. Ziglar. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008.

D. Wettergreen, P. Tompkins, C. Urmson, M. Wagner, and W. Whittaker. Sun-synchronous robotics exploration: Technical description and field experimentation. *International Journal of Robotics Research*, 24(1):3–30, January 2005.

R. Zhou and E. Hansen. Multiple sequence alignment using anytime a*. In *In Proceedings of the 18th National Conference on Artificial Intelligence*, 2002.