Improved Action and Path Synthesis using Gradient Sampling

by

Neil Traft

BSc. Computer Science, Tulane University, 2007

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES
(Computer Science)

The University of British Columbia
(Vancouver)

October 2017

Abstract

An autonomous or semi-autonomous powered wheelchair would bring the benefits of increased mobility and independence to a large population of cognitively impaired older adults who are not currently able to operate traditional powered wheelchairs. Algorithms for navigation of such wheelchairs are particularly challenging due to the unstructured, dynamic environments older adults navigate in their daily lives. Another set of challenges is found in the strict requirements for safety and comfort of such platforms. We aim to address the requirements of safe, smooth, and fast control with a version of the gradient sampling optimization algorithm of [Burke, Lewis & Overton, 2005]. We suggest that the uncertainty arising from such complex environments be tracked using a particle filter, and we propose the Gradient Sampling with Particle Filter (GSPF) algorithm, which uses the particles as the locations in which to sample the gradient. At each step, the GSPF efficiently finds a consensus direction suitable for all particles or identifies the type of stationary point on which it is stuck. If the stationary point is a minimum, the system has reached its goal (to within the limits of the state uncertainty) and the algorithm naturally terminates; otherwise, we propose two approaches to find a suitable descent direction. We illustrate the effectiveness of the GSPF on several examples with a holonomic robot, using the Robot Operating System (ROS) and Gazebo robot simulation environment, and also briefly demonstrate its extension to use a version of the RRT* planner instead of a value function.

Lay Summary

Some of those who are most in need of improved mobility are unable to operate a powered wheelchair safely and are denied this crucial improvement to their quality of life.

If a robotic platform were able to liberate the user from the low-level tasks of collision avoidance and smooth steering, it would open up usage to a much wider set of individuals and only require high-level directional commands. Furthermore, this kind of human-robot teamwork would have wide-ranging applications far beyond powered wheelchairs.

Our work is a start down the path toward forms of human-robot teamwork that can meet the safety requirements needed to pass certification for operation of a powered wheelchair. It exhibits safety benefits over more traditional approaches while remaining efficient enough to be applied on a platform of limited power and compute such as a wheelchair.

Preface

The algorithm described in Section 3.4 was originally conceived by my supervisor, Ian Mitchell. This algorithm, along with the experiments in Sections 5.1, 5.2, and 5.4 were presented in our publication: N. Traft and I. M. Mitchell. Improved Action and Path Synthesis Using Gradient Sampling. In *55th IEEE Conference on Decision and Control*, pages 6016–6023. IEEE, 2016. [37] Chapters 3, 4, and 5 were borrowed, with modifications and additions, from this paper.

The implementation of GSPF is based on an earlier version by students Branden Fung and Carolyn Shen.

Table of Contents

List of Tables

List of Figures

Glossary

GSPF Gradient Sampling with Particle Filter

LQG Linear Quadratic Gaussian

MCL Monte Carlo Localization

POMDP Partially Observable Markov Decision Process

QMDP Q-value Markov Decision Process

ROS Robot Operating System

SGD Stochastic Gradient Descent

Acknowledgments

I owe my sincere and heartfelt thanks to my supervisor, Dr. Ian M. Mitchell. He accepted a student who had less than stellar undergraduate grades and some minor exposure to robotics, took him under his wing, gave him ample freedom to explore, gave him direction when he needed it, and introduced him to countless new ideas. My tenure at UBC has been a defining experience and I cannot overestimate its impact on my life and how lucky I have been to have been part of this community.

I would also like to thank Dr. Dinesh Pai and Dr. Elizabeth Croft for allowing me the use of their labs and their robots. It was very rewarding to gain experience on such a wide variety of hardware with distinctly different sensors and linkages, and that has had its own special role in expanding my horizons.

I thank my parents for their love and patience sent from afar, and especially for their support and encouragement on this journey, even though they would have loved to have me close by in Massachussetts. It's good to be missed.

Finally, I would like to thank the Canadian Institutes of Health Research (CIHR) for funding this research through the CANWHEEL Emerging Team in Wheeled Mobility for Older Adults Grant #AMG-100925, and the National Science and Engineering Council of Canada (NSERC) for their support through Discovery Grant #298211.

Chapter 1

Introduction

This research was supported by CanWheel[1], an interdisciplinary team of clinical researchers and basic scientists. The engineering team at CanWheel is concerned with bringing the benefits of power wheelchairs to a wider group of users than is currently possible, particularly by making wheelchairs easier to operate. A key breakthrough in this regard would be a power wheelchair that largely drove itself, or assisted in the avoidance of obstacles. Thus, one of the high-level goals of the researchers at CanWheel is to design, build, and evaluate an autonomous or semi-autonomous powered wheelchair.

Such a wheelchair should be able to operate in ordinary homes and hospitals, should offer a comfortable ride, and should above all have strong safety guarantees. These requirements constitute significant challenges to the underlying robotic system: *navigation in an uncertain environment* and *safe, smooth control*. In this thesis we propose a simple solution to these two common control problems.

1.1 Navigation Under Uncertainty

One of the most significant challenges on the way to realizing this goal is the dynamic, unstructured environments that this type of wheelchair will need to operate in to be successful. A powered wheelchair is most useful in the user's home, in a hospital, or in an elder care facility where older adults with mobility challenges

[1]Website: http://www.canwheel.ca/.

may frequently find themselves. Homes are varying and sometimes cluttered—this makes for an unstructured environment that is difficult to measure and map precisely. Hospitals are even more difficult for the fact of being filled with other people, moving about to their own destinations. This makes for a dynamic environment whose future state is difficult to predict.

Despite decades of robotics research in these kinds of environments—for instance, the museum robot RHINO [6] of 1998 and robotic wheelchairs from at least as early as 1990[2]—navigating safely and robustly in these environments is still a difficult and unsolved problem.

The difficulty of such environments is twofold:

1. Their unstructured, possibly cluttered nature gives rise to a large amount of *uncertainty* in the current and future state of the environment. Planning under uncertainty is a rich field and will be discussed in Chapter 2.

2. Their dynamic nature requires the robot to perceive and react quickly to changes in the environment.

These two features are difficult to reconcile. Planning under uncertainty requires the consideration of not a single observed state, but *all possible* states of the environment. The resulting computational complexity makes it much harder to plan as quickly as feature #2 requires.

1.2 Safe, Smooth Control

It is our goal to generate a *safe and smooth* control signal, given a map of obstacles. These two goals can be contradictory.

To synthesize safe controls, i.e. controls which avoid collisions to the best of the robot's ability, we require that the robot's movements be subject to a cost which increases as the robot approaches an obstacle. Given such a cost function, we generate an optimal control policy. In this framework, an optimal control is a safe control.

Many techniques have been developed for synthesizing (approximately) optimal feedback control inputs/actions in the control and robotics literature. Informally,

[2]See [14] for a survey of early progress in intelligent wheelchairs.

we seek progression to a prescribed state with minimal cost and a smooth input sequence. This can often be accomplished for systems with linear dynamics and quadratic cost functions (or problems which can be reasonably approximated in that manner). However, the dynamics of a powered wheelchair are distinctly nonlinear, and a cost function which depends on cluttered obstacles will be significantly more complex than a simple quadratic. Hence, our requirement for safety gives rise to nonsmooth feedback controls.

1.3 Navigation with Value Functions

The uncertainty in future state makes it important to plan via feedback control and not open loop: since we cannot know future states, we cannot plan a sequence of optimal actions ahead of time. One method for optimal feedback control is the use of *value functions*.

A feedback control requires us to define a *navigation function* over the robot's configuration space. A navigation function is a function from state space to control space, $\pi\colon X \to U$. For each state in the state space, it gives an appropriate action, so the robot can make progress toward the goal from wherever it happens to find itself. For optimal control, the navigation function can be efficiently derived from the value function.

The value function returns the minimum cost to go to the goal from any given state. As we will explain in Chapter 3, these functions can be constructed as the solution to a dynamic programming problem. As our cost functions from the previous section are potentially nonconvex and nonsmooth, so too is the value function. The optimal action is to take the action which results in the maximum decrease in this function; i.e., to follow the direction of steepest descent. Thus, our solution to this problem amounts to steepest descent on a nonsmooth, nonconvex function—albeit a special class of nonconvex functions with a single global minimum, thus not having the possibility of converging to a local minimum.

1.4 An Example

In order to illustrate this challenge, consider the narrow corridor scenario shown in figure 1.1. For simplicity, we work with an isotropic, holonomic vehicle in the

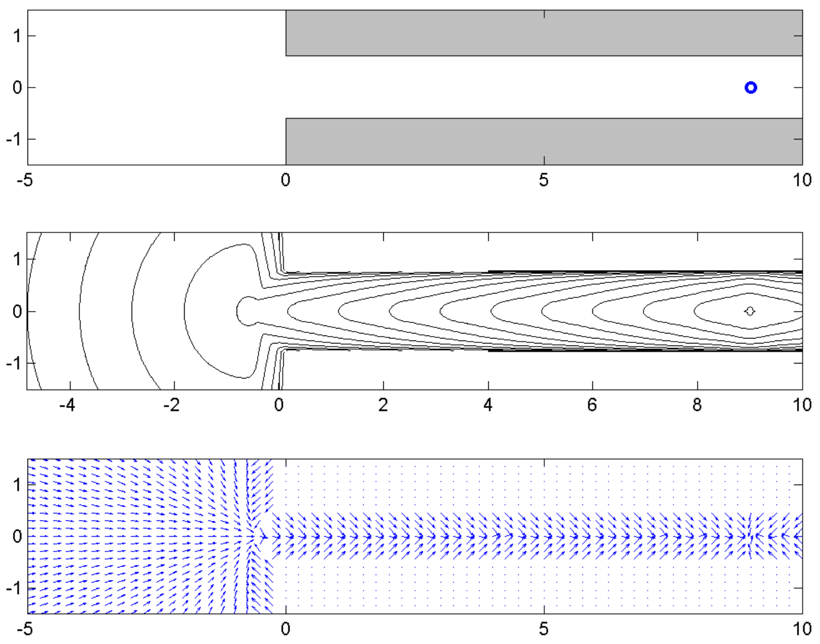

Figure 1.1: Narrow corridor example. Top: Obstacles are gray, the goal location is a blue circle. Middle: Contours of the value function. The corresponding cost function penalizes states near the walls. Bottom: Vector field of the gradients of the value function (on a subsampled grid for visibility). All gradients in the corridor have a rightward component, but there is an abrupt jump between upward and downward components.

plane: It can move in any direction at some bounded maximum speed. The vehicle is trying to reach a goal location on the right side of a narrow corridor. The objective is to minimize path length, but for safety purposes we penalize states that are close to the walls. To solve the scenario, we approximate a minimum time to reach value function in the obstacle free space, and then the resulting optimal feedback controller follows the gradient of the value function.

The first problem arises even with numerical simulations where we know the exact state: Chattering of the optimal control as illustrated in figure 1.2. A typical cyber-physical control system for vehicles and robots operates on a loop in which new control signals are generated at roughly periodic time intervals. If the control signals arise from gradients of a value function, this approach is mathematically

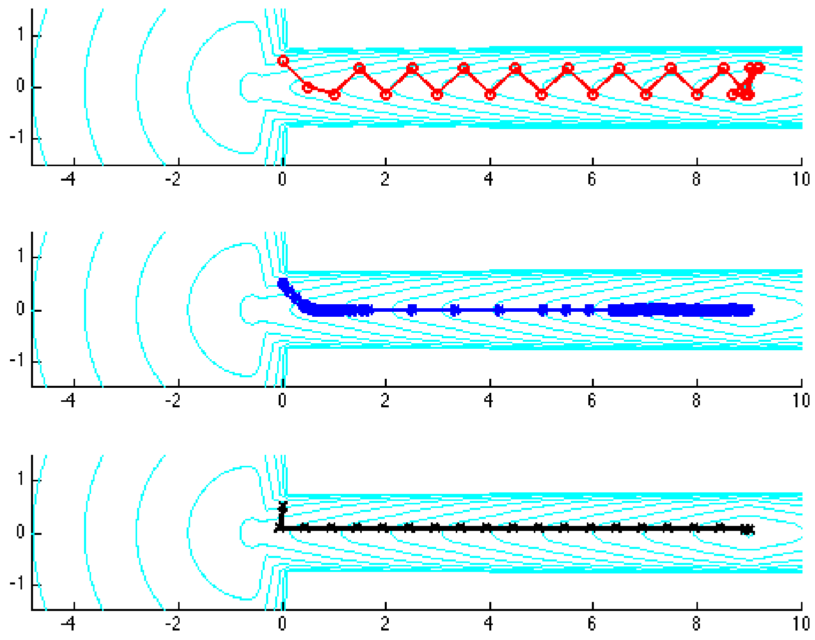

Figure 1.2: Narrow corridor example paths. Every figure is overlayed with contours of the value function, and markers are placed along the trajectory at each step. Top: Fixed stepsize path. Middle: Adaptive stepsize path from MATLAB's `ode15s` (an implicit scheme). Bottom: Sampled gradient path.

equivalent to fixed stepsize gradient descent (essentially forward Euler integration) and gives rise to significant chattering. A naive response is to use a variable stepsize integrator such as MATLAB's `ode23` or `ode45`. It turns out that they can generate decent paths, but require an unreasonable number of tiny steps. Those familiar with numerical integration will immediately diagnose stiffness as the problem, and prescribe an implicit integrator such as MATLAB's `ode23t` or `ode15s`. They take fewer steps but require much longer to run because they still generate many tiny steps in parts of the domain where the optimal path is straight. In fact, the problem arises because the value function is (nearly) non-differentiable along the center of the corridor, and consequently the gradient is (nearly) discontinuous precisely where the optimal path lies. Put another way, the gradient descent differential equation is not just stiff in a normal sense, but infinitely so. Figure 1.2 also illustrates how our

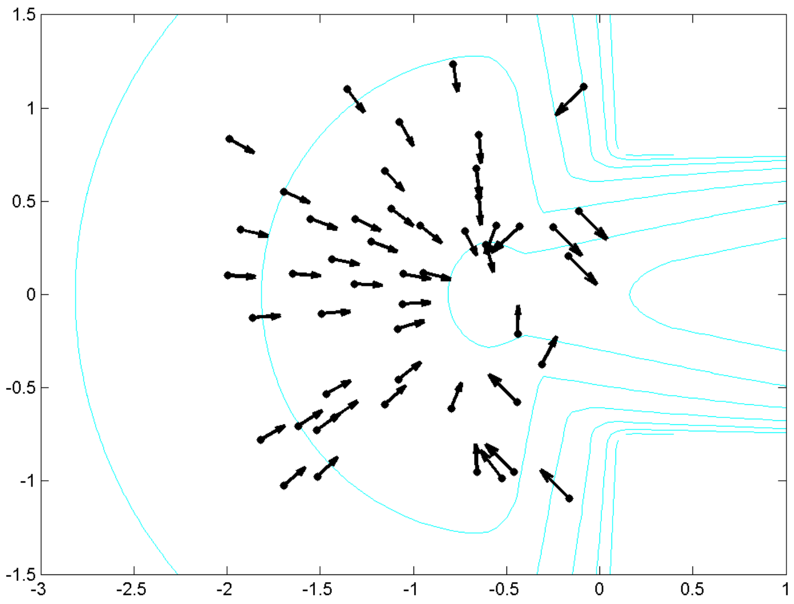

Figure 1.3: State uncertainty, as represented by samples from a (simulated) particle filter. Figure shows a zoomed in view of the left side of the corridor scenario from figure 1.1. Each dot is a state sample, and the attached arrow shows the optimal action for that sample. Depending on which sample most accurately represents the actual state, the optimal action could be in any direction.

proposed approach generates a much more desirable solution with large stepsize and accurately optimal path.

The second problem arises in physical systems in which the state is not accurately known. Figure 1.3 shows a commonly used non-parametric representation of state uncertainty called the particle filter: Each point represents a possible state of the system. Every direction of the compass is covered by at least one particle's optimal action. The typical mechanism for choosing an action—extract the mean state or most probable state and use the corresponding optimal action—will choose one of these actions and will not even recognize that any action may be counter-productive. In fact, the system should refine its state estimate if possible before choosing an action.

With these two problems in mind, the contributions of this thesis are to show:

- How the simple gradient sampling algorithm [8] borrowed from nonsmooth optimization can easily be combined with a standard particle filter representation of state uncertainty.
- That the resulting algorithm can at each step generate a productive action choice which takes into account current state uncertainty or determine that no such action exists.
- That the resulting action choices nearly eliminate the chattering often observed over multiple steps when using gradient descent based path planners.

We illustrate its success in simple simulation with figure 1.2, but more importantly we illustrate its success in a full scale robotic simulation with state uncertainty and simulated noisy sensors and motion, using the widely adopted Robot Operating System (ROS) / Gazebo environment.

The algorithm does give rise to one undesirable behavior: it tends to steer toward not just minima of the value function but any stationary point, including saddle points. Consequently, we propose a procedure for categorizing whether a stationary point is the desired minimum and discuss two approaches to resolve saddle points. A pleasant side-effect of the categorization algorithm is automatic determination of whether the goal has been reached to the degree possible given the current state uncertainty. Finally, we demonstrate that our algorithm is easily adapted to use other underlying optimal planners, such as RRT*.

Chapter 2

Related Work

The problem of "robustness to uncertainty" is broad and can be tackled in many ways. Under this umbrella there exists everything from very general mathematical formalisms to highly domain-specific solutions to a narrow aspect of the problem. The Gradient Sampling with Particle Filter (GSPF) algorithm sits somewhere in between these two extremes, providing robustness to a very general set of systems and uncertainty, but with assumptions that limit its applicability in exchange for remaining tractable. It is a relatively uncomplicated way to account for existing state uncertainty and produce more stable paths, but does not attempt to model future uncertainty.

2.1 Types of Uncertainty

There are many sources of uncertainty in real-world robotic planning, often grouped into three main categories [42]:[1]

- Motion uncertainty
- Environmental uncertainty
- Sensing uncertainty

Motion uncertainty is non-determinism in the robotic control system. It is an expression of the fact that the next state is a stochastic function of the current state

[1]Sometimes the second category is split to create a total of four groups, as in [18].

and the control. Even when the current state and control are perfectly known, future robot states cannot be exactly predicted.

The future state of the environment around the robot is similarly non-deterministic. This is the result of the many possible forms of *environmental uncertainty*: our prior maps of the environment may not be exact; other objects in the environment may be subject to motion uncertainty; there may be other agents in the environment who operate under unknown motion models or whose controls we cannot observe.

Sensing uncertainty is non-determinism in the observation function. It contributes uncertainty to both the state of the robot and the state of the environment. When taken together with the first two types of uncertainty, the result is that not only can the state not be predicted in the future, it cannot even be observed at the present time.

2.1.1 Selective Focus

Work which addresses planning under uncertainty may sometimes only address one or two of these categories. For instance, works which address adversarial games [33] or navigation among pedestrians [13, 38] often focus only on the uncertain intention of their opponent (i.e. environmental uncertainty).

More commonly, motion and sensing uncertainty are considered together, but environmental uncertainty is ignored. This is because many works assume a static world with an accurate map; in this case, the only uncertainty is in the state of the robot (the sources of that uncertainty being motion and sensing).

This is even the case for implementations of the very general Partially Observable Markov Decision Process (POMDP) model, since many POMDP solutions assume the cost and transition functions are time-invariant, which is only true under a static environment. Some attempt to get around this by computing a policy over the belief state of the entire environment, but this is impractical for all but the smallest environments, such as the *Tag* example in [28] or the intersection example in [2]. An exception is online POMDP solvers, which can update their policy according to new observations and beliefs [31], but these still have the difficulty of representing all possible observations (and thus all possible evolutions of the environment) which could occur from the current belief state.

2.1.2 Where This Work Fits

GSPF can deal with motion and sensing uncertainty, by sampling from the state estimate. The formulations demonstrated in this thesis cannot deal well with environmental uncertainty, because the underlying planners assume a static environment and are not well-equipped for fast replanning. However, if given a planner with that capability, then the gradient sampling method would naturally accommodate it. Recent work in high-frequency replanning shows that some highly parallelized planners may be fast enough to introduce some robustness to dynamic obstacles [34].

2.2 Types of Robustness

In addition to there being many types of uncertainty to be robust to, there are many different *ways* that one can be robust. One might minimize the probability of collision, or maximize the probability of success. One might minimize some cost; this could be the expected cost, or the nominal cost, or the cost subject to some chance constraints on the probability of collision. Or one may simply minimize uncertainty directly, with hope of enabling traditional collision avoidance approaches.

2.2.1 POMDP Solvers

Minimizing the *expected* cost of a policy is a common approach. This is the objective of the general POMDP formulation. The general applicability of POMDPs makes them very attractive, but they are also known to be PSPACE-complete (and thus very unlikely to be solvable in polynomial time) [26]. While there have been significant advances in making POMDPs tractable by sampling beliefs [2, 28], to our knowledge there is not yet any solution which can be applied in real-time (replanning multiple times per second) to continuous state and action spaces—a prerequisite to employing the technique in robotics.

There has, however, been some promising recent work in online and real-time POMDP solvers. Monte Carlo Tree Search within the belief space has enabled online solvers in large state spaces [33], but is yet to be applied to robotic applications or continuous action spaces. More recently, a similar concept called an Adaptive Belief Tree was combined with Generalized Pattern Search to enable the use of continuous action spaces, and although it is not quite as fast as we would like, the initial results are very promising [32]. Similarly to our method, both of these can be applied using only a black-box simulator of the robotic system, and do not require representing the full transition or observation probability densities. Like real-time performance, this is another essential practicality which many POMDP solutions lack—essential because enumerating or even sampling from the space of possible observations is very difficult.

2.2.2 Other Approaches

In the absence of the ideal POMDP solver, there have been a plethora of alternative methods of varying applicability. Some even boast features that are not enabled by POMDPs, such as directly maximizing the probability of success [34]. Others are slightly more limited, minimizing expected cost subject to an upper bound on probability of collision [1, 9, 12]. Still more limited are the methods which only minimize expected cost, with no estimation on probability of collision; this includes most methods based on Linear Quadratic Gaussian (LQG) control, e.g. [35, 41].

Moving on from there, there have been methods which minimize the cost of the *nominal* trajectory and then attempt to track this trajectory, rather than minimizing the *expected* cost of a feedback policy [21, 22] (although [22] does admit modeling risk in the objective function, something that is not often permitted). Alternatively, LQG-MP [40] does not optimize the cost directly, but is able to choose the plan with lowest probability of collision from a set of candidate plans.

Some methods minimize the state uncertainty directly, rather than having this fall out naturally as a side effect of minimizing expected cost. The Bayesian framework presented in [9] can minimize uncertainty while keeping task time bounded, or minimize time while keeping uncertainty bounded. Belief Road Map [29] does not bound or minimize covariance over the whole trajectory, but it can minimize *final* covariance.

As methods vary in their attack on uncertainty, so do they vary in their assumptions and limitations. It is common to address only Gaussian beliefs [1, 5, 9, 12, 27]. Many methods are designed specifically for linear systems, or linear-quadratic, or linearized-quadratic [21, 22, 34, 40, 41]. Some methods are based on tracking a nominal trajectory, and if they deviate too far from the nominal plan their policy may no longer be applicable [5]—this includes most of the LQ methods as well. Similarly, some methods perform local optimizations on a nominal plan, and cannot claim to be globally optimal [27, 41].

Certain methods operate on discrete space/observations only [28]. BU-RRT* [21] requires polytopic obstacles. It also requires knowing and modeling of all uncertainties present in the environment, as do many POMDP solvers [2, 28] (because they need to be able to fully enumerate transition and observation probability densities).

2.2.3 Comparison To This Work

GSPF does a bit more than minimizing the cost of a deterministic trajectory—at each step, it takes the state uncertainty into account to guarantee progress. We have a mechanism to identify when uncertainty is too large to guarantee progress, and to attempt relocalization in that event. In this way, the system localizes to the extent necessary to reach the goal, but otherwise does not explicitly work to minimize or bound uncertainty.

Although not intentional, GSPF does something very similar to Q-value Markov Decision Process (QMDP) [20, 31]. QMDP is a POMDP approximation which assumes that all partial observability will disappear after a single step. The current state uncertainty and the motion uncertainty are represented, but it is assumed that in the next step the state will be perfectly known. Likewise, in GSPF the action chosen for each particle is optimal assuming that further evolution is deterministic, so the GSPF consensus direction likewise incorporates the assumption of deterministic future evolution. Because they do not consider partial observability, neither method is capable of information-gathering actions which may help reduce uncertainty, although GSPF is capable of identifying when the uncertainty is too large to guarantee progress.

QMDP resolves conflicting actions by choosing the one with the highest expected value (where the expectation is run over only the current belief state). This typically requires discrete state and action sets so that this optimization can be performed efficiently. GSPF, on the other hand, is framed around a continuous action space, and in fact *requires* continuity of the actions and the value function. Rather than the maximum *expected* reward, we choose an action which *guarantees a positive reward*. This favors choosing actions safely over choosing actions greedily.

Chapter 3

Problem & Background

At a high level, we seek to choose a sequence of direction commands which will navigate a robot from its current location to a known goal location in a known map. Although the robot is able to move in any direction, the problem is non-trivial because the map contains obstacles which must be avoided. Furthermore, the robot is unsure of its current position, may not move exactly as commanded, and receives noisy sensor information from which it must estimate its position and movement. A very common approach to solve this problem is to plan deterministically optimal paths, track state uncertainty online, but then choose the action which is deterministically optimal for some single possible state. Such an approach can be theoretically justified by the separation principle for linear time-invariant systems with Gaussian noise [3], but most mobile ground robots and their sensors are highly nonlinear so the widespread adoption of this pipeline is due to its ease of implementation and frequent experimental success (modulo the chattering issue mentioned earlier).

In this chapter we outline the elements of our optimal navigation pipeline, and in the next chapter we will show how these elements can be naturally integrated. The individual elements are common robotic tools, but are not typically used together. We track uncertainty with the popular particle filter representation. Optimal actions are chosen by gradient descent on a value function approximation. These two elements will be unified by the gradient sampling algorithm from the non-smooth optimization literature, which is also described in this chapter.

3.1 Problem Formulation

Our goal is to navigate the robot through the state space Ω to some compact target set $\mathscr{T}$; we present results for $\Omega \subseteq \mathbb{R}^2$, but it is straightforward to extend the approach to higher dimensions. We seek paths $x(\cdot) : [t_0, t_f] \to \Omega$ that are optimal according to an additive cost metric

$$\psi(x_0) = \inf_{x(\cdot)} \int_{t_0}^{t_f} c(x(s))\, ds, \tag{3.1}$$

where $x(0) = x_0$, $x(t_f) \in \mathscr{T}$ and $x(\cdot)$ are drawn from the set of feasible paths such that $x(t) \in \Omega \setminus \mathscr{T}$ for $t_0 \leq t < t_f$. The value function $\psi(x)$ measures the minimum cost to go from state x to $\mathscr{T}$ (this cost may not be achievable, but paths whose costs are arbitrarily close to $\psi(x)$ exist). The cost function $c(\cdot)$ is assumed to be strictly positive and Lipschitz continuous.

The optimal solution of (3.1) depends on the feasible paths. Here we will assume only the simplest form of isotropic holonomic dynamics

$$\tfrac{d}{dt}x(t) = \dot{x}(t) = u(t), \tag{3.2}$$

where $\|u(t)\| \leq 1$ (all norms are assumed to be Euclidean $\|\cdot\| = \|\cdot\|_2$ unless otherwise specified). We assume the input signal $u(\cdot)$ is measurable and hence $x(t)$ is continuous.

3.2 Planning with a Value Function

The value function (3.1) satisfies a dynamic programming principle and can be shown to be the viscosity solution of the Eikonal equation (for example, see [39])

$$\begin{aligned} \|\nabla \psi(x)\| &= c(x), && \text{for } x \in \Omega \setminus \mathscr{T}; \\ \psi(x) &= 0, && \text{for } x \in \mathscr{T}. \end{aligned} \tag{3.3}$$

Under the dynamics (3.2), the viscosity solution $\psi(x)$ of (3.3) is continuous and almost everywhere differentiable. The only local minima of $\psi(x)$ occur at $\mathscr{T}$, but the function will usually be non-convex: it can have saddle points in $\Omega \setminus \mathscr{T}$ and local maxima at boundaries of Ω which are not also boundaries of $\mathscr{T}$. It will typically not be differentiable at these critical points, as well as on other lower dimensional subsets of the domain.

Except on simple domains Ω and for simple cost functions $c(x)$, it is not practical to find the viscosity solution of (3.3) analytically; however, there exist many efficient approaches to approximate it; for example [11, 16, 19, 39]. In order to handle complex domains and cost functions, approximations are generally constructed on a discrete grid.

Given the value function, the optimal state feedback action is easily extracted

$$u^*(x) = \frac{\nabla \psi(x)}{\|\nabla \psi(x)\|}. \tag{3.4}$$

Consequently, generation of an optimal path is equivalent to gradient descent of $\psi(x)$. Of course we typically cannot represent the exact solution of (3.2) and (3.4) either, so we seek an approximate path in the form of a sequence of waypoints $\{x(t_i)\}_i$ for some sequence of timesteps $t_0 < t_1 < t_2 < \cdots < t_f$ and some initial x_0. A common approach is essentially a forward Euler integration with fixed timestep Δt

$$\begin{aligned} t_{i+1} &= t_i + \Delta t, \\ x(t_{i+1}) &= x(t_i) + \Delta t u^*(x(t_i)). \end{aligned} \tag{3.5}$$

Unfortunately, a straightforward implementation of (3.5) to generate paths from the value function (or even a fancier implementation with adaptive stepsize) falls prey to the well-established problem with gradient descent (as illustrated in the top two subplots of figure 1.2): The resulting paths chatter or take many steps to achieve the optimum. This outcome is not surprising, since the optimal paths often proceed down the middle of steep-sided valleys in the value function, and in these valleys the value function displays the large disparity in curvature that causes gradient descent such problems. In fact, the value function may not be differentiable there, so the gradients change discontinuously and the disparity in curvature is infinite.

A further complication arises because $\psi(x)$ and hence $\nabla\psi(x)$ are approximated numerically. The numerical algorithms will typically return an approximation of $\nabla\psi(x)$ even at values of x where the true $\psi(x)$ is not differentiable, and more generally the approximate values of $\nabla\psi(x)$ may be inaccurate near these regions where differentiability fails.

3.3 State Estimation with Particle Filters

The particle filter is a popular technique for state estimation or localization of systems with nonlinear dynamics and/or sensor models. We focus on a version commonly used in robotics called Monte Carlo Localization (MCL) [36]. The state estimate is represented by a collection of weighted samples $\{(w^{(k)}(t), x^{(k)}(t))\}$. This estimate is updated by predictions whenever the system state evolves and corrections whenever sensor readings arrive, typically in an alternating iteration. Predictions update only the state component by drawing a new sample

$$x^{(k)}(t_{i+1}) \sim p(x(t_{i+1}) \mid x^{(k)}(t_i), u(t_i)), \tag{3.6}$$

where $p(x(t_{i+1})|x(t_i), u(t_i))$ is the probability distribution over future states given past state and input; in other words, the dynamics (3.2) plus some motion noise. Corrections update only the weight component by multiplication

$$w^{(k)}(t_{i+1}) = p(\text{sensor reading} \mid x^{(k)}(t_{i+1}))\, w^{(k)}(t_i),$$

where $p(\text{sensor reading}|x^{(k)}(t_{i+1}))$ models the probability of seeing the sensor reading given the particle's current state.

In MCL the particle representation is also regularly resampled, typically after each sensor reading. During resampling, a new collection of particle locations is drawn (with replacement) from the existing locations with probability proportional to the existing particles' weights, and the weights are all reset to unity. In the remainder of the paper we will work with the state estimate only after resampling, so we assume unit weights.

3.4 The Gradient Sampling Algorithm

It is well known that gradient descent performs poorly on nonsmooth optimization problems, and many algorithms have been proposed to overcome its limitations. Here we draw inspiration from the gradient sampling algorithm [8], which is designed to generate a sequence of high quality linesearch directions despite the presence of discontinuous and/or poorly approximated derivatives in the objective function.

The basic algorithm is given a function ψ and a starting point $x(0)$, and aims to efficiently locate a minimum of the function. In each iteration starting at the current point $x(t)$, it seeks to choose a point $x(t+1)$ in a way that guarantees descent $\big($so that $\psi(x(t+1)) \leq \psi(x(t))\big)$. For a differentiable function this is trivially achieved by performing a line search in the direction of the negative gradient to find an appropriate step size $\gamma(t)$:

$$x(t+1) = x(t) - \gamma(t)\frac{\nabla\psi(x(t))}{\|\nabla\psi(x(t))\|}.$$

However, as we have mentioned before, we are interested in value functions which are not differentiable everywhere. This can be addressed through the use of the *generalized gradient*, also referred to as the *Clarke subdifferential*. Informally, the Clarke subdifferential is the compact set of directions of maximum descent from a given point. It can be defined in terms of the convex hull of gradients in the neighborhood around a point $\bar{x} \in \mathrm{R}^n$

$$\partial_C\psi(\bar{x}) = \mathrm{conv}\{\lim_r \nabla\psi(x_r) : x_r \to \bar{x}, x_r \in Q\}$$

where Q is a full-measure subset of points in the neighborhood of $\bar{x}$ which are differentiable. See [10] for a full exposition.

Given the set $\partial_C\psi(\bar{x})$, we can use any vector in this set as a valid direction of descent, and perform a line search as in the steepest descent case. However, this set is difficult to compute. We can instead *approximate* the generalized gradient by taking a sampling of gradients in the same neighborhood [7]. This is the key idea in the gradient sampling algorithm.

The algorithm takes a sampling of gradients at points within a radius ε,

$$x^{(k)}(t) = x(t) + \varepsilon\,\delta x^{(k)}, \tag{3.7}$$

$$p^{(k)}(t) = \nabla\psi(x^{(k)}(t)) \tag{3.8}$$

for $k = 1, \ldots, K$, where $\{\delta x^{(k)}\}$ are sampled independently and uniformly from the

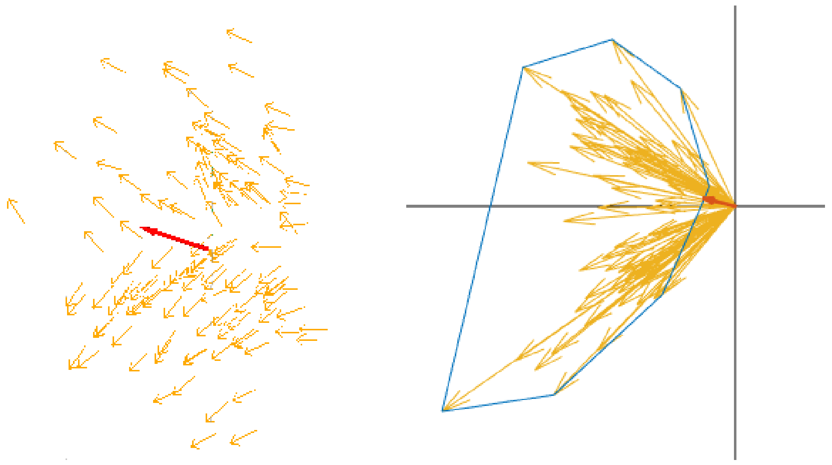

Figure 3.1: Finding a "consensus" among different actions. Left: Gradient vectors (yellow) shown at the corresponding samples' locations, and the resulting consensus action (red). In this neighbourhood, the value function has a ridge. Right: The gradients (yellow) plotted in gradient space, their convex hull (blue) and $p^*(t)$ (red). The consensus action is a leftward movement, which is a descent direction for all samples.

unit ball. Next, the algorithm computes the generalized gradient approximation,

$$P(t) = \mathrm{conv}\{p^{(1)}(t), \ldots, p^{(K)}(t)\}.$$

The algorithm chooses the point with the minimum norm

$$p^*(t) = \operatorname*{argmin}_{p \in P(t)} \|p\|^2 \tag{3.9}$$

The authors of [8] cite a desire to be conservative in making this choice, as well as the ability to prove convergence under this choice.

Thus, the direction $p^*(t)$ is a consensus direction: a direction of descent from all samples $x^{(k)}(t)$. Any rescaling of $p^*(t)$ is also a consensus direction. Most importantly, it can easily be found by solving a simple convex quadratic optimization problem. The technique is illustrated in figure 3.1.

If $\|p^*(t)\| = 0$, there is a Clarke ε-stationary point—conceptually a local minimum, maximum or saddle point—somewhere within the ε-ball around $x(t)$, and no direction can be agreed upon by all samples. In this case the radius ε is reduced and a new sample set (3.7)–(3.8) is obtained.

Otherwise, the algorithm performs an Armijo line search along the vector given by $p^*(t)$ to determine an appropriate step length s, and the update is given by

$$x(t_{i+1}) = x(t_i) - s\frac{p^*(t)}{\|p^*(t)\|} \tag{3.10}$$

The algorithm terminates when ε shrinks to a predetermined threshold. When paired with a linesearch procedure that ensures sufficient descent, such as Armijo, it is shown to converge to a Clarke stationary point under suitable conditions. When we repurpose this algorithm to path planning, we will have to deal with the possibility of arriving at a stationary point which is not the desired minimum.

Chapter 4

Gradient Sampling with Particle Filter

The basic gradient sampling algorithm can generate a path for simulations, visualizations, and other situations where the initial condition x_0 is known, the dynamics (3.2) are accurate, and the chosen input (3.4) is accurately implemented. Unfortunately, for most robotic systems these assumptions do not hold. In this chapter we consider how the algorithm can be adapted to the case where $x(t)$ can only be estimated.

4.1 Gradient Sampling with State Uncertainty

The state estimate representation used by particle filters suggests a natural adaptation of the gradient sampling algorithm: Instead of choosing the gradient sample locations with (3.7), use the particles' locations (3.6) directly. We call this version GSPF, and outline its key properties in the following propositions.

Proposition 1. *If the solution $p^*(t)$ of (3.9) is such that $\|p^*(t)\| \neq 0$, then $p^*(t)$ is a descent direction in the value function for all particles.*

Proof. A direction d is a descent direction at point x if and only if $d^T p(t) < 0$, where $p(t)$ is the gradient of the value function at point $x(t)$. The descent direction chosen by GSPF will be $d = -p^*(t)$. So to produce a descent direction for all particles, we must show that $p^*(t)^T p^{(k)}(t) > 0$, $\forall i \in \{1, \dots, K\}$. In other words, the minimum norm convex combination of gradients forms an acute angle with every

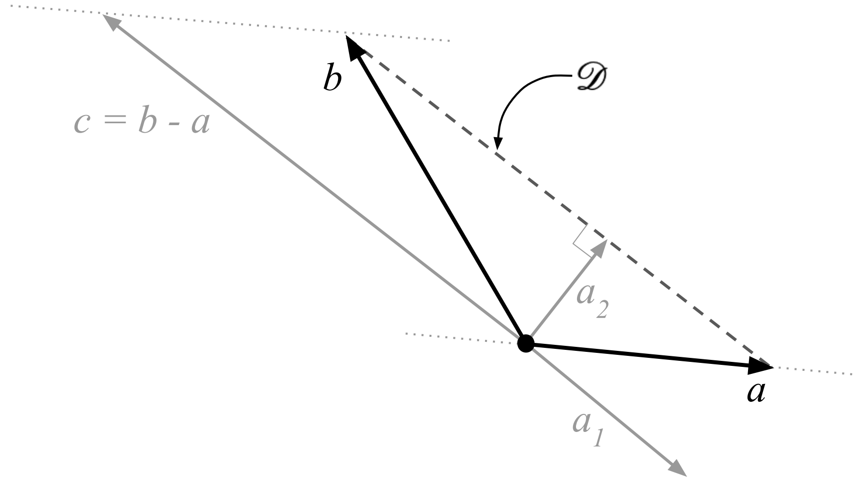

Figure 4.1: The minimum norm direction $p^*(t)$ must form an acute angle with every point in the convex hull in order to be a direction of descent for all particles. We show that if this is not the case for a potential solution a, then a must not be minimum norm.

particle's gradient. We will prove the stronger property that the minimum norm choice forms a descent direction with not just any *single* particle, but any *convex combination* of particles: $p^*(t)^T p > 0,\ \forall p \in P(t)$.

For simplicity of notation, let a be the proposed solution to (3.9), and let $b \in \mathscr{C}$ be any convex combination of gradient samples:

$$\mathscr{C} = \Big\{ \sum_k \lambda_k p^{(k)}(t) : \sum_k \lambda_k = 1,\ \forall k : \lambda_k \geq 0 \Big\}.$$

To be the true solution, the proposal a must have the minimum norm:

$$a = \operatorname{argmin} \|a'\|\ \forall a' \in \mathscr{C}.$$

We will show that a must have a positive dot product with all gradient samples, because if it did not, then there would be a choice with a smaller norm:

$$a^T b \leq 0 \implies \exists a' \in \mathscr{C} : \|a'\| < \|a\|. \tag{4.1}$$

Note that since the minimum norm $p^*(t) > 0$, the magnitudes of all vectors in the convex combintation must be greater than zero: $||a|| > 0$ and $||b|| > 0$.

Let $c = b - a$ and $\mathscr{D} = \{(1-\lambda)a + \lambda b : \lambda \in [0,1]\}$. Note that $\mathscr{D} \subset \mathscr{C}$ (by virtue of convexity, all points on the line segment between a and b are within the convex hull).

Let a_1 be the projection of a onto c.

$$\begin{aligned}
a_1 &= \frac{a^T c}{c^T c} c \\
&= \frac{a^T (b-a)}{(b-a)^T (b-a)} (b-a) \\
&= \frac{a^T b - ||a||^2}{||a||^2 + ||b||^2 - 2a^T b} (b-a) \\
\text{Let } \rho &= \frac{a^T b - ||a||^2}{||a||^2 + ||b||^2 - 2a^T b} \\
\text{So } a_1 &= \rho(b-a) \qquad (4.2)
\end{aligned}$$

Let a_2 be the rejection of a from c.

$$\begin{aligned}
a_2 &= a - a_1 \\
&= a - \rho(b-a) \\
&= (1+\rho)a - \rho b
\end{aligned}$$

By Pythagoras, we have:

$$\begin{aligned}
& ||a_1||^2 + ||a_2||^2 = ||a||^2 \\
\implies & ||a_2||^2 = ||a||^2 - ||a_1||^2 \\
\implies & ||a_2|| < ||a|| \text{ if } ||a_1|| > 0 \qquad (4.3)
\end{aligned}$$

Now we have that a_2 is a vector which lies somewhere on the line passing through a and b. If we let $\lambda = -\rho$ as defined by equation 4.2, we note it has the same structure as our set $\mathscr{D}$:

$$\begin{aligned} a_2 &= (1+\rho)a - \rho b \\ &= (1-\lambda)a + \lambda b \end{aligned}$$

Expanding this choice of λ,

$$\lambda = -\rho = \frac{\|a\|^2 - a^T b}{\|a\|^2 + \|b\|^2 - 2a^T b} \tag{4.4}$$

we can see that if $a^T b \leq 0$ then the above quantity is positive. We can also see the denominator is strictly greater than the numerator. Therefore, $0 < \lambda < 1$, which places a_2 within the set $\mathscr{D}$.

Referring back to equation 4.2, and the fact that $\|a\| > 0$ and $\|b\| > 0$ and $a^T b \leq 0 \implies \rho \neq 0$, we can also see that $\|a_1\| > 0$. Therefore, by equations 4.3 and 4.4, we have demonstrated our assertion 4.1: specifically, we have shown that $a_2 \in \mathscr{C}$ and $\|a_2\| < \|a\|$, so a is not the minimum norm in the convex hull $\mathscr{C}$ and our leading assumption that $p^*(t) = a$ cannot be true if there exists a b such that $a^T b \leq 0$.

The particle gradient samples are within the convex hull, so the property which holds for b holds for them as well: the minimum norm choice must be a descent direction for all.

□

Proposition 2. *If the solution $p^*(t)$ of (3.9) is such that $\|p^*(t)\| = 0$, there is no direction which is a descent direction for all particles.*

Proof. We will propose that there is some direction d which is a direction of descent for all gradient samples, and draw a contradiction showing this cannot be the case. Let the chosen direction d be the opposite of some gradient combination such that $-d = a \in \mathscr{C}$. Assume that $a^T p^{(k)}(t) > 0$, $\forall k \in [1, K]$.

Since $(0,0)$ is one of the points in the set $\mathscr{C}$, we know there is some setting of

$\lambda_1, \dots, \lambda_K$ such that $\sum_k \lambda_k p^{(k)}(t) = (0,0)$. Call this point b. Therefore,

$$a^T b = \sum_k \lambda_k a^T p^{(k)}(t) = 0.$$

All λ_k are nonnegative and at least one must be positive. Without loss of generality, consider all terms where $\lambda_k > 0$; there will be at least one such term, and they will sum to 0. Therefore either $a^T p^{(k)}(t) = 0$ for all such terms, *or* at least one term satisfies $a^T p^{(k)}(t) < 0$. Thus, a contradiction: we cannot have $a^T p^{(k)}(t) > 0$, $\forall k \in [1,K]$. There is no choice of a which results in a descent direction for all particles.

$\square$

4.1.1 Contrasting Gradient Sampling and GSPF

We note that in GSPF, we do not have direct control over the step length s; instead, the step length is implicitly determined by how long $(t_{i+1} - t_i)$ the system evolves before the particle filter is again resampled. This choice implicitly replaces (3.10) with

$$x(t_{i+1}) = x(t_i) + (t_{i+1} - t_i)\left(\frac{p^*(t)}{\|p^*(t)\|_2}\right).$$

It is straightforward to modify the probability distribution in (3.6) to take this change into account. From a theoretical point of view, the goal of the Armijo line search which provided the step size in the basic algorithm was to ensure a sufficient descent condition, which is then used in the proof of convergence to Clarke ε-stationary points of the value function. Assuming that resampling occurs sufficiently often, it should be possible to ensure a similar sufficient descent condition in the GSPF, but that by itself will not rescue the convergence proof because we have also replaced (3.10) with (3.6). Whether or not the convergence theorem still holds, GSPF appears to converge to stationary points in practice.

4.2 Classifying and Resolving Stationary Points

Adapting gradient sampling to the case where the system state is estimated by a particle filter is straightforward, but we no longer have direct control over the sampling radius ε and so we must devise an alternative termination criterion. Furthermore, the gradient sampling algorithm converges to stationary points of any kind, but we seek a local minimum (which by construction (3.3) is guaranteed to be a global minimum and occur at the target set); consequently, we must devise a mechanism for escaping stationary points which are not local minima.

Fortunately, we do have *indirect* control over our sampling radius: We can perform more sensor updates (and resamplings) in the hope that the spread of the state estimate is reduced with the introduction of more observations. In Chapter 5 we simulate a robot which can choose to use either a low or high precision sensor, and which would prefer to use the low precision version whenever possible to conserve power. Similar multi-tiered sensing solutions may include cases where multiple sensors are available but not always deployed, or where the robot could re-orient a sensor with a limited field of view.

4.2.1 Stationary Points Classification

Before going to the trouble of gathering additional sensor readings, we should first determine whether a stationary point is a desirable minimum or an undesirable saddle point or maximum. To do so, we will locally approximate the value function as a quadratic

$$\bar{\psi}(x) = \tfrac{1}{2}(x - x_c)^T \mathrm{A}(x - x_c) + b^T(x - x_c) + c \tag{4.5}$$

in the neighborhood of the particles, where x_c is the center of curvature and matrix A is symmetric. Rather than fitting the value function directly, we fit the gradient of the quadratic approximation

$$\nabla\bar{\psi}(x) = \mathrm{A}(x - x_c) + b \tag{4.6}$$

to the set of gradient samples $\{p^{(k)}(t)\}$ that we have already collected (3.8). In our implementation, we set x_c to be the mean of the particle locations $\{x^{(k)}(t)\}$, use least squares to fit A and b, update $x_c = \mathrm{A}^{-1}b$, and refit A and b (at which point b is very close to zero). The resulting matrix A is an approximation of the Hessian of the value function in the neighbourhood of the stationary point.

We note in passing that quasi-Newton optimization algorithms, such as BFGS, also approximate the Hessian of the objective function [24]. Such algorithms are designed to construct their approximation efficiently in high dimensions using a single objective function sample from each of multiple steps. In our case we wish to approximate a low dimensional Hessian which takes into account information from all of the current particles and only the current step, so we find the least squares fit described above more efficient and appropriate than an adapted quasi-Newton update.

If the stationary point is a minimum, A will be positive definite. Algorithmically, we compute the eigenvalues of A (relatively inexpensive for the low to moderate dimensional systems in which we are interested) and declare victory if they are all positive. If there are negative eigenvalues, then we can attempt to improve the state estimate through additional sampling (as described above) and thereby escape the stationary point.

4.2.2 Resolution by Eigenvector

In practice, improved sensing is not always available or is insufficient to resolve undesirable stationary points. Fortunately, the eigenvalue decomposition of the Hessian A also provides us with an alternative method to determine a reasonable action. Each eigenvector v corresponding to a negative eigenvalue of A is locally a direction of descent for the value function. By proposition 2 there is no consensus direction of descent for all particles, but we can choose some direction from among these eigenvectors in the hope of escaping the stationary point.

If there is only a single negative eigenvalue (which must be the case at a saddle point in 2D) with corresponding eigenvector v, there are only two descent directions: v or $-v$. A simple voting procedure can determine which of these the majority of the particles prefer. Let

$$\alpha = \sum_{k=1}^{K} \operatorname{sign}(-v^T p^{(k)}).$$

If $\alpha < 0$ we travel in the direction of $-v$, otherwise we travel in the direction of $+v$. Figure 4.2 illustrates this procedure. If multiple eigenvalues of A are negative, one could use the simple voting procedure on the eigenvector associated with the most negative eigenvalue, or one could devise a more complex procedure for searching

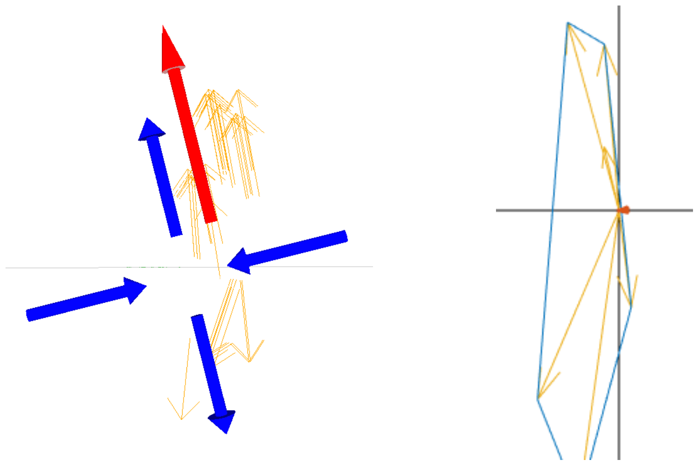

Figure 4.2: Using a quadratic approximation to resolve a saddle point. In both plots gradient samples from an area where there is a saddle point are shown in yellow. Left: Gradient vectors shown at their corresponding particle locations and the eigenvectors of the local Hessian approximation (blue). The vectors pointing inward correspond to a positive eigenvalue, while those pointing outward correspond to a negative eigenvalue. The action chosen (red) is upward, because more gradient samples agree with this sense of the eigenvector corresponding to the negative eigenvalue. Right: Gradient vectors shown in gradient space and their convex hull (blue). The convex hull contains the origin so there is no consensus direction.

over the space spanned by the eigenvectors associated with all of the negative eigenvalues to find a direction favorable to more of the particles.

4.2.3 Other Failures to Achieve Consensus

In practice, there are scenarios which do not achieve consensus, and yet are not straddling stationary points. An example is shown in the "Hallway Entrance" example of Section 5.3: near the entrance to a hallway, it is possible for gradients to span more than 180°—but there is no stationary point anywhere in this example. Thus, we cannot assume that lack of consensus signifies a stationary point.

Therefore, before we take action based on the quadratic approximation, we must determine if the results are sane. We do this by checking if the center of curvature of the quadratic approximation (x_c from equation 4.5) is within the convex hull of the particles. If the quadratic's center is not in the span of the particles, then it is likely not a stationary point as we were expecting. In this case, we do not use the eigenvalues of the quadratic to determine descent direction, and we must relocalize if we hope to continue.

Chapter 5

Experiments

To test GSPF in as realistic an environment as possible in silico, we use the MCL particle filter implementation found in ROS [30], and hook it to Gazebo [17] to simulate the robot's (noisy) motion and sensor systems. We have modified the original MATLAB code from [8] to accept the particle locations as the sample locations, and communicate with ROS through MATLAB's Engine API for C++. For convenience, we construct the value functions using a transformation of (3.3) to a time-dependent Hamilton-Jacobi PDE [25] that is easily solved using existing software [23, section 2.7]. Given the value function approximation, we approximate the gradients numerically at the nodes of the grid using (upwind) finite differences, and then (linearly) interpolate these approximate gradients to states which are not on the grid. The value function and gradient approximations are currently constructed offline.

In each of the following examples, we simulate a holonomic disc robot with a sweeping single-beam LIDAR range sensor. The robot can travel equally fast in any direction in the plane. The sensor has a 260° horizontal field of view. We can simulate either a low precision (noisier) or high precision version of the sensor. Unless otherwise specified, the high precision version of the sensor is used.

In all figures that depict paths taken by the robot, both the ground truth trajectory (green solid lines) and state estimate (blue stippled lines) are shown. The "state estimate" is the value returned by the MCL implementation, which is usually the mean of the particles' locations (precisely, it is the mean of the dominant cluster,

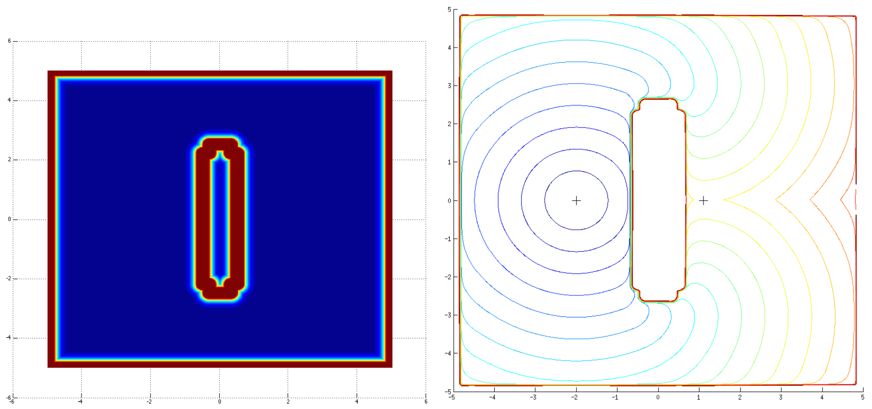

Figure 5.1: Single obstacle scenario. Left: Costmap, with costs ranging from low (blue) to high (red). Costs are highest near the outer boundary of the domain and near the obstacle. Right: Contours of the resulting value function. Stationary points are the minimum at the goal $(-2.0, 0.0)$ and the saddle point on the east side of the obstacle $(0.9, 0.0)$. The horizontal ridge running through this saddle point is the decision boundary between going north or south around the obstacle.

and most of the time there is only one cluster).

5.1 Single Obstacle

Our first example has only a single symmetric obstacle, and is used to demonstrate the main features of the GSPF. The robot must travel around the obstacle from east $(4.0, 0.2)$ to west $(-2.0, 0.0)$ while avoiding a saddle point. Figure 5.1 illustrates the scenario.

The robot starts by using the noisier sensor and an initial state estimate with large covariance. The optimal deterministic path goes northwest around the obstacle, but this cannot be determined from the noisy state estimate, which straddles the north/south decision boundary. The GSPF identifies that west is a consensus direction, so the robot moves that way. The state estimate improves, but not enough to resolve the choice between north and south; consequently, the system encounters the saddle point. As explained in section 4.2, an approximate local Hessian is constructed and the presence of positive and negative eigenvalues identifies the

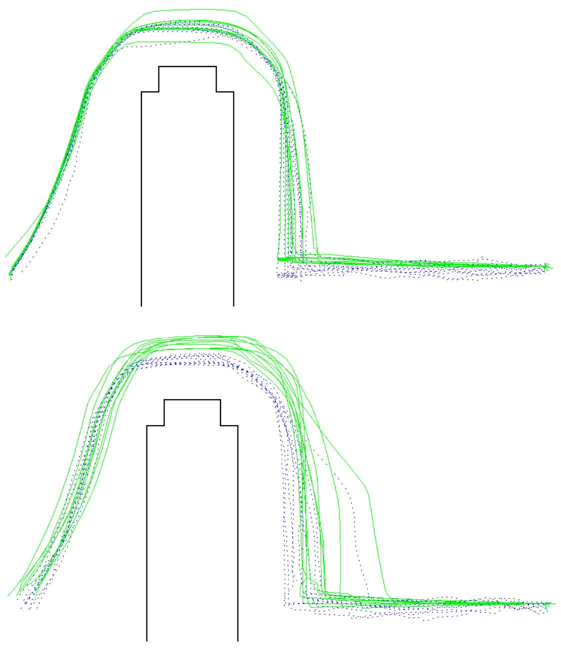
Figure 5.2: The single obstacle example using two different stationary point resolution methods (zoomed to show only the relevant portion of the domain). Top: Improved localization. Bottom: Eigenvector voting. Ten trials of each are shown.

stationary point as a saddle. Two approaches to resolve such undesirable stationary points were proposed, and we illustrate both in figure 5.2 and below.

To demonstrate improved localization, we switch to the high precision version of the range finder and perform MCL sensor corrections (and resampling). Figure 5.3 shows how the covariance of the state estimate drops in one of the trials as these improved corrections are incorporated, until finally a consensus direction emerges (when the particles all end up on the north side of the decision boundary) and the robot escapes the saddle.

For resolution by eigenvector, we keep the noisy sensor. When we identify the saddle point, we examine the eigenvectors of the approximate Hessian. The particles vote on the direction of the eigenvector associated with the negative eigenvalue with which they most agree. The majority votes to go north, allowing escape from the

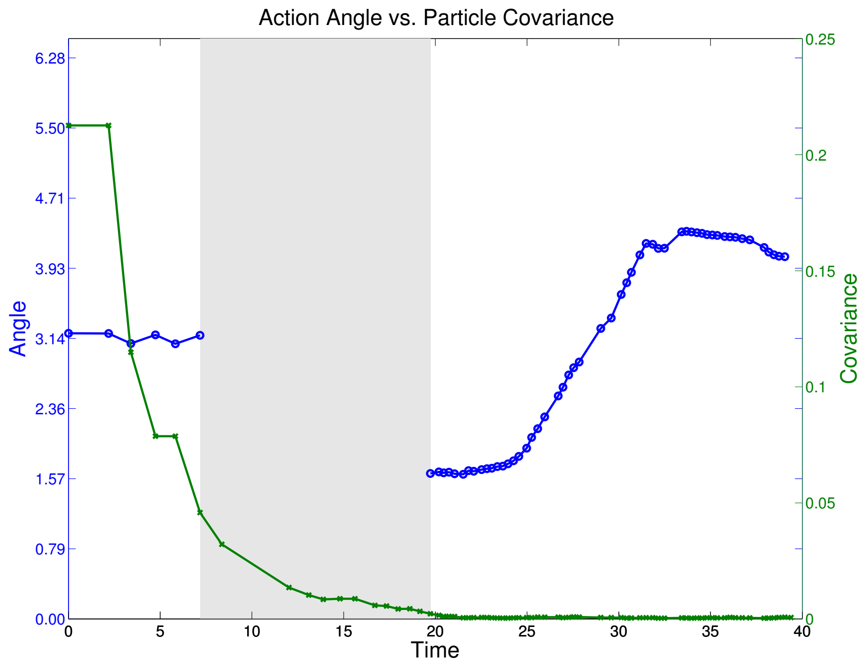

Figure 5.3: Covariance of the particle cloud and the consensus action direction as a function of time for a single run of the improved localization approach. The covariance of the state distribution decreases as we move west ($\theta \approx 3.14$ rad) until we get stuck on the saddle point (the grey shaded segment). Covariance further decreases as we take new observations from the improved sensor. We see a sudden change in direction ($\theta \approx 1.57$ rad) when the covariance drops sufficiently low that the particles no longer surround the saddle.

saddle.

Whichever resolution procedure is used, once the robot escapes the saddle point GSPF continues easily around the north of the obstacle and then southwest toward the goal. When the particles are in the goal region another stationary point is identified. A new approximate Hessian is constructed, its positive eigenvalues confirm that we have reached the target, and GSPF terminates.

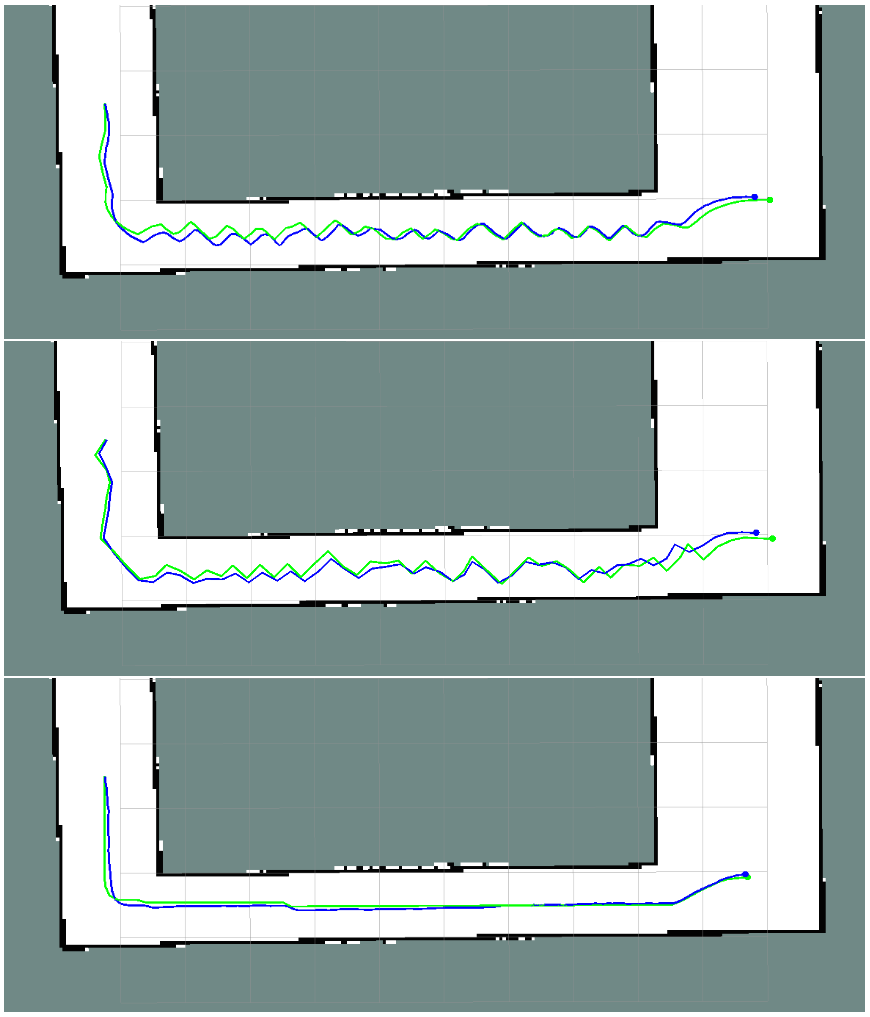

Figure 5.4: Navigating to a goal in the bottom right corner. In all cases, actions are chosen at roughly 0.05m intervals. Top: Action chosen by steepest descent on the expected state generates chattering. Middle: Action chosen by SGD generates a more randomized, but still jagged path. Bottom: Action chosen by GSPF generates a smoother path.

Table 5.1: Average Angular Difference Between Successive Actions

Update Rate	**Expected State**	SGD	GSPF
	mean (variance)	mean (variance)	mean (variance)
0.01 m	16.6° (0.05)	16.0° (0.08)	**2.37**° (0.02)
0.05 m	16.6° (0.21)	22.6° (0.29)	**0.54**° (0.00)
0.1 m	15.2° (0.04)	23.5° (0.79)	**0.39**° (0.00)
0.2 m	23.3° (0.15)	29.4° (2.76)	**0.01**° (0.00)

5.2 Narrow Hallway

We illustrated that the basic gradient sampling algorithm can resolve the chattering problem encountered when the value function is (nearly) non-differentiable in figure 1.2. In this section we illustrate that the same chattering behaviour can arise when using the default ROS approach of planning based on the expected state of the MCL filter, and that the chattering cannot be resolved simply by randomly sampling states.

The robot starts to the left of a narrow hallway, and must travel down the hallway to its goal on the bottom right. The cost function is chosen to penalize states which are too close to the walls, and the resulting value function displays a steep sided valley through the narrow hallway. Figure 5.4 illustrates a single run for each of the approaches. Choosing actions based on the expected state shows the same chattering behaviour as the fixed stepsize approach in figure 1.2, while GSPF shows a relatively smooth path.

We also include a comparison against actions chosen by Stochastic Gradient Descent (SGD) [4]. One might hypothesize that by taking a random sample from the state estimation, we might give all gradients equal contribution and offset opposing gradients. However, this does not introduce *consensus* among drastically opposing gradients; in fact, it can even exacerbate the problem by sampling from particles very close to the wall. Our results in Table 5.1 corroborate this and emphasize the need for consensus, not just equal contribution.

It is also possible that an improved path might be generated by adjusting the stepsize. To test that hypothesis we ran simulations at a variety of update rates and measured the angle between successive action choices:

$$\sum_i \left| \arctan\left(\frac{y(t_i)}{x(t_i)}\right) - \arctan\left(\frac{y(t_{i-1})}{x(t_{i-1})}\right) \right|, \quad \text{where } p^*(t) = [x(t), y(t)]^T. \tag{5.1}$$

We refer to this as the Angle Metric. Only the portion of each trajectory in the hallway was used (from $x = -4.0$ to $x = 3.0$). Results are shown in Table 5.1. The large heading changes characteristic of chattering persist for the steepest descent and SGD approaches over a wide range of update rates, while GSPF avoids the problem even at fast update rates. We were able to observe chattering by GSPF if we stringently limited the number of particles (around twenty or fewer), but such a small number is not enough to localize reliably anyway.

5.3 Hallway Entrance

One of the convenient advantages of our technique is the ability to halt the robot in the abscence of consensus. This often happens in situations where the robot is at risk of collision if it proceeds without taking uncertainty into account. Take, for instance, the situation illustrated in Figure 1.3. The mean state would command the robot to proceed directly east to enter the hallway. However, many of the particles are not centered on the entrance or are closer to the wall. If the true robot state is more accurately represented by one of these outliers, then the command by expected state will take it straight into collision. The robot would benefit from instead pausing and attaining better localization before proceeding.

We reconstructed this scenario in a ROS/Gazebo simulation to measure the potential safety improvements. The robot is initialized just outside the entrance to a narrow hallway, with a state estimate sampled from a 2D Gaussian with $\sigma_{xx} = \sigma_{yy} = 0.1$ meters. The correct direction for the robot to move is directly east, but some particles would quickly be in collision if they were commanded to move in that direction. From the knowledge available to the robot, no consensus can be reached.

This kind of scenario was touched upon in Section 4.2.3. We are not near a stationary point, and yet we are unable to reach consensus. Therefore GSPF will

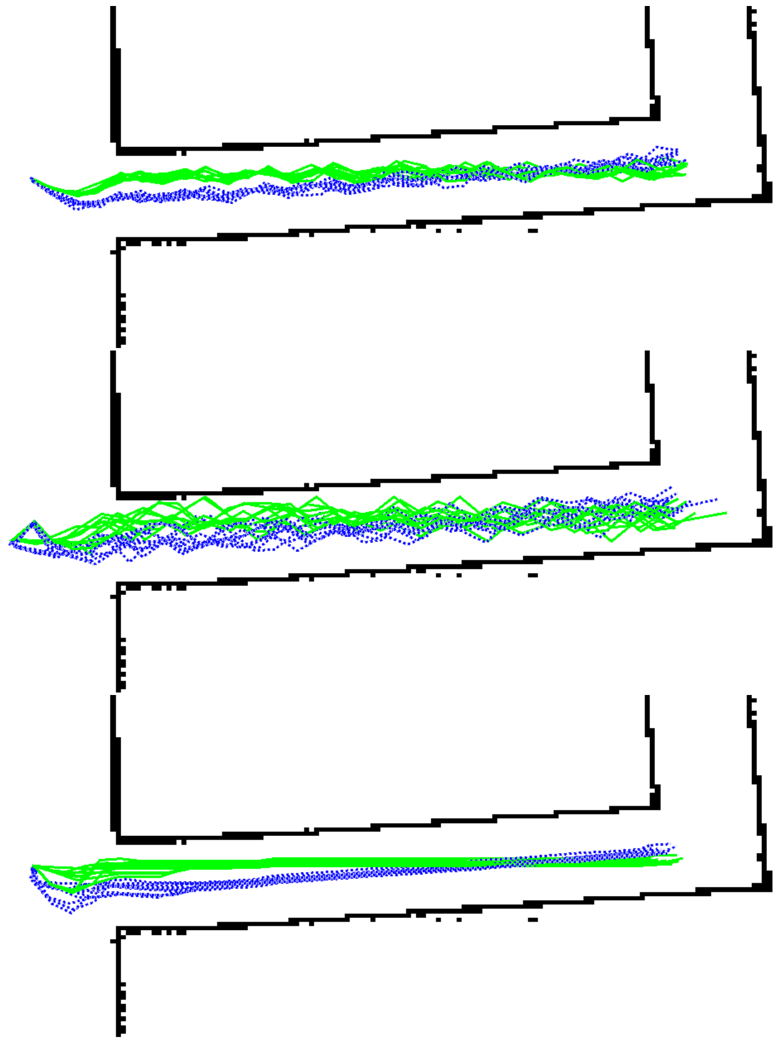

Figure 5.5: Navigating the entrance to a narrow hallway. Compare to figure 5.4. Not depicted is how many particles are in collision; we only show the expected state here. Top: Action chosen by steepest descent on the expected state generates chattering. Middle: Action chosen by SGD generates an even less stable path. Bottom: Action chosen by GSPF generates a smoother path.

Table 5.2: Angular Change and Safety Metrics

Statistic	**Expected State** mean (variance)	**SGD** mean (variance)	**GSPF** mean (variance)
Angle Change	22.2° (0.3)	42.7° (3.7)	**9.3**° (2.1)
Probability of Collision	15.2% (37.4)	46.6% (530.0)	**2.9%** (3.8)
Mean Particle Cost	66.0 (0.4)	67.9 (3.6)	**59.7** (1.2)

activate its high-fidelity sensor and attempt relocalization until consensus is reached. We compare this to the alternative approaches using the expected state estimate. Trials for all three methods are depicted in Figure 5.5.

Table 5.2 shows the comparison between our three different control schemes on both the Angle Metric (Equation 5.1) and a new metric designed to measure safety. The Probability of Collision Metric is given by the maximum of the fraction of particles currently in collision at any time during a single trial:

$$\max_i (c_i/K), \tag{5.2}$$

where c_i is the number of particles in collision at time i. We also report the mean particle cost over the whole trial, which is a proxy to how "close" the particles are to being in collision (a cost of 99 or 100 is in collision; a cost of 0 is ≥ 5.75 meters from an obstacle; and costs 1–98 scale exponentially with their proximity to an obstacle). All metrics are averaged over 10 trials. GSPF shows the best performance on all three metrics.

This demonstrates the natural affect that GSPF has on safety without any additional modifications. However, this is not a fair comparison to methods which explicitly consider safety. For example, one way to mitigate the risk of the traditional method would be to simulate the motion of each particle before issuing the command, and perform relocalization if this action would bring any particle into collision. We could then compare this technique to our own relocalization technique to see if they are comparable safety measures. We leave this as future work in exploring the safety aspects of GSPF.

5.4 Cluttered Scene

Here we examine a slightly more complex scene with several obstacles of different shapes, as illustrated in figure 5.6. The function exhibits many saddle points (typically one near each separate obstacle) and ridges that represent decision surfaces.

Trials using action selection based on expected state and GSPF are shown in figure 5.7. Eigenvector voting procedures were used to resolve all stationary points. We observe several differences in the paths generated by the two algorithms. First, GSPF does not make an immediate decision on which side of the first obstacle to pass, and sometimes eventually chooses to pass to the west. Second, GSPF exhibits a tendency to stay further away from obstacles. In both cases these behaviours arise because GSPF takes into account the position (and hence optimal action) of all particles.

The second behaviour can be considered an appropriate response to a potential collision (if the true state turns out to be the particle which is near the obstacle). The first, however, may generate paths which are worse than those which immediately choose a route around the obstacle, even if the route turns out to be slightly suboptimal for the true state. This behaviour occurs along ridges in the value function, so a potential solution is to approximate the Hessian at every step and examine its eigenvalues. The presence of a (sufficiently) negative eigenvalue indicates that the particle filter is straddling a ridge in the value function (and is therefore likely headed to a saddle point anyway), so we could immediately apply one of the saddle point resolution approaches from section 4.2 to choose a better direction.

5.5 Other Planners: RRT*

We have so far assumed that actions are chosen based on the (approximated) value function as discussed in section 3.2; however, there is no reason the gradient sampling approach cannot be combined with other algorithms that generate action choices. Any planner which can quickly evaluate action choices for a set of sample states is suitable, although the stationary point procedures from section 4.2 construct an approximate Hessian and so assume that actions are consistent with an underlying (but potentially unknown) value function. In this final example we demonstrate the use of GSPF with a multi-query version of the RRT*, a convergent optimal planner

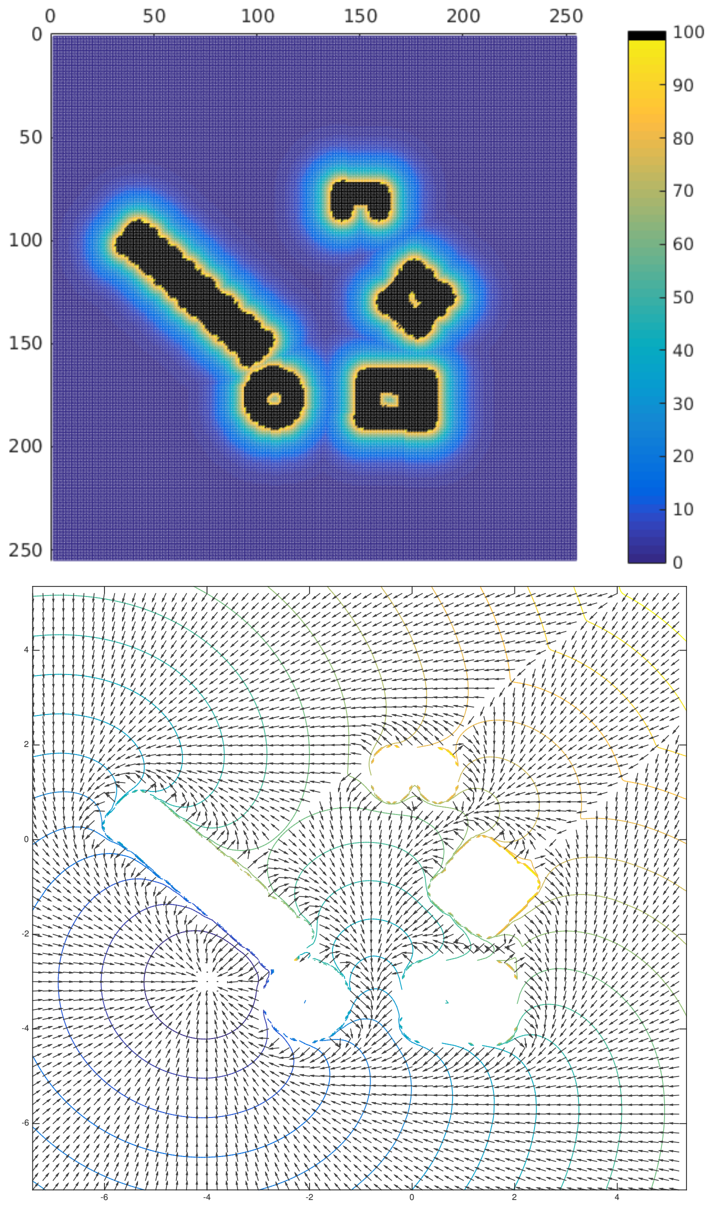

Figure 5.6: The cluttered scenario. Top: Costmap, with costs ranging from low (blue) to high (yellow). Black indicates (inflated) obstacles. Obstacles are inflated by the robot radius (0.2 m) to prevent collision. Costs extend out to 6.0 m from the obstacles. Bottom: The value function for the cluttered scene. The goal location is in the bottom left. Contours of the value function and the resulting gradient vector field are shown.

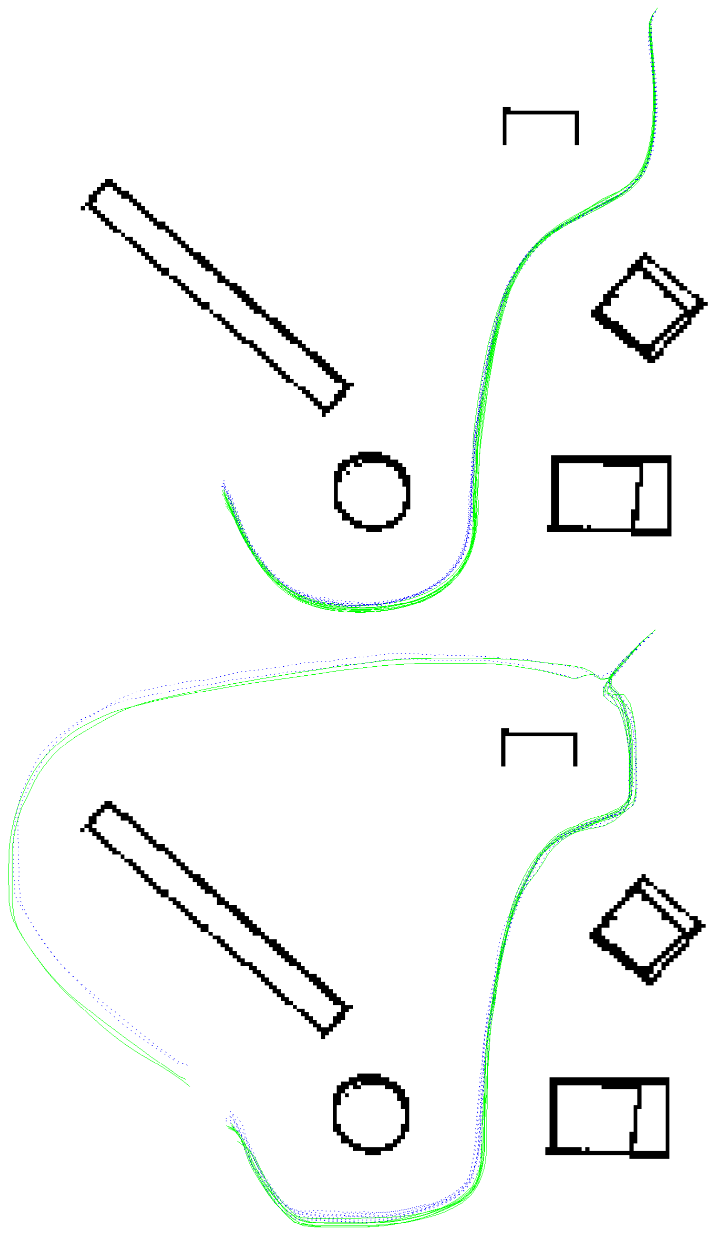

Figure 5.7: Navigation with value function planner through the cluttered scene. The robot travels from the upper right to the lower left. Obstacles are shown in black. Ten trials are shown in each case. Top: Action chosen based on expected state. Bottom: Action chosen by GSPF.

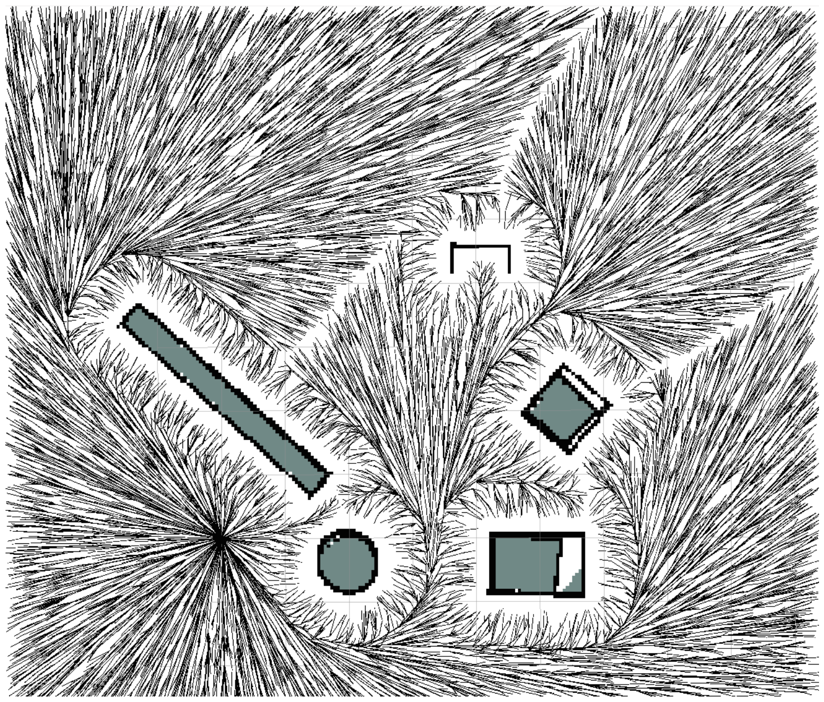

Figure 5.8: The result of running RRT* on the cluttered scene. Compare to the vector field given by the value function in figure 5.6.

based on Rapidly Exploring Random Trees [15].

Figure 5.8 shows the result of generating an RRT* plan starting from the goal state. This tree acts as our navigation function: at any state, it can return the (approximately) optimal action from the RRT node which is the nearest neighbor to that state (in this case it returns a 2D vector toward the goal, scaled to the maximum speed of the robot). This experiment is performed on the same cluttered scenario as the previous section. We start the robot with a slightly different initial state to force it to start on the decision boundary for which way it should pass the first obstacle, as this boundary is slightly different according to RRT*.

Figure 5.9 illustrates the performance of GSPF under this type of navigation function. GSPF behaves very similar to the way it behaved with the value function. The results are also very similar to the typical planning by expected state, but the expected state approach had a hard time satisfying the RRT* termination criterion

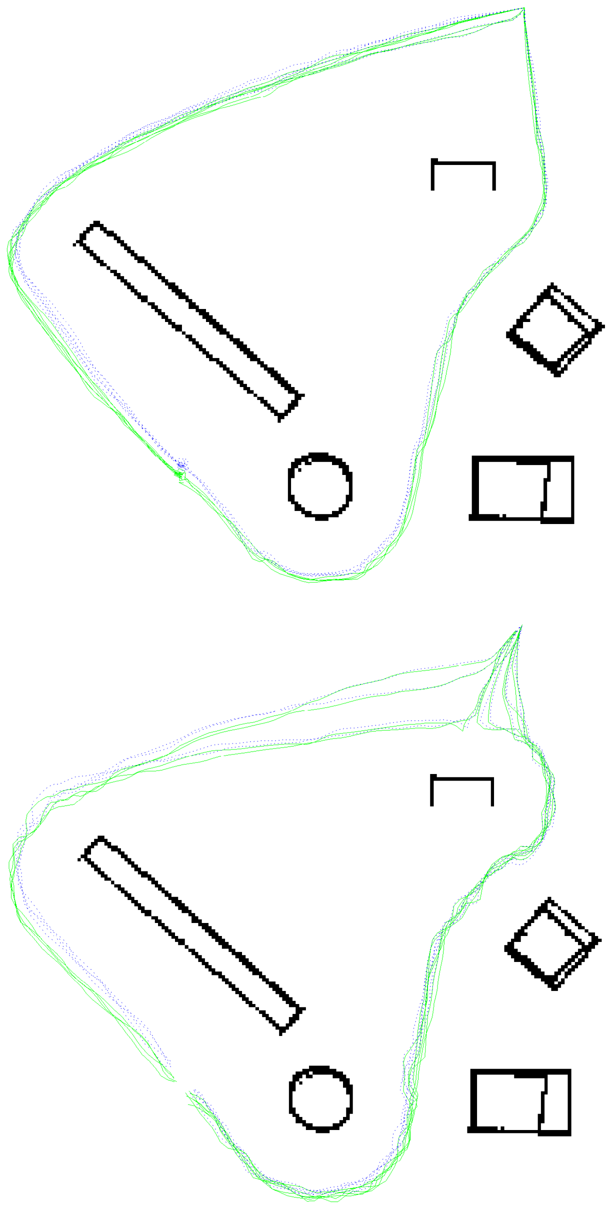

Figure 5.9: Navigation with RRT* planner through the cluttered scene. The robot travels from the upper right to the lower left. Obstacles are shown in black. Ten trials are shown in each case. Top: Action chosen based on expected state. Bottom: Action chosen by GSPF.

and hence tended to chatter around the goal location. GSPF's termination criterion naturally takes into account the distribution of the particles and hence it did not suffer from this issue.

Chapter 6

Conclusion

We observed that the gradient sampling algorithm from [8] can be used to resolve the chattering problem commonly encountered when generating optimal paths from value function approximations. We then proposed the GSPF algorithm, which uses the particles as the gradient sample locations, thereby naturally and efficiently generating consensus directions suitable for all particles or detecting that no such consensus can be reached. When no consensus exists we use the eigenvalues of an approximate Hessian to diagnose whether we have arrived at the goal or are stuck on an undesirable stationary point; in the latter case two approaches were described for finding a descent direction. The scheme was illustrated on three examples in the ROS / Gazebo simulation environment. GSPF was also briefly demonstrated using an RRT* planner instead of a value function approximation. Although not illustrated, it is also straightforward to apply the gradient sampling approach to systems whose state uncertainty is characterized by parametric representations, such as the Kalman filter.

In the future we intend to further explore the use of GSPF with other planners and its extension to anisotropic and non-holonomic dynamics. The use of a faster, multi-query planner will bring us closer to robust operation in bustling hospital settings. And the implementation atop a non-holonomic differential drive system will constitute a crucial step toward our goal of developing a safe, comfortable robotic wheelchair.

Bibliography

[1] A.-A. Agha-Mohammadi, S. Chakravorty, and N. M. Amato. FIRM: Sampling-based feedback motion planning under motion uncertainty and imperfect measurements. *The International Journal of Robotics Research*, 33 (2):268–304, 2014. → pages 11

[2] H. Bai, D. Hsu, and W. S. Lee. Integrated perception and planning in the continuous space: A POMDP approach. *The International Journal of Robotics Research*, 33(9):1288–1302, 2014. → pages 9, 10, 12

[3] D. P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 1. → pages 14

[4] L. Bottou. Online algorithms and stochastic approximations. In D. Saad, editor, *Online Learning and Neural Networks*. Cambridge University Press, Cambridge, UK, 1998. → pages 35

[5] A. Bry and N. Roy. Rapidly-exploring random belief trees for motion planning under uncertainty. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 723–730. IEEE, 2011. → pages 11, 12

[6] W. Burgard, A. B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. The interactive museum tour-guide robot. In *Proceedings of the 15th AAAI Conference on Artificial Intelligence*, pages 11–18, 1998. → pages 2

[7] J. V. Burke, A. S. Lewis, and M. L. Overton. Approximating subdifferentials by random sampling of gradients. *Mathematics of Operations Research*, 27 (3):567–584, 2002. → pages 18

[8] J. V. Burke, A. S. Lewis, and M. L. Overton. A robust gradient sampling algorithm for nonsmooth, nonconvex optimization. *SIAM Journal on Optimization*, 15(3):751–779, 2005. → pages 7, 17, 19, 30, 45

[9] A. Censi, D. Calisi, A. De Luca, and G. Oriolo. A Bayesian framework for optimal motion planning with uncertainty. In *Robotics and Automation (ICRA), 2008 IEEE International Conference on*, pages 1798–1805. IEEE, 2008. → pages 11

[10] F. H. Clarke. *Optimization and Nonsmooth Analysis*. SIAM, 1990. → pages 18

[11] E. Cristiani and M. Falcone. Fast semi-Lagrangian schemes for the Eikonal equation and applications. *SIAM Journal on Numerical Analysis*, 45(5): 1979–2011, 2007. → pages 16

[12] N. E. Du Toit and J. W. Burdick. Robot motion planning in dynamic, uncertain environments. *IEEE Transactions on Robotics*, 28(1):101–115, 2012. → pages 11

[13] S. Ferguson, B. Luders, R. C. Grande, and J. P. How. Real-time predictive modeling and robust avoidance of pedestrians with uncertain, changing intentions. In *Algorithmic Foundations of Robotics XI*, volume 107, pages 161–177. Springer, 2015. → pages 9

[14] T. Gomi and A. Griffith. Developing intelligent wheelchairs for the handicapped. In *Assistive Technology and Artificial Intelligence*, pages 150–178. Springer, 1998. → pages 2

[15] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011. → pages 42

[16] R. Kimmel and J. A. Sethian. Optimal algorithm for shape from shading and path planning. *Journal of Mathematical Imaging and Vision*, 14(3):237–244, 2001. → pages 16

[17] N. Koenig and A. Howard. Design and use paradigms for Gazebo, an open-source multi-robot simulator. In *Intelligent Robots and Systems (IROS), 2004 IEEE/RSJ International Conference on*, volume 3, pages 2149–2154. IEEE, 2004. → pages 30

[18] S. M. LaValle and R. Sharma. On motion planning in changing, partially predictable environments. *The International Journal of Robotics Research*, 16 (6):775–805, 1997. → pages 8

[19] F. Li, C.-W. Shu, Y.-T. Zhang, and H. Zhao. A second order discontinuous Galerkin fast sweeping method for Eikonal equations. *Journal of Computational Physics*, 227(17):8191–8208, 2008. → pages 16

[20] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling. Learning policies for partially observable environments: Scaling up. In *Proceedings of International Conference on Machine Learning*, 1995. → pages 12

[21] B. D. Luders and J. P. How. An optimizing sampling-based motion planner with guaranteed robustness to bounded uncertainty. In *Proceedings of the American Control Conference (ACC)*, pages 771–777. IEEE, 2014. → pages 11, 12

[22] B. D. Luders, S. Karaman, and J. P. How. Robust sampling-based motion planning with asymptotic optimality guarantees. In *AIAA Guidance, Navigation, and Control Conference (GNC)*, 2013. → pages 11

[23] I. M. Mitchell. A toolbox of level set methods (version 1.1). *Department of Computer Science, University of British Columbia, Vancouver, BC, Canada, Tech. Rep. TR-2007-11*, June 2004. → pages 30

[24] J. Nocedal and S. J. Wright. Numerical optimization. *Springer Science*, 35 (67-68):7, 1999. → pages 27

[25] S. Osher. A level set formulation for the solution of the Dirichlet problem for Hamilton-Jacobi equations. *SIAM Journal on Mathematical Analysis*, 24(5): 1145–1152, 1993. → pages 30

[26] C. H. Papadimitriou and J. N. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987. → pages 10

[27] S. Patil, G. Kahn, M. Laskey, J. Schulman, K. Goldberg, and P. Abbeel. Scaling up Gaussian belief space planning through covariance-free trajectory optimization and automatic differentiation. In *Algorithmic Foundations of Robotics XI*, volume 107, pages 515–533. Springer, 2015. → pages 11, 12

[28] J. Pineau, G. Gordon, S. Thrun, et al. Point-based value iteration: An anytime algorithm for POMDPs. In *International Joint Conference on Artificial Intelligence*, volume 3, pages 1025–1032, 2003. → pages 9, 10, 12

[29] S. Prentice and N. Roy. The belief roadmap: Efficient planning in belief space by factoring the covariance. *The International Journal of Robotics Research*, 2009. → pages 11

[30] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. ROS: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5, 2009. → pages 30

[31] S. Ross, J. Pineau, S. Paquet, and B. Chaib-Draa. Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Research*, 32:663–704, 2008. → pages 9, 12

[32] K. M. Seiler, H. Kurniawati, and S. P. Singh. An online and approximate solver for POMDPs with continuous action space. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 2290–2297. IEEE, 2015. → pages 11

[33] D. Silver and J. Veness. Monte-Carlo planning in large POMDPs. In *Advances in Neural Information Processing Systems*, pages 2164–2172, 2010. → pages 9, 10

[34] W. Sun, S. Patil, and R. Alterovitz. High-frequency replanning under uncertainty using parallel sampling-based motion planning. *IEEE Transactions on Robotics*, 31(1):104–116, 2015. → pages 10, 11

[35] W. Sun, J. van den Berg, and R. Alterovitz. Stochastic extended LQR: Optimization-based motion planning under uncertainty. In *Algorithmic Foundations of Robotics XI*, volume 107, pages 609–626. Springer, 2015. → pages 11

[36] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005. → pages 17

[37] N. Traft and I. M. Mitchell. Improved action and path synthesis using gradient sampling. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pages 6016–6023. IEEE, 2016. → pages iv

[38] P. Trautman and A. Krause. Unfreezing the robot: Navigation in dense, interacting crowds. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 797–803. IEEE, 2010. → pages 9

[39] J. N. Tsitsiklis. Efficient algorithms for globally optimal trajectories. *IEEE Transactions on Automatic Control*, 40(9):1528–1538, 1995. → pages 15, 16

[40] J. Van Den Berg, P. Abbeel, and K. Goldberg. LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information. *The International Journal of Robotics Research*, 30(7):895–913, 2011. → pages 11

[41] J. Van Den Berg, S. Patil, and R. Alterovitz. Motion planning under uncertainty using iterative local optimization in belief space. *The International Journal of Robotics Research*, 31(11):1263–1278, 2012. → pages 11, 12

[42] M. P. Vitus and C. J. Tomlin. A hybrid method for chance constrained control in uncertain environments. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 2177–2182. IEEE, 2012. → pages 8