Improved Action and Path Synthesis using Gradient Sampling

Neil Traft and Ian M. Mitchell

***Abstract*— Shortest paths generated through gradient descent on a value function have a tendency to chatter and/or require an unreasonable number of steps to synthesize. We demonstrate that the gradient sampling algorithm of [Burke, Lewis & Overton, 2005] can largely alleviate this problem. For systems subject to state uncertainty whose state estimate is tracked using a particle filter, 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 using the ROS and Gazebo robot simulation environment.**

I. Introduction

Many techniques have been developed for synthesizing (approximately) optimal feedback control inputs/actions in the control and robotics literature. When dynamics are nonlinear, inputs are constrained, or cost functions are not quadratic, the optimal feedback controls are often nonsmooth. In this paper we propose a simple solution to two commonly encountered problems that arise when synthesizing control inputs (and the paths that they generate) from such control functions.

In order to illustrate these two problems, consider the narrow corridor scenario shown in Figure 1. For simplicity, we work with an isotropic, holonomic vehicle in the 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 which are close to the walls. To solve the scenario, we approximate a minimum time to reach value function in the obstacle free space. 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 at each step: Chattering of the optimal control as illustrated in Figure 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 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

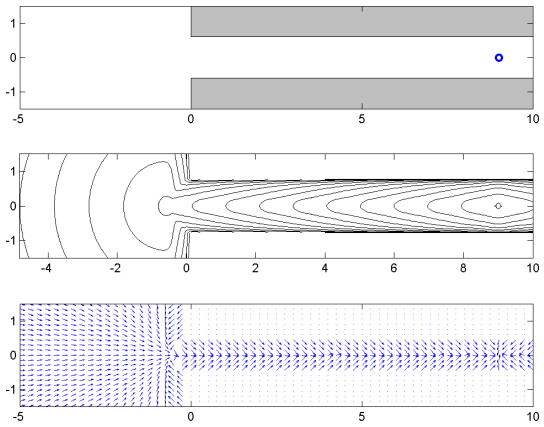

Fig. 1. Narrow corridor example. Top: Obstacles are gray, the goal location is a blue circle. Middle: Contours of the value function. The corresponding cost function penalizes states near the walls. Bottom: Vector field of the gradients of the value function (on a subsampled grid for visibility). All gradients in the corridor have a rightward component, but there is an abrupt jump between upward and downward components.

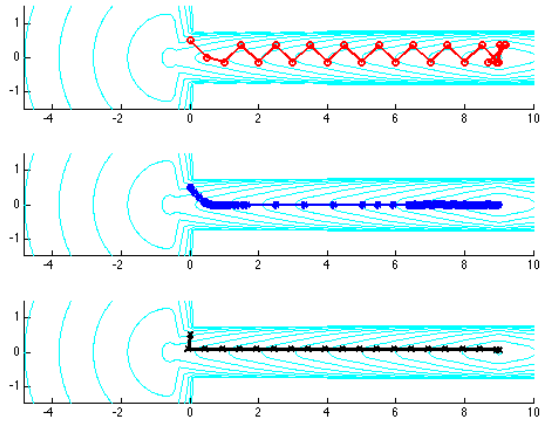

Fig. 2. Narrow corridor example paths. Every figure is overlayed with contours of the value function, and markers are placed along the trajectory at each step. Top: Fixed stepsize path. Middle: Adaptive stepsize path from MATLAB's `ode15s` (an implicit scheme). Bottom: Sampled gradient path.

This research was supported by CANWHEEL (the Canadian Institutes of Health Research (CIHR) Emerging Team in Wheeled Mobility for Older Adults Grant #AMG-100925) and National Science and Engineering Council of Canada (NSERC) Discovery Grant #298211.

Department of Computer Science, University of British Columbia, Vancouver, Canada. Email: `ntraft@gmail.com`, `mitchell@cs.ubc.ca`

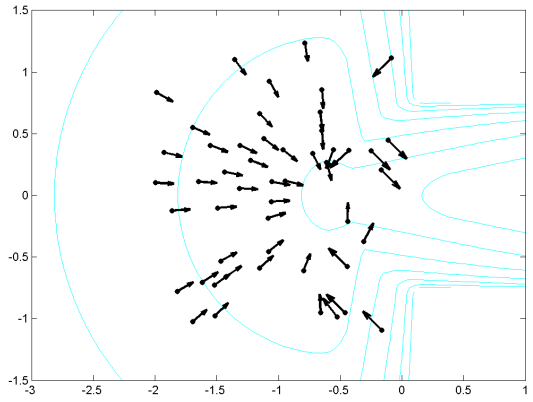

Fig. 3. State uncertainty, as represented by samples from a (simulated) particle filter. The figure shows a zoomed in view of the left side of the corridor scenario from Figure 1. Each dot is a state sample, and the attached arrow shows the optimal action for that sample. Depending on which sample represents the actual state, the optimal action could be in any direction.

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. Figure 2 also illustrates how our proposed approach generates a much more desirable solution with large stepsize and straight path.

The second problem arises in physical systems in which the state is not accurately known. Figure 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 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 counterproductive. In fact, in this case the system should, if possible, refine its state estimate before choosing an action.

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

- How the gradient sampling algorithm [1] 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 success in a simple simulation with Figure 2, but more importantly we illustrate success in a full scale robotic simulation with state uncertainty and simulated noisy sensors and motion using the widely adopted 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.

II. 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, 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 section we outline the elements of a typical optimal navigation pipeline, with actions chosen by gradient descent of a value function approximation and uncertainty tracked by a particle filter representation. The section concludes with a brief description of the gradient sampling algorithm from the non-smooth optimization literature.

A. Problem Formulation

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

$$\psi(x_0) = \inf_{x(\cdot)} \int_{t_0}^{t_f} c(x(s))\, ds, \tag{1}$$

where $x(0) = x_0$, $x(t_f) \in \mathcal{T}$ and $x(\cdot)$ are drawn from the set of feasible paths such that $x(t) \in \Omega \setminus \mathcal{T}$ for $t_0 \leq t < t_f$. The value function $\psi(x)$ measures the minimum cost to go from state x to $\mathcal{T}$ (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 (1) depends on the feasible paths. Here we will assume only the simplest form of isotropic holonomic dynamics

$$\tfrac{d}{dt} x(t) = \dot{x}(t) = u(t), \tag{2}$$

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.

B. Planning with a Value Function

The value function (1) satisfies a dynamic programming principle and can be shown to be the viscosity solution of the Eikonal equation (for example, see [2])

$$\begin{aligned} \|\nabla\psi(x)\| &= c(x), && \text{for } x \in \Omega \setminus \mathcal{T}; \\ \psi(x) &= 0, && \text{for } x \in \mathcal{T}. \end{aligned} \tag{3}$$

Under the dynamics (2), the viscosity solution $\psi(x)$ of (3) is continuous and almost everywhere differentiable. The only local minima of $\psi(x)$ occur at $\mathcal{T}$, but the function will usually be non-convex: it can have saddle points in $\Omega \setminus \mathcal{T}$ and local maxima at boundaries of Ω which are not also boundaries of $\mathcal{T}$. 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) analytically; however, there exist many efficient approaches to approximate it; for example [2]–[5]. 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

$$u^*(x) = \frac{\nabla\psi(x)}{\|\nabla\psi(x)\|}. \tag{4}$$

Consequently, generation of an optimal path is equivalent to gradient descent of $\psi(x)$. Of course we typically cannot represent the exact solution of (2) and (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

$$\begin{aligned} t_{i+1} &= t_i + \Delta t, \\ x(t_{i+1}) &= x(t_i) + \Delta t u^*(x(t_i)). \end{aligned} \tag{5}$$

Unfortunately, a straightforward implementation of (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 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.

C. 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) [6]. 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

$$x^{(k)}(t_{i+1}) \sim p(x(t_{i+1}) \mid x^{(k)}(t_i), u(t_i)), \tag{6}$$

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 (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.

D. 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 [1], 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 samples the gradient at points within a radius ϵ of the current point $x(t)$

$$x^{(k)}(t) = x(t) + \epsilon\,\delta x^{(k)}, \tag{7}$$

$$p^{(k)}(t) = \nabla\psi(x^{(k)}(t)) \tag{8}$$

for $k = 1, \ldots, K$, where $\{\delta x^{(k)}\}$ are sampled independently and uniformly from the unit ball centered on $x(t)$. Next, the algorithm approximates the Clarke subdifferential [7] using the convex combination of the gradient samples

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

The point in this set with minimum norm

$$p^*(t) = \underset{p \in P(t)}{\text{argmin}}\, \|p\|^2 \tag{9}$$

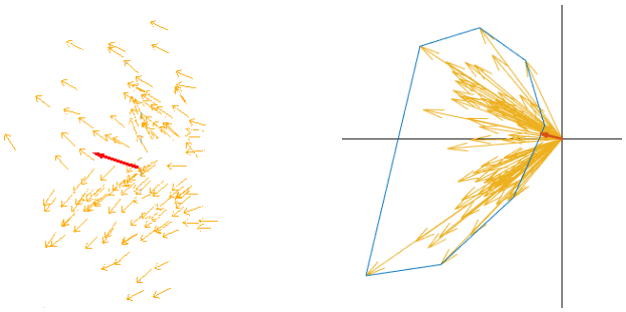

Fig. 4. Finding a "consensus" among different actions. Left: Gradient vectors (yellow) shown at the corresponding samples' locations, and the resulting consensus action (red). In this neighbourhood, the value function has a ridge. Right: The gradients (yellow) plotted in gradient space, their convex hull (blue) and $p^*(t)$ (red). The consensus action is a leftward movement, which is a descent direction for all samples.

is a consensus direction: a direction of descent from all samples $x^{(k)}(t)$. The direction $p^*(t)$ is the solution of a simple convex quadratic optimization problem. The technique is illustrated in Figure 4.

If $\|p^*(t)\| = 0$, there is a Clarke ϵ-stationary point—conceptually a local minimum, maximum or saddle point—somewhere within the ϵ-ball about $x(t)$, and no direction can be agreed upon by all samples. In this case the radius ϵ is reduced and a new sample set (7)–(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

$$x(t_{i+1}) = x(t_i) - s\frac{p^*(t)}{\|p^*(t)\|} \tag{10}$$

The algorithm terminates when ϵ shrinks to a predetermined threshold. When paired with a linesearch procedure that ensures sufficient descent, it is shown to converge to a Clarke stationary point under suitable conditions. The bottom of Figure 2 shows that the resulting optimal trajectory approximation is accurate and uses a reasonable stepsize throughout.

III. Other Related Work

The algorithm described below accounts for the state uncertainty in the particle filter by seeking a consensus direction for the current action. It does not attempt full treatment of the problem of planning under uncertainty, which has been tackled from many angles, for example: differential games and the Hamilton-Jacobi-Isaacs equation [8]; robust model predictive control (see [9] for an extensive list of citations); various versions of asymptotically optimal Rapidly-exploring Random Trees (RRT*) such as [10], [11]; and algorithms for efficient solution of very large Partially Observable Markov Decision Processes (POMDPs) [12] and POMDPs for continuous state spaces [13].

In fact, the algorithm does something very similar to QMDP [14]: The action chosen for each particle is optimal assuming that further evolution is deterministic, so the consensus direction likewise incorporates the assumption of deterministic future evolution. However, while QMDP selects the action among a discrete set with minimum expected cost, our choice is the minimum norm (9), motivated by the convergence proof in [1] and the ease of computing this value.

The algorithm described below was intended as a lightweight addition to a standard pipeline which separately applies a deterministic planner and online state estimation. We suspect that the more comprehensive treatments of planning under uncertainty would produce better results, albeit with additional implementation and computational effort. In future work we plan to quantify those differences experimentally.

IV. Merging the Methods

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 (2) are accurate, and the chosen input (4) is accurately implemented. Unfortunately, for most physical systems these assumptions do not hold. In this section we consider how the algorithm can be adapted to the case where $x(t)$ can only be estimated.

A. 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 (7), use the particles' locations (6) directly. We call this version Gradient Sampling with Particle Filter (GSPF).

Proposition 1: If the solution $p^*(t)$ of (9) is such that $\|p^*(t)\| \neq 0$, then $p^*(t)$ is a descent direction in the value function for all particles. If $\|p^*(t)\| = 0$, there is no direction which is a descent direction for all particles.

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 (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 (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 (10) with (6). Whether or not the convergence theorem still holds, GSPF still appears to converge to stationary points in practice.

B. 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) 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 Section V 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.

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

$$\bar{\psi}(x) = \tfrac{1}{2}(x - x_c)^T \mathrm{A}(x - x_c) + b^T(x - x_c) + c \qquad (11)$$

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

$$\nabla\bar{\psi}(x) = \mathrm{A}(x - x_c) + b \qquad (12)$$

to the set of gradient samples $\{p^{(k)}(t)\}$ that we have already collected (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. Such algorithms are designed to construct their approximation efficiently in high dimensions over multiple steps using only a single objective function sample at each step. 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 traditional sampling (as described above) and thereby escape the stationary point.

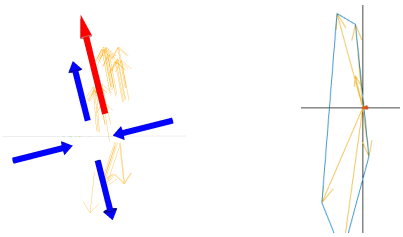

Fig. 5. Using a quadratic approximation to resolve a saddle point. In both plots gradient samples from an area where there is a saddle point are shown in yellow. Left: Gradient vectors shown at their corresponding particle locations and the eigenvectors of the local Hessian approximation (blue). The 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.

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 1 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

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

If $\alpha < 0$ we travel in the direction of $-v$, otherwise we travel in the direction of $+v$. Figure 5 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 over the space spanned by the eigenvectors associated with all of the negative eigenvalues to find a direction favorable to more of the particles.

V. Examples

To test GSPF in as realistic an environment as possible in silico, we use the MCL particle filter implementation found in ROS [15], and hook it to Gazebo [16] to simulate the robot's (noisy) motion and sensor systems. We have modified the original MATLAB code from [1] 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) to a time-dependent Hamilton-Jacobi PDE [17] that is easily solved using our own software [18, section 2.7]. Given the

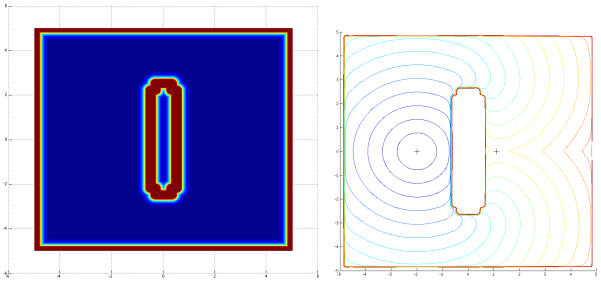

Fig. 6. Single obstacle scenario. Left: Costmap ranging from low (blue) to high (red). Right: Contours of the resulting value function. Stationary points are the minimum at the goal $(-2.0, 0.0)$ and the saddle point on the east side of the obstacle $(0.9, 0.0)$. The horizontal ridge running through this saddle point is the decision boundary between going north or south around the obstacle.

value function approximation, we approximate the gradients numerically at the nodes of the grid using (upwind) finite differences, and then interpolate these approximate gradients to states which are not on the grid. The value function and gradient approximations on the grid are currently constructed offline, while the interpolation is done online.

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.

A. *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 6 illustrates the scenario.

The robot starts using the noisier sensor and an initial state estimate with large covariance. The robot does not know that the optimal deterministic path goes northwest around the obstacle because the noisy state estimate 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 IV-B, an approximate local Hessian is constructed and the presence of positive and negative eigenvalues identifies the stationary point as a saddle. We illustrate the two approaches to resolve such stationary points in Figure 7 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 8 shows how the

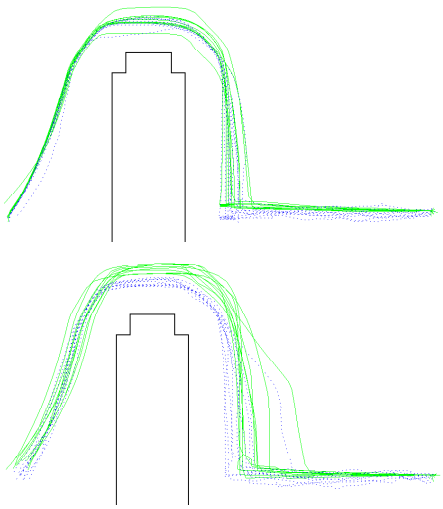

Fig. 7. The single obstacle example using two different stationary point resolution methods (zoomed to show only the relevant portion of the domain). Top: Improved localization. Bottom: Eigenvector voting. Ten trials of each are shown.

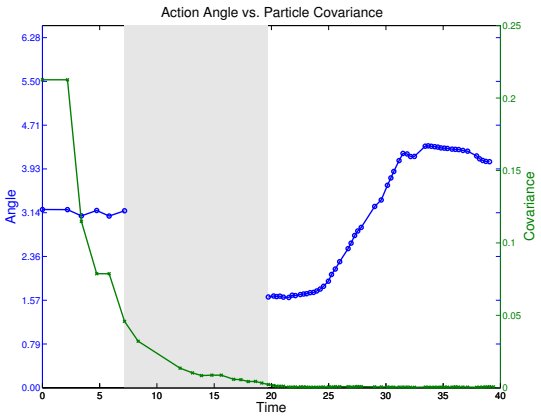

Fig. 8. Covariance of the particle cloud and the consensus action direction as functions of time for a single run of the improved localization approach. The covariance decreases as we move west ($\theta \approx 3.14$) until we get stuck on the saddle point (the grey shaded segment). Covariance further decreases as we take new observations from the improved sensor. We see a sudden change in direction ($\theta \approx 1.57$) when the covariance drops sufficiently low that the particles no longer surround the saddle.

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 their action most agrees. The majority vote to go north, allowing escape from the 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

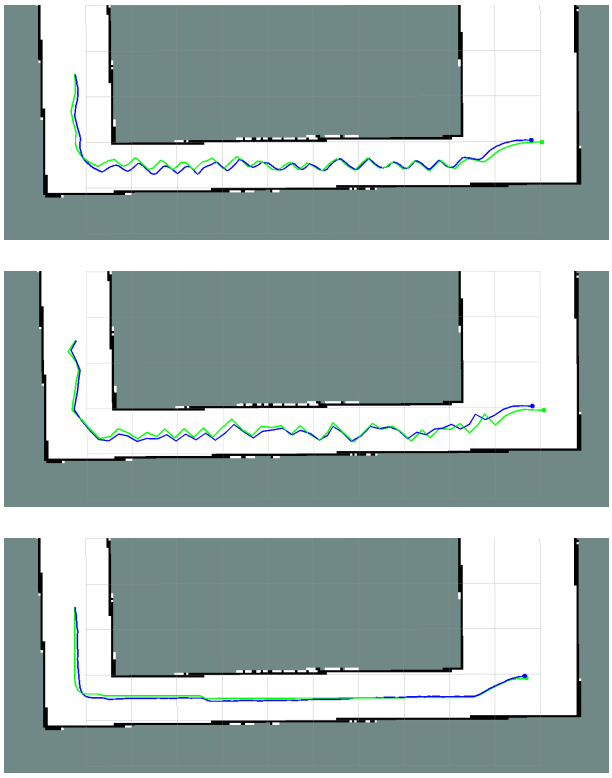

Fig. 9. Navigating to a goal in the bottom right corner. In all cases, actions are chosen at roughly 0.05m intervals. Top: Action chosen by steepest descent on the expected state generates chattering. Middle: Action chosen by stochastic gradient descent (SGD) generates a more randomized, but still jagged path. Bottom: Action chosen by GSPF generates a smoother path.

stationary point is identified. A new approximate Hessian is constructed, its positive eigenvalues confirm that we have reached the target, and GSPF terminates.

B. 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 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. We furthermore demonstrate that adapting stochastic gradient descent (SGD)—another popular optimization algorithm for potentially nonsmooth objective functions—to a particle filter does not resolve the chattering behavior either.

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 9 illustrates a single run for each of the approaches: Choosing the optimal action for the MCL filter's expected state (equivalent to steepest descent), choosing the optimal action for a random particle (the equivalent of SGD), or choosing the action based on GSPF. The first two approaches show chattering, while the last shows that the consensus action generated by GSPF results in a relatively smooth path.

TABLE I

AVERAGE ANGULAR DIFFERENCE BETWEEN SUCCESSIVE ACTIONS

Update Rate	**Steepest Descent** mean (variance)	**SGD** mean (var)	**GSPF** mean (var)
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)

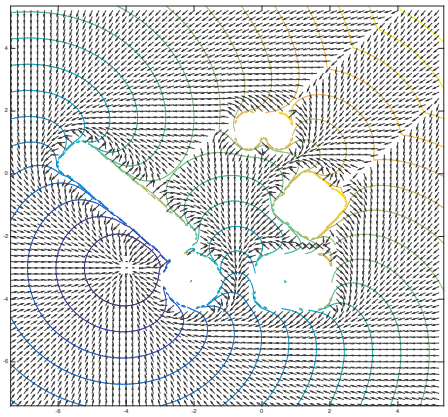

Fig. 10. The value function for the cluttered scene. The goal location is in the bottom left. Contours of the value function and the resulting gradient vector field are shown.

To test the hypothesis that chattering might be reduced by adjusting the stepsize, we ran simulations at a variety of update rates and measured the angle between successive action choices:

$$\sum_i \left| \arctan\left(\frac{y(t_i)}{x(t_i)}\right) - \arctan\left(\frac{y(t_{i-1})}{x(t_{i-1})}\right) \right|,$$

$$\text{where } p^*(t) = [x(t), y(t)]^T.$$

Only the portion of each trajectory in the hallway was used in this calculation. Results are shown in Table I. 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.

C. Cluttered Scene

In our final example we examine a scene with several obstacles of different shapes, as illustrated in Figure 10. 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 11. 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 for sharp turns when squeezing between obstacles,

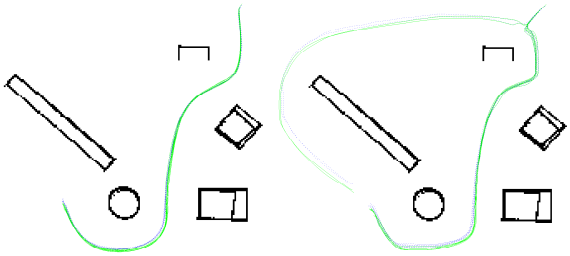

Fig. 11. Navigation through the cluttered scene. The robot travels from the upper right to the lower left. Ten trials are shown in each case. Left: Action chosen based on expected state. Right: Action chosen by GSPF.

tightly tracking the valleys of the value function. Both of 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 IV-B to choose a steeper descent direction.

VI. Conclusion

We observed that the gradient sampling algorithm from [1] can be used to resolve the chattering problem commonly encountered when generating optimal paths from value function approximations. We then proposed the Gradient Sampling with Particle Filter (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. 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, or to alternative sources of optimal actions, such as RRT* planners.

In the future we intend to explore the theoretical properties of the GSPF, its extension to anisotropic and non-holonomic dynamics, and its use with other planners. We also plan to develop a highly optimized version of the algorithm to explore the upper limits of the real-time update rate for a large number of particles.

Acknowledgment

The authors would like to thank Branden Fung and Carolyn Shen for their work on an earlier version of this algorithm, as well as Michael Overton and Michael Friedlander for several enlightening discussions about gradient sampling and other non-smooth optimization algorithms.

References

[1] J. V. Burke, A. S. Lewis, and M. L. Overton, "A robust gradient sampling algorithm for nonsmooth, nonconvex optimization," *SIAM Journal of Optimization*, vol. 15, pp. 751–779, 2005.

[2] J. N. Tsitsiklis, "Efficient algorithms for globally optimal trajectories," *IEEE Transactions on Automatic Control*, vol. AC-40, no. 9, pp. 1528–1538, 1995.

[3] R. Kimmel and J. A. Sethian, "Optimal algorithm for shape from shading and path planning," *Journal of Mathematical Imaging and Vision*, vol. 14, no. 3, pp. 237–244, 2001.

[4] E. Cristiani and M. Falcone, "Fast semi-Lagrangian schemes for the Eikonal equation and applications," *SIAM Journal on Numerical Analysis*, vol. 45, no. 5, pp. 1979–2011, 2007.

[5] 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*, vol. 227, no. 17, pp. 8191–8208, 2008.

[6] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA: MIT Press, 2005.

[7] F. H. Clarke, Y. S. Ledyaev, R. J. Stern, and P. R. Wolenski, *Nonsmooth Analysis and Control Theory*, ser. Graduate Texts in Mathematics. Springer, 1998.

[8] L. C. Evans and P. E. Souganidis, "Differential games and representation formulas for solutions of Hamilton-Jacobi-Isaacs equations," *Indiana University Mathematics Journal*, vol. 33, no. 5, pp. 773–797, 1984.

[9] D. Q. Mayne, "Model predictive control: Recent developments and future promise," *Automatica*, vol. 50, no. 12, pp. 2967 – 2986, 2014.

[10] A. Bry and N. Roy, "Rapidly-exploring random belief trees for motion planning under uncertainty," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Shanghai, China, May 2011, pp. 723–730.

[11] 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*, Porland, OR, June 2014, pp. 771–777.

[12] J. Pineau, G. Gordon, and S. Thrun, "Point-based value iteration: An anytime algorithm for POMDPs," in *International Joint Conference on Artificial Intelligence*, August 2003, pp. 477–484.

[13] H. Bai, D. Hsu, and W. S. Lee, "Integrated perception and planning in the continuous space: A POMDP approach," *International Journal of Robotics Research*, vol. 33, no. 9, pp. 1288–1302, 2014.

[14] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling, "Learning policies for partially observable environments: Scaling up," in *International Conference on Machine Learning*, July 1995.

[15] 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*, vol. 3, no. 3.2, 2009, p. 5.

[16] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *International Conference on Intelligent Robots and Systems (IROS)*, September 2004, pp. 2149–2154.

[17] S. Osher, "A level set formulation for the solution of the Dirichlet problem for Hamilton-Jacobi equations," *SIAM Journal of Mathematical Analysis*, vol. 24, no. 5, pp. 1145–1152, 1993.

[18] 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 2007. [Online]. Available: http://www.cs.ubc.ca/~mitchell/ToolboxLS/toolboxLS.pdf