Control Synthesis and Classification for Unicycle Dynamics using the Gradient and Value Sampling Particle Filters ⋆

Ariadna Estrada * **Ian M. Mitchell** *

* *Department of Computer Science, The University of British Columbia. Vancouver, BC, Canada (e-mail: {aestra42,mitchell}@cs.ubc.ca).*

Abstract: Value functions arising from dynamic programming can be used to synthesize optimal control inputs for general nonlinear systems with state and/or input constraints; however, the inputs generated by steepest descent on these value functions often lead to chattering behavior. In [Traft & Mitchell, 2016] we proposed the Gradient Sampling Particle Filter (GSPF), which combines robot state estimation and nonsmooth optimization algorithms to alleviate this problem. In this paper we extend the GSPF to velocity controlled unicycle (or equivalently differential drive) dynamics. We also show how the algorithm can be adapted to classify whether an exogenous input—such as one arising from shared human-in-the-loop control—is desirable. The two algorithms are demonstrated on a ground robot.

Keywords: Path planning, state uncertainty, state constraint, control synthesis, value function, Hamilton-Jacobi equation, particle filter, shared control, human-in-the-loop.

1. INTRODUCTION

Synthesis of globally optimal control actions for nonlinear systems with constrained inputs and/or dynamics is a challenging task, but it can be accomplished under quite general conditions through various forms of dynamic programming; for broad surveys of such techniques, see Bardi and Capuzzo-Dolcetta (1997) and Bertsekas (2007). The output of the dynamic programming process is generally some form of value function which encodes the optimal possible cost for each state in the state space. Given the value function, choosing the optimal control reduces to finding the input whose resulting motion through the state space generates the steepest descent of the value function.

In this paper we explore an algorithm designed to handle two challenges which arise when we attempt to apply this value function descent procedure to synthesize an (approximately) optimal path for a physical robot. First, the optimal inputs are often discontinuous with respect to the state, which can lead to chattering. Second, in practice, we have only an estimate of the robot's true state, not its exact value. Our algorithm takes advantage of the latter property to help resolve the former problem.

Of course, control synthesis is a well-studied problem, and yet another path planner is not likely to be an exciting novelty. However, our goal is not a fully autonomous robot, but rather one which can share control intuitively, productively, safely, and continuously with a human operator. We have not yet solved that bigger problem, but here we demonstrate how the control synthesis algorithm can be adapted to classify whether an exogenous input signal is productive according to an underlying value function.

In Traft and Mitchell (2016) we described the Gradient Sampling Particle Filter (GSPF) algorithm for an omnidirectional, velocity-controlled robot. The GSPF solves the two challenges described above for synthesizing control inputs from a value function. The contributions of this paper are:

- To extend the GSPF to synthesize control inputs from a value function for a robot with velocity-controlled unicycle dynamics. Unicycle dynamics are mathematically equivalent to differential drive, and so describe the most commonly encountered type of wheeled ground robot.
- To modify the GSPF to classify whether an exogenous input signal is productive with respect to the value function. The proposed algorithm samples the value function rather than its gradient, so we call it the Value Sampling Particle Filter (VSPF).
- To demonstrate both algorithms on a physical robot.

2. BACKGROUND

In this section we briefly describe the three main algorithmic ideas that we make use of in this paper. Each derives from a distinct area of mathematics and engineering; the connection between them will be described in section 3.

2.1 The Particle Filter

The particle filter is a non-parametric implementation of the Bayes filter and—under the moniker Monte Carlo Localization (MCL)—it is one of the most robust and

⋆ This work was supported by the AGE-WELL Network Center of Excellence and the National Science and Engineering Research Council of Canada (NSERC) Discovery Grant #298211.

commonly used approaches to robot localization (for example, see Thrun et al. (2005)). The idea behind MCL is to represent state uncertainty with a set of samples from a posterior distribution that approximates the true state of the system. Each sample (called particle) is a hypothesis of what the true state might be.

An MCL state estimate takes the form of a set of particle states $\{\boldsymbol{x}^{[m]}\}_{m=1}^{M}$ and particle weights $\{w^{[m]}\}_{m=1}^{M}$. Similar to other Bayes filter algorithms, MCL constructs its approximation of the state at time t based on the set of samples from time $t-1$, the current observation $z(t)$, and the control $\boldsymbol{u}(t)$. First, for every particle in the set, a new state hypothesis $\boldsymbol{x}^{[m]}(t)$ is generated by sampling the state transition distribution (also called the motion model)

$$\boldsymbol{x}^{[m]}(t) \sim p(\boldsymbol{x}^{[m]}(t)|\boldsymbol{u}(t), \boldsymbol{x}^{[m]}(t-1)) \tag{1}$$

Next, the particle's weight is updated based on how well the prediction $\boldsymbol{x}^{[m]}(t)$ matches the observation $z(t)$ using the observation model $w^{[m]}(t) = p(z(t)|\boldsymbol{x}^{[m]}(t))$. Finally, a "resampling" step is performed where a new set of particles is sampled (with replacement) from the current set with probability proportional to the particle's weight. The weights of the particles in the resampled distribution are reset to unity. In the remainder of this paper, we will only use the particle filter after resampling and hence can ignore the weights.

2.2 Path Planning with Value Functions

We will work with a robot represented by state $\boldsymbol{x} \in \Omega$ subject to dynamics $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$ which are Lipschitz continuous in $\boldsymbol{x}$ and continuous in input $\boldsymbol{u} \in \mathcal{U}$. We assume that Ω is the closure of an open set and $\mathcal{U}$ is compact and convex. Our goal is to navigate the robot from its current state to a known target set $\mathcal{T} \subset \Omega$ which is also the closure of an open set. Trajectories $\boldsymbol{x}(\cdot) : [t_0, t_f] \to \Omega$ will be judged by an additive cost metric

$$\psi(\boldsymbol{x}_0) = \inf_{\boldsymbol{x}(\cdot)} \int_{t_0}^{t_f} c(\boldsymbol{x}(s))\, ds, \tag{2}$$

where the cost function $c(\boldsymbol{x}) > 0$, and feasible paths are such that $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$, $\boldsymbol{x}(t_f) \in \mathcal{T}$ and $\boldsymbol{x}(t) \in \Omega \setminus \mathcal{T}$ for $t_0 \leq t < t_f$. It may not be possible to achieve the minimum cost $\psi(\boldsymbol{x})$ to go from $\boldsymbol{x}$ to $\mathcal{T}$, but paths exist which get arbitrarily close.

The value function ψ in (2) satisfies a dynamic programming principle and can be shown (for example, see Bardi and Capuzzo-Dolcetta (1997)) to be the viscosity solution of a static Hamilton-Jacobi (HJ) partial differential equation (PDE)

$$\begin{aligned} G(\boldsymbol{x}, \nabla\psi(\boldsymbol{x})) = \min_{\boldsymbol{u}} \nabla\psi(\boldsymbol{x}) \cdot \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}), \quad & \text{for } \boldsymbol{x} \in \Omega \setminus \mathcal{T} \\ \psi(\boldsymbol{x}) = 0 \quad & \text{for } \boldsymbol{x} \in \partial\mathcal{T}. \end{aligned} \tag{3}$$

It is not generally possible to solve (3) analytically. For the purposes of this paper, we follow Osher (1993) and transform the problem into a time-dependent HJ PDE

$$\frac{\partial}{\partial t}\phi(t, \boldsymbol{x}) + \min_{\boldsymbol{u}} \frac{\nabla\phi(t, \boldsymbol{x}) \cdot \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})}{c(\boldsymbol{x})} = 0 \text{ for } \boldsymbol{x} \in \Omega, \tag{4}$$

where $\phi(0, \boldsymbol{x})$ is chosen as an implicit surface function for the target set $\{\boldsymbol{x} \mid \phi(0, \boldsymbol{x}) \leq 0\} = \mathcal{T}$. After solving (4), the value function can be extracted as

$$\psi(\boldsymbol{x}) = \min_{t}\{t \mid \phi(t, \boldsymbol{x}) = 0\}.$$

It is also not generally possible to solve (4) analytically, but we can approximate its solution using the Toolbox of Level Set Methods (see Mitchell (2007)). It should be noted that faster algorithms for solving equations of the form (3) are available, such as Sethian and Vladimirsky (2003); Kao et al. (2005); Falcone and Ferretti (2013), but for the purposes of this paper we compute the approximation of $\psi(\boldsymbol{x})$ offline and hence are not so concerned about the inefficiency of using (4).

Given the value function, the optimal action (which may not be unique) at state $\boldsymbol{x}$ is to choose $\boldsymbol{u}$ to minimize the Hamiltonian $G(\boldsymbol{x}, \nabla\psi(\boldsymbol{x}))$ in (3):

$$\boldsymbol{u}^* \in \operatorname*{argmin}_{\boldsymbol{u}} \nabla\psi(\boldsymbol{x}) \cdot \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}). \tag{5}$$

Although ψ may not be differentiable we can develop some intuition about this choice observing that formally

$$\nabla\psi(\boldsymbol{x}) \cdot \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}). = \frac{d\psi(\boldsymbol{x}(t))}{dt}$$

Consequently, the choice of action (5) results in the fastest possible decrease in ψ among feasible trajectories. However, this choice is equivalent to (constrained) steepest descent of ψ (where the dynamics constrain the feasible directions of motion) and so it can lead to the chattering behavior often seen in steepest descent optimization.

2.3 The Gradient Sampling Algorithm

The gradient sampling (GS) algorithm of Burke et al. (2005) is a simple yet powerful optimization method designed to find a local minimum of a function which is not necessarily differentiable or convex. At each iteration of GS, the gradient of the objective function is sampled on a set of randomly chosen points within a radius $\epsilon > 0$ of the current iterate $\boldsymbol{x}(t)$. It can be shown that the convex hull of this set of gradients is an approximation of the Clarke ϵ-subdifferential of the objective function at $\boldsymbol{x}(t)$. The minimum norm vector $\boldsymbol{g}(t)$ in this convex hull can be found by solving a quadratic program. If $\|\boldsymbol{g}(t)\| \neq 0$, then $\boldsymbol{g}(t)$ is a consensus descent direction for all gradient samples, and a standard line search procedure is used to determine a new iterate $\boldsymbol{x}(t+1) = \boldsymbol{x}(t) + \alpha\boldsymbol{g}(t)$ for some $\alpha > 0$. If $\|\boldsymbol{g}(t)\| = 0$, then a Clarke ϵ-stationary point has been found, and no consensus descent direction exists. In this case the standard GS algorithm reduces the sampling radius ϵ and generates a new set of samples. The algorithm terminates when the radius ϵ decreases to a predetermined threshold. In practice, the GS robustly converges to a local minimum for the types of objective functions considered here, and largely avoids the chattering associated with steepest descent optimization.

3. THE GRADIENT SAMPLING PARTICLE FILTER (GSPF) FOR CONTROL SYNTHESIS

The GSPF algorithm of Traft and Mitchell (2016) combines path planning using gradient descent on a value function and tracking state uncertainty with a particle filter. The resulting algorithm converges to stationary points while significantly reducing the chattering effect generated when running gradient descent on nearly non-differentiable value functions. The key idea behind the GSPF is to

sample the gradient at the particles' locations instead of sampling it at random points.

$$\boldsymbol{p}^{[m]}(t) = \nabla\psi(\boldsymbol{x}^{[m]}(t)) \tag{6}$$

Denoting the convex hull of the set of gradient samples as

$$\mathcal{P}(t) = \text{conv}(\{\boldsymbol{p}^{[1]}(t), ..., \boldsymbol{p}^{[m]}(t)\}),$$

a consensus direction that guarantees descent is computed by finding the point $\boldsymbol{p}^{\circledast}(t)$ with minimum norm

$$\boldsymbol{p}^{\circledast}(t) = \underset{\boldsymbol{p}\in\mathcal{P}(t)}{\text{argmin}} \|\boldsymbol{p}\|^2 . \tag{7}$$

A convex quadratic program is used to find this point efficiently. If the solution $\|\boldsymbol{p}^{\circledast}(t)\| \neq 0$, then $\boldsymbol{p}^{\circledast}(t)$ is a descent direction for all particles. For the omnidirectional dynamics studied in Traft and Mitchell (2016), the chosen action is simply to move in the direction

$$\boldsymbol{u}^{\circledast}(t) = \frac{\boldsymbol{p}^{\circledast}(t)}{\|\boldsymbol{p}^{\circledast}(t)\|_2}. \tag{8}$$

In this section we extend the GSPF to handle a robot with kinematic unicycle dynamics

$$\frac{d\boldsymbol{x}}{dt} = \dot{\boldsymbol{x}} = \begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}v\cos\theta\\ v\sin\theta\\ \omega\end{bmatrix} = \boldsymbol{f}(\boldsymbol{x}, v, \omega) \tag{9}$$

where the state $\boldsymbol{x} \in \Omega \subseteq \mathbb{R}^2 \times [0, 2\pi]$ consists of a planar position x, y and a heading θ and the control $\boldsymbol{u}$ consists of a linear velocity $v \in [v_{\min}, v_{\max}]$ and an angular velocity $\omega \in [\omega_{\min}, \omega_{\max}]$. These dynamics are not omnidirectional, so we cannot use (8) to choose our action. Substituting (6) and (9) into (5) we arrive at a nicely separable optimization

$$\begin{aligned}
&\underset{\boldsymbol{u}}{\text{argmin}} \nabla\psi(\boldsymbol{x}) \cdot \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})\\
&\quad = \underset{v,\omega}{\text{argmin}}\, \boldsymbol{p} \cdot \boldsymbol{f}(\boldsymbol{x}, v, \omega),\\
&\quad = \underset{v,\omega}{\text{argmin}}\, p_x v\cos\theta + p_y v\sin\theta + p_\theta\omega,\\
&\quad = \underset{v}{\text{argmin}}\, v(p_x\cos\theta + p_y\sin\theta) + \underset{\omega}{\text{argmin}}\, p_\theta\omega,
\end{aligned}$$

where p_x, p_y and p_θ are the components of the gradient in the state dimensions x, y and θ respectively. The choice of optimal action is then given by

$$\begin{aligned}
v^* &= \begin{cases} v_{\max} & \text{if } p_x\cos\theta + p_y\sin\theta < 0;\\ 0 & \text{if } p_x\cos\theta + p_y\sin\theta = 0;\\ v_{\min} & \text{otherwise.}\end{cases}\\
\omega^* &= \begin{cases} \omega_{\max} & \text{if } p_\theta < 0;\\ 0 & \text{if } p_\theta = 0;\\ \omega_{\min} & \text{otherwise.}\end{cases}
\end{aligned} \tag{10}$$

Note that even though the state and the gradient space are three-dimensional, the optimal actions are chosen in two separate one-dimensional spaces.

There are two approaches to adapting (10) to the GSPF framework: Compute a consensus gradient $p^{\circledast}(t)$ using (7) and substitute it in (10) to find the optimal actions, or evaluate the optimal controls $v^{[m]}$ and $\omega^{[m]}$ for each particle using (10) and then seek consensus choices among these actions. In this paper, we will explore the latter approach because we can demonstrate that if consensus actions are found then progress is made toward the target.

We explain the procedure for finding a consensus angular velocity $\omega^{\circledast}(t)$ first.

(1) For each particle location, sample the gradient of the value function: $\boldsymbol{p}^{[m]}(t) = \nabla\psi(\boldsymbol{x}^{[m]}(t))$.
(2) Compute the optimal angular velocity, $\omega^{[m]}(t)$, for each gradient in $\boldsymbol{p}^{[m]}(t)$ following (10).
(3) Compute the convex hull of these angular velocities

$$\mathcal{A}_\omega(t) = \text{conv}(\{\omega^{[1]}(t), \ldots, \omega^{[m]}(t)\}).$$

Note that the convex hull in this one-dimensional space will be an interval.
(4) Determine the consensus $\omega^{\circledast}$ as the minimum norm point in $\mathcal{A}_\omega$. Since this set is an interval, there is no need for a convex optimization as in (7):

$$\omega^{\circledast}(t) = \begin{cases} \omega_{\max} & \text{if } \forall m, \omega^{[m]}(t) = \omega_{\max};\\ \omega_{\min} & \text{if } \forall m, \omega^{[m]}(t) = \omega_{\min};\\ 0 & \text{otherwise.}\end{cases}$$

The procedure for finding a consensus linear velocity $v^{\circledast}(t)$ is the same, except that we follow the formula for v^* in (10) for step 2.

Like the original version of the GSPF, this unicycle version will converge to all stationary points rather than only the desirable local minimum at the target set. Therefore, it will not find a nonzero consensus action whenever the particle cloud surrounds a stationary point in the value function. For example, if $\omega^{\circledast}(t) = 0$ then some particles wish to turn left while others want to turn right and no consensus is possible. Fundamentally, this situation signals a localization failure: The uncertainty in the robot state is sufficient that any nonzero input may be counterproductive. Ideally, our response would be to improve our localization by, for example, taking additional sensor readings. We recognize that such a response is not always possible or desirable, so in the remainder of this section, we adapt the stationary point classification and resolution procedure from Traft and Mitchell (2016) to the unicycle dynamics. Similar to the consensus policy discussed above, we describe this strategy for the angular velocity. We denote the action choice from this method with $\omega^{\circ}(t)$ because it is not necessarily optimal when it takes nonzero value.

The purpose of the stationary point classification procedure is to determine whether the particles surround a local minimum (which by construction implies that the target has been reached to the extent possible) or a local saddle or maximum (from which we should try to escape by some nonzero choice of action). To differentiate these two cases, we construct a linear approximation of the particles' optimal angular velocity $\omega^{[m]}(t)$ as a function of the particles' heading $\theta^{[m]}(t)$. We use a least squares best fit because it is the cheapest to evaluate, but other choices are possible. Some care must also be taken to handle the periodic boundary conditions for $\theta \in [0, 2\pi]$.

If the slope of this approximation is negative, particles with smaller heading want to turn left while particles with larger heading want to right; in other words, we are at a local minimum in the value function with respect to angular velocity. This situation is likely to arise when the robot is moving toward the goal with some particles slightly to the right of the optimal path and some slightly to the left but no intervening obstacles. In this case, there is no clear benefit to turning, and we choose $\omega^{\circ}(t) = 0$.

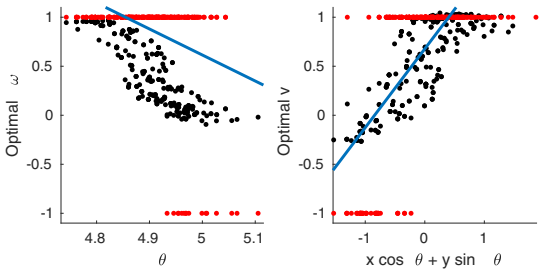


Fig. 1. Classification at a stationary point. Angular velocity is at a minimum because the slope of the fitted line is negative so we set $\omega^\circ(t) = 0$. The linear velocity is at a maximum so we vote and choose $v^\circ(t) = v_{\max}$.

On the other hand, if the slope of the approximation is positive, particles with smaller heading want to further decrease it by turning negatively while particles with larger heading want to further increase it; in other words, we are at a local maximum in the value function with respect to angular velocity. This situation is likely to arise when the robot is facing almost directly away from the target or the target is almost exactly on the far side of an obstacle. In such situations, there may be no consensus as to whether left or right is optimal, but because the value function is continuous, there is also little difference in terms of time to reach the target no matter which choice is made. Therefore, we use a simple voting procedure to break the deadlock. For $\sigma \in \{-1, 0, +1\}$ we define

$$\alpha_\omega^\sigma(t) = \operatorname*{count}_m \left(\{ \boldsymbol{x}^{[m]}(t) \mid \operatorname{sign}(p_\theta^{[m]}(t)) = \sigma \} \right),$$

to determine how many particles want to turn right ($\alpha_\omega^{-1}(t)$), not turn ($\alpha_\omega^0(t)$) and turn left ($\alpha_\omega^{+1}(t)$). We identify which option $\sigma_\omega^*(t)$ is favored by the plurality of particles and we choose a corresponding input value

$$\sigma_\omega^*(t) = \operatorname*{argmax}_\sigma \alpha_\omega^\sigma(t),$$

$$\omega^\circ(t) = \begin{cases} \omega_{\min} & \text{if } \sigma_\omega^*(t) = -1; \\ 0 & \text{if } \sigma_\omega^*(t) = 0; \\ \omega_{\max} & \text{if } \sigma_\omega^*(t) = +1; \end{cases}$$

The procedure to find a $v^\circ(t)$ is the same, except that the abscissae of the linear fit are provided by $x^{[m]}(t)\cos\theta^{[m]}(t) + y^{[m]}(t)\sin\theta^{[m]}(t)$ instead of $\theta^{[m]}(t)$ and

$$\alpha_v^\sigma(t) = \operatorname*{count}_m \left(\left\{ \boldsymbol{x}^{[m]}(t) \,\middle|\, \operatorname{sign} \begin{pmatrix} p_x^{[m]}(t)\cos\theta^{[m]}(t) \\ +p_y^{[m]}(t)\sin\theta^{[m]}(t) \end{pmatrix} = \sigma \right\} \right).$$

Figure 1 illustrates the classification and resolution strategy using data from the first example in section 5. The plot on the left shows the results for angular velocity. The black dots are $p_\theta^{[m]}(t)$ and red dots are $\omega^{[m]}(t)$ as functions of $\theta^{[m]}(t)$. A line is fit to the red dots, and the negative slope indicates a local minimum, so we choose $\omega^\circ(t) = 0$. The plot on the right shows the results for linear velocity. The positive slope of the fitted line indicates a local maximum, so the voting procedure is invoked, $\sigma_v^*(t) = +1$ (in fact, $\alpha_v^{+1}(t)$ is the majority vote in this case, not just the plurality) and we choose $v^\circ(t) = v_{\max}$.

We conclude our description of the adaptation of GSPF to the unicycle dynamics by noting that the classification and resolution procedure also serves as a termination criterion, just as it did in the omnidirectional version from Traft and Mitchell (2016). Any particle which lies inside the target set will have $\boldsymbol{p}^{[m]}(t) = 0$; consequently, if any particle lies within the target set no nonzero consensus action will be found. If improved localization is not pursued, one of three outcomes will occur. If the particles surround the target, a local minimum will be detected, and the input will be chosen as zero. If the particles do not surround the target, but the majority lies inside it, then $\sigma_\omega^*(t) = 0$ and $\sigma_v^*(t) = 0$ and the input will again be chosen as zero. Finally, if the particles do not surround the target, but the majority lies outside it, then the voting procedure will find a nonzero input which favors this group of particles.

4. VALUE SAMPLING PARTICLE FILTER FOR CONTROL CLASSIFICATION

GSPF synthesizes actions which are optimal under the cost metric (2) to the extent possible given the state uncertainty captured by the particle filter. Assuming that the robot is granted the authority to control its own motion completely, this (admittedly limited) form of optimality is appropriate given the state uncertainty and the fact that the cost function encodes all information available to the robot about the desirability of trajectories.

By contrast, in the shared control scenario the robot is sharing motion decisions with an exogenous input coming from another agent, such as a human-in-the-loop. We assume that the agent generating this input may have access to information beyond that available to the robot and which cannot be communicated to the planner in an appropriate form (such as modification to the cost function or particle weights). On the other hand, we also assume that neither the robot's information nor the other agent's strictly dominates, so neither should be ignored.

In the previous section, we explored the scenario of how to choose among a set of conflicting actions generated from the samples in the particle filter. In this section, we pursue that same capability for the case of an exogenous input: Determine whether that input is consistent with the robot's current knowledge of its location and how the target set can be reached. Such a classification could be used to drive an intervention or other mode switching behavior in a shared control system. We leave to future research the subsequent question of what intervention to apply when the input is inconsistent with robot knowledge.

In the GSPF, an input $\boldsymbol{u}(t)$ is desirable if $\nabla\psi(\boldsymbol{x}(t)) \cdot \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \leq 0$; in other words, the chosen action does not lead to an instantaneous increase in ψ. However, this requirement to align with (the negative of) $\nabla\psi$ may be too tight a constraint. To see why, consider the case where a symmetric obstacle lies exactly between the robot and the target set. Paths to the left or right of the obstacle take approximately the same time, which manifests as a ridge in the value function ψ. With probability one, the actual state of the robot does not lie exactly on the ridge, so $\nabla\psi$ exists and either left or right is technically optimal. In this scenario both choices could be reasonably classified as desirable even though one will generate motion leading to an instantaneous increase in ψ; at the same time moving straight the obstacle is likely not desirable, even though it might generate an instantaneous decrease

in ψ. Therefore, instead of using $\nabla\psi(\boldsymbol{x}(t))$ ("gradient sampling") to measure the decrease in ψ, we will use the difference given by

$$\Delta\psi(\boldsymbol{x}(t)) = \psi(\boldsymbol{x}(t+1)) - \psi(\boldsymbol{x}(t)) \tag{11}$$

("value sampling"). The future state $\boldsymbol{x}(t+1)$ is estimated according to the motion model (1) using the exogenous input signal, $\boldsymbol{u}^e(t)$, and the current state, $\boldsymbol{x}(t)$.

In the GSPF, we sought a motion command that achieved descent for all current particle samples. In the context of classifying an input from an agent which may have additional knowledge, such a constraint can be relaxed because the agent may know that certain states currently represented by particles are highly unlikely. Therefore, instead of seeking consensus we ask only that the chosen action produces descent for a sufficient fraction of the particles. That fraction is an adjustable parameter which could range from a single particle to the entire particle set.

Finally, GSPF considers inputs desirable as long as they do not lead to an instantaneous increase in the value function. For classification purposes, we may want to set a minimum acceptable level of decrease as measured by (11).

In summary, the Value Sampling Particle Filter (VSPF) evaluates an exogenous input $\boldsymbol{u}^e(t)$ by

(1) For each particle, predict what the future state would be if we applied $\boldsymbol{u}^e(t)$

$$\boldsymbol{x}^{[m]}(t+1) \sim p(\boldsymbol{x}^{[m]}(t+1) \mid \boldsymbol{u}^e(t), \boldsymbol{x}^{[m]}(t)). \tag{12}$$

(2) For each particle, evaluate the difference in the value function

$$\Delta\psi(\boldsymbol{x}^{[m]}(t)) = \psi(\boldsymbol{x}^{[m]}(t+1)) - \psi(\boldsymbol{x}^{[m]}(t)). \tag{13}$$

(3) Classify the input as desirable based on a summary criterion across the particles; for example,

$$\operatorname*{count}_m \left(\{\boldsymbol{x}^{[m]}(t) \mid \Delta\psi(\boldsymbol{x}^{[m]}(t)) \le \mu\}\right) \ge k \tag{14}$$

or

$$\sum_m \{\Delta\psi(\boldsymbol{x}^{[m]}(t)) \mid \Delta\psi(\boldsymbol{x}^{[m]}(t)) \le 0\} \le \mu$$

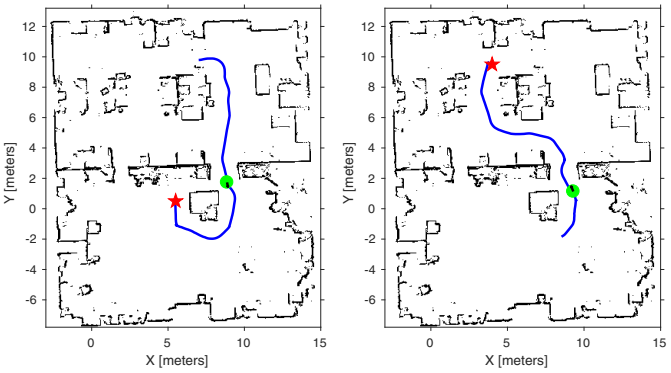
for some thresholds $k > 0$ and $\mu \le 0$.

Computationally, the cost of VSPF is linear in the number of particles: For each particle we need one motion prediction, two interpolations of ψ, and a small constant number of basic arithmetic and comparison operations.

5. EXAMPLES

We test the GSPF and VSPF on a Segway Robotic Mobility Platform (RMP) 100 equipped with a Hokuyo UTM-30LX laser rangefinder with a field of view of 150 degrees. We use a Lenovo ThinkPad W530 (8 cores, 2.6 GHz Intel Core i7) with 8GB of RAM and Ubuntu 14.04 LTS. We implement the algorithms using the ROS framework paired with MATLAB R2016b and the Robotics System Toolbox. The robot's compatibility with ROS allows us to efficiently use the adaptive Monte Carlo Localization (AMCL) implementation approach described in Thrun et al. (2005).

We approximate the value function using the Toolbox of Level Set Methods from Mitchell (2007) and approximate the gradient using upwinded first order finite differences at each node of the grid. To sample the gradient at any point within the grid we use MATLAB's `interpn` function with the default linear interpolation (trilinear in this case).



(a) (b)

Fig. 2. Examples of paths generated by the GSPF.

5.1 Control Synthesis

Our first two examples illustrate how we can synthesize control actions for the unicycle dynamics with the modified GSPF algorithm. Figure 2 exhibits two navigation scenarios with a red star at the robot's target and a blue line denoting its trajectory (as estimated by AMCL). The green circles are examples of locations along the trajectories where there is no consensus action. The algorithm was able to resolve these stationary points using the method described in section 3 without triggering additional sensor readings. For the location in figure 2a, the resolution resulted in forward motion (see figure 1 and accompanying text). For the location in figure 2b, the classification occurs according to the plots in figure 3. Since the slopes of the fitted lines are positive, we classify the saddle point as a maximum in both v and ω. Following the voting procedure we set $v^\circ(t) = v_{\max}$ and $\omega^\circ(t) = \omega_{\min}$.

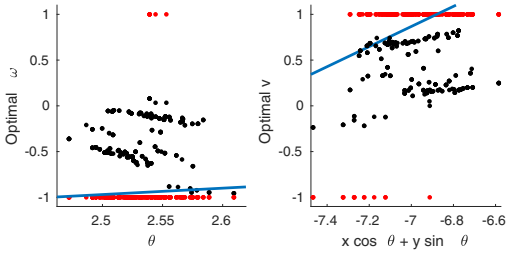
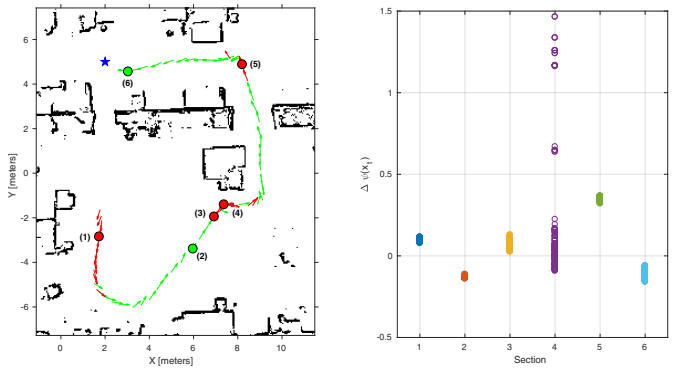


Fig. 3. Stationary point classification for Fig. 2b. Classified as local maximum in v and ω. Voting generates clearly favored actions $\omega^\circ(t) = \omega_{\min}$ and $v^\circ(t) = v_{\max}$.

These examples show that the GSPF is capable of generating smooth paths and navigating the robot through a doorway while keeping a safe distance from obstacles.

5.2 Control classification

Our third example demonstrates the proposed VSPF approach to classify motion commands according to their degree of productivity towards a goal. In this case, the user has a predefined target and provides velocity controls to the robot using a joystick device. For this example, we used



(a) Sample trajectory for a user operated robot. The direction of the arrows denote $\boldsymbol{u}^e(t)$.

(b) Difference in ψ (11) for the identified points in figure 4a. Negative values are desirable.

Fig. 4. Classification of exogenous control signals.

metric (14) with $\mu = 0$ and $k = 0.7$ to determine whether the user input signal is desirable. Figure 4a depicts the scenario with the direction of the arrows encoding the user's input at each point along the trajectory.

At location (1), the user is deliberately driving away from the goal. The control classification algorithm predicts the future states of the particles under the user input (12), samples the value function at the current and predicted particle locations, and evaluates the difference (13). The corresponding change for each particle in the value function, $\Delta\psi(\boldsymbol{x}^{[m]})$, is represented by the left-most set of circles in figure 4b. Since there is no decrement for any of the particles, the algorithm classifies the user signal as undesirable (red arrows represent undesirable inputs).

The value function differences at the remaining labeled locations on the trajectory show distinct patterns. When the user maintains the controls consistent with the goal, such as locations (2) and (6), the differences are negative for all particles and the control is classified as productive (green arrows). If the user steers too close to obstacles, as in location (3), there is also an increment in the value function meaning that this action is undesirable. At location (4) the user steers away from the obstacle but also away from the goal, resulting in an insufficient portion of the particles (less than 70%) predicting a decrease in the value function, thus the input is classified as undesirable. In location (5), we see the case where the goal is to the left of the current position. When the user continues without turning, the estimated differences are all positive, and the control is classified as undesirable.

This example shows that the VSPF can flexibly classify an exogenous input signal according to whether it is likely to be making progress toward a target set.

6. CONCLUSIONS AND FUTURE WORK

We extended the GSPF algorithm for control synthesis with value functions under state uncertainty from omnidirectional to unicycle dynamics. We then demonstrated how the algorithm can be adapted to classify whether an exogenous input is desirable (as judged by the value function), this time by sampling the value function itself rather than its gradient (the VSPF).

For the GSPF synthesis algorithm, we are exploring methods of extending it to more complex dynamics and underlying quality metrics other than value functions. We demonstrated the VSPF classification algorithm with the same value function used for GSPF synthesis, but VSPF can also be used for safety assurance through appropriately defined value function representations of viability kernels or reachable sets, such as those described in Kaynama et al. (2015); Mitchell et al. (2016).

ACKNOWLEDGEMENTS

The authors would like to thank Neil Traft and Ren Yi for their work on earlier versions of the GSPF, Lili Meng and Zicong Fan for their help with ROS and the robots, and Dr. Clarence de Silva and UBC's Industrial Automation Lab for the loan of the Segway RMP.

REFERENCES

Bardi, M. and Capuzzo-Dolcetta, I. (1997). *Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman equations*. Birkhäuser, Boston.

Bertsekas, D.P. (2007). *Dynamic Programming and Optimal Control*. Athena Scientific, Belmont, Massachusetts, 3 edition.

Burke, J.V., Lewis, A.S., and Overton, M.L. (2005). A robust gradient sampling algorithm for nonsmooth, nonconvex optimization. *SIAM Journal on Optimization*, 15(3), 751–779.

Falcone, M. and Ferretti, R. (2013). *Semi-Lagrangian Approximation Schemes for Linear and HamiltonJacobi Equations*. SIAM. doi:10.1137/1.9781611973051.

Kao, C.Y., Osher, S., and Tsai, Y.H. (2005). Fast sweeping methods for static Hamilton-Jacobi equations. 42(6), 2612–2632.

Kaynama, S., Mitchell, I.M., Oishi, M.M.K., and Dumont, G.A. (2015). Scalable safety-preserving robust control synthesis for continuous-time linear systems. *IEEE Transactions on Automatic Control*, 60(11), 3065–3070.

Mitchell, I.M. (2007). A toolbox of level set methods (version 1.1). Technical Report TR-2007-11, Department of Computer Science, University of British Columbia, Vancouver, BC, Canada. URL `http://www.cs.ubc.ca/~mitchell/ToolboxLS/toolboxLS.pdf`.

Mitchell, I.M., Yeh, J., Laine, F.J., and Tomlin, C.J. (2016). Ensuring safety for sampled data systems: An efficient algorithm for filtering potentially unsafe input signals. In *Proceedings of the IEEE Conference on Decision and Control*, 7431–7438. Las Vegas, NV. doi:10.1109/CDC.2016.7799417.

Osher, S. (1993). A level set formulation for the solution of the Dirichlet problem for Hamilton-Jacobi equations. 24(5), 1145–1152.

Sethian, J.A. and Vladimirsky, A. (2003). Ordered upwind methods for static Hamilton-Jacobi equations: Theory and algorithms. 41(1), 325–363.

Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic robotics*. MIT press.

Traft, N. and Mitchell, I.M. (2016). Improved action and path synthesis using gradient sampling. In *Proceedings of the IEEE Conference on Decision and Control*, 6016–6023. Las Vegas, NV. doi:10.1109/CDC.2016.7799193.