2011 IEEE International Conference on Robotics and Automation
Shanghai International Conference Center
May 9-13, 2011, Shanghai, China

Minimum Uncertainty Robot Navigation Using Information-guided POMDP Planning

Salvatore Candido and Seth Hutchinson

Abstract— **A ubiquitous problem in robotics is determining policies that move robots with uncertain process and observation models (partially-observed state systems) to a goal configuration while avoiding collision. We propose a new method to solve this minimum uncertainty navigation problem. We use a continuous partially-observable Markov decision process (POMDP) model and optimize an objective function that considers both probability of collision and uncertainty at the goal position. By using information-theoretic heuristics, we are able to find policies that are effective for both minimizing collisions and stopping near the goal configuration. We additionally introduce a filtering algorithm that tracks collision free trajectories and estimates the probability of collision.**

I. INTRODUCTION

We propose a new motion planning algorithm to solve the minimum uncertainty navigation problem for stochastic robot systems. For deterministic robot systems, motion planning research has progressed from considering strict feasibility to computing optimum robot plans. Analogously, in the stochastic case, different control policies will result in different likelihoods of collision and uncertainty at the goal position. Thus, it is natural to synthesize control policies that seek to minimize these criteria. We address the problem of moving a robot with stochastic motion and observation models from a start configuration to a goal configuration with intent to minimize expected distance of the robot from the goal and likelihood of collision.

To cope with inherent noise in system components and imperfect modeling of sensors, robots, and environments, robot motion and sensor models are typically formulated probabilistically. Thus, many robot models fall into the class of continuous partially-observable Markov decision processes (POMDPs). Unless the models are of a special form, e.g., linear quadratic Gaussian, for this class of processes an exact optimum feedback control law is hard to find due to representation and computation issues. Our approach belongs to a recently emerging class of methods, e.g., [1]–[3], that use *local policies* or *macro-actions* to construct a POMDP policy. Essentially, rather than optimizing single controls (single time steps), we optimize the choice of controls for a short time span (over multiple time steps). We use the terminology local policy to make clear that our approach is to use feedback policies (not open loop sequences of controls) and our final product is a switched belief-feedback policy that uses local policies as modes. (This will be defined and explained in Section II.)

Our general approach has been discussed in [3], so the purpose of this paper is to discuss how this method can be adapted to the continuously modeled state, control, and observation spaces of the robot navigation task. We demonstrate that using information-theoretic heuristics to guide local policies results in a viable motion planning procedure. This introduces a need for regularization kernels [4] to adapt our particle filter representation of belief for use with these information-theoretic heuristics. This leads to a novel filtering algorithm that tracks collision-free trajectories of a system and simultaneously estimates the probability of trajectories leading to collision. We demonstrate the results of our method in simulation.

Most realistic sensor models are not invariant to the robot's configuration and even if we could evaluate all paths through the configuration space to determine the one that provides the most localization information, we cannot hope to follow that path exactly because of uncertainty in the robot's process model. Thus, a minimum uncertainty path involves a trade-off between exploring regions of the state space where more information about the state can be gained and taking a path to the goal that introduces the smallest amount of process noise into the estimate of the system's state. Elements of this trade-off can be captured by incorporating ideas from *information theory*. (The use of information theory in robotics and planning has been explored in many contexts, e.g., [5]–[9].) Paths that minimize *entropy* serve to reduce uncertainty. Goal-seeking behavior can be achieved by minimizing the *Kullback-Leibler* (KL) *divergence* between the expected future posterior pdf (i.e., the pdf over the robot's state at some future stage) and a pdf representing the goal. In practice, one can compute controls that achieve one of these objectives over a short time span, but no one of these heuristics is equivalent to our objective. Our approach is to define a set of *local policies* based on these criteria that specify closed-loop control laws to be executed for a finite (but not fixed) period of time. We then optimize the switching law of a switched policy, whose modes correspond to the local policies, to synthesize a composite policy to achieve the specified control objective.

Once an objective function, set of local policies, and belief approximation method is in place, we employ our optimization algorithm. Our approach is an anytime algorithm that computes a belief-feedback policy in advance of the robot moving. It is based on a method like value iteration to learn a switching law. We use a Monte Carlo approximation to

This material is based in part upon work supported by the National Science Foundation under award CNS 0931871.

S. Candido is in the Department of Electrical and Computer Engineering at the University of Illinois. candido@illinois.edu

S. Hutchinson is a professor of Electrical and Computer Engineering at the University of Illinois. seth@illinois.edu

evaluate system behavior and optimize a switching law based on sampling.

A. Problem Definition

We now formally define our minimum uncertainty navigation objective. Let $\boldsymbol{b}_t$ represent a belief, the posterior probability distribution over the state space at time t. The quantity $\boldsymbol{p}_t^{\text{fail}}$ denotes the probability the robot collides with an obstacle in the first t stages, and let c^{fail} be a user-defined constant in $[0, +\infty)$ that weights the importance of safety from collision versus the eventual distance from the desired goal state. We will quantify that distance using the KL divergence, explicitly defined in (2), which measures the dissimilarity between two probability density functions.

The target pdf b_{goal} will be selected based on target positions in the configuration or workspace. This distribution can be specified for a number of nontrivial goal objectives, but we will consider it to simply encode the robot attempting to reach x_{goal}, i.e.,

$$b_{\text{goal}} := \delta(x_{\text{goal}})$$

where $\delta(\cdot)$ is the Dirac delta function. The planning objective[1] is then to find a belief-feedback policy π^* that minimizes

$$J(b_0, \pi) = \mathrm{E}\left[c^{\text{fail}} \boldsymbol{p}_T^{\text{fail}} + \mathrm{KL}(\boldsymbol{b}_T || b_{\text{goal}})\right]. \qquad (1)$$

where b_0 is the initial belief of the robot's state and the policy modifies the behavior of the system, i.e., the evolution of $\boldsymbol{b}_t$ and $\boldsymbol{p}_t^{\text{fail}}$. The planning objective is defined in terms of a dynamically chosen (as part of the control law) stopping time T.

B. Related Research

There has been a large volume of research on optimizing systems modeled as POMDPs. Some work is directed to specific robotics problems, while other methods focus on computing optimal policies for general systems modeled with stochastic difference equations. Consideration of uncertainty, created by an uncertain process model, was first combined with sampling-methods in [10] to predict the behavior of a system. Rather than just predicting, in [11]–[13] attempts were made to extend sampling-based algorithms to plan for systems with uncertainty. The Belief Roadmap planner (BRM) [9], [14] attempts to build a roadmap over a parameterized belief space by sampling configurations, lifting them into the belief space, and connecting them using a local planner. The Sampling Hyperbelief Optimization Technique (SHOT) [15] uses a similar methodology in the space of posterior probability functions over the belief space. A similar approach can be used in a forward-based, diffusive search of the reachable belief space, and several approaches using this design principle have been put forth, e.g., [1]–[3], [16].

Other work in the robotics community generates plans to minimize uncertainty for specific robot tasks. For example, the active localization algorithms of [17] and [18] make robot localization more effective by specifically considering expected uncertainty of the localization algorithm while planning the next control the robot will receive. These algorithms generate a control to minimize uncertainty at the next stage, but unlike this work, do not optimize over a path.

The general POMDP optimization problem has a rich history. In [19], it was shown that the optimal value function of a finite POMDP is piecewise-linear and convex and, for any finite horizon, one could construct the optimal value function exactly by considering only a finite number of points. While the structure of the value function and an algorithm to exactly compute it is known, the intractability of computing exact solutions is well-known [20]. While a number of general methodologies have been proposed to approximate exact solutions, the most popular trend has been point-based approximation of the value function by Monte Carlo sampling, e.g., [21]–[25]. However, these methods have not been shown to scale to many robotic tasks in continuous spaces.

II. Background

The geometry of a robot system can be parameterized by a configuration $q \in \mathcal{Q}$. For a purely kinematic system model, the system's state $x \in \mathcal{X}$ is identical to the configuration. If the system dynamics are also modeled, the state will be $x = [q, \dot{q}]'$. A robot system has a process model

$$\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, u_t, \boldsymbol{n}_t)$$

where $u_t \in \mathcal{U}$ is the control (action) and $\boldsymbol{n}_t$ is a random vector representing process noise which is drawn from an iid random process. This model encodes the kinematic or dynamic trajectory constraints of the robot system, and how the process is affected by uncertainty $\boldsymbol{n}_t$. The observation model dictates how the robot's state affects its sensors, and is modeled

$$\boldsymbol{y}_t = h(\boldsymbol{x}_t, \boldsymbol{m}_t)$$

where $\boldsymbol{m}_t$ represents the sensor noise. For our analysis, we will assume $\mathcal{X}$, $\mathcal{U}$, and $\mathcal{Y}$ are continuous. However, the random vectors $\boldsymbol{y}$, $\boldsymbol{x}$, or $\boldsymbol{u}$ may also be discrete or mixed. The same analysis can be applied in these cases with minor modification.

We rely on the posterior probability function, referred to as a *belief*, to act as a sufficient statistic of the information state [26], i.e., the initial belief and the set of controls and observations up to this point. We will denote the belief[2] at stage t as b_t, i.e.,

$$b_t(x) := \mathrm{f}_{\boldsymbol{x}_t | \Gamma_{x_0}, \boldsymbol{y}_{1:t}, \boldsymbol{u}_{1:t}}(x \mid \Gamma_{x_0}, y_{1:t}, u_{1:t})$$

[1]This is a scalarized multi-objective cost function, where the two objectives are often in competition with one another. In principle, any multi-objective optimization characterization could be used in place of scalarization, e.g., computing a Pareto optimal set of policies. Our optimization procedure allows separation each objective until the value is computed, so it does not preclude other techniques.

[2]In practice, we often cannot represent the exact pdf and instead use an approximation. We will use $\hat{b}_t$ to refer to an approximation of the exact belief b_t.

where Γ_{x_0} is a pdf over the state space that represents the initial condition of the system. The space of all possible probability distributions on $\mathcal{X}$ is referred to as the *belief space* $\mathcal{P}_b$. Given a control u and observation y, the evolution of the process in the belief space is well-defined and deterministic. The *belief transition operator*, denoted $\phi_{y,u}$ for a particular control and observation, for the POMDP is defined $b_{t+1} = \phi_{y,u} b_t$ and it corresponds to a standard Bayesian filtering procedure. Thus, the POMDP can be viewed as an MDP whose state variable is the belief, and observations act as the source of uncertainty. A more detailed discussion of MDP's with continuous state variables and the precise form of $\phi_{y,u}$ can be found in [27].

Since the sequence of observations $y_{1:\infty}$ will not be known at planning time, the evolution of the POMDP in the belief space remains a random variable. If we want to analyze the complete behavior of a POMDP in the future, e.g., to evaluate the possible effect of a control policy, we must consider all sample paths that have nonzero probability. We accomplish this by lifting the POMDP process to a higher dimensional space of probability functions defined on the belief space. We refer to this lifted space as the *hyperbelief space* and, in the hyperbelief space, the system evolves deterministically, thus eliminating (in some sense) the explicit consideration of uncertainty during the planning process at the expense of a much higher representational cost. A *hyperbelief* β is a probability density functional defined over $\mathcal{P}_b$.

$$\beta_t(b) = \mathrm{f}_{\boldsymbol{b}_t}(b \mid b_0, \pi)$$

The hyperbelief transition operator is well-defined.

For deterministic motion planning, the planner computes a path. However, with a stochastic system we will not be able to track a single path. Thus, the goal is to find a belief-feedback policy $\pi : \mathcal{P}_b \to \mathcal{U}$ that minimizes the cost criterion in (1). This implies that rather than computing a sequence of landmarks or a path through the configuration space, our goal is to compute a mapping from a sufficient statistic of the information state to a control. This mapping should be optimized for every possible information state that can be reached during an execution. We will achieve this by using a switched policy,

$$\overline{\pi} : \mathcal{P}_b \times \Psi \times \mathbb{Z}^+ \to \mathcal{U}$$

where Ψ is a set of local policies available for use. A local policy $\psi = \{\pi, a\}$ is a 2-tuple where π is a belief-feedback policy and the stopping condition $a : \mathcal{P}_b \times \mathbb{Z}^+ \to \{0, 1\}$ indicates when the local policy should stop. The second and third input parameters of the switched policy can be considered to be the state of the policy. We will assume that the policy can be implemented on the robot in a way that makes use of some internal memory. Under this assumption, the switched policy formulation above can be used in place of a standard belief-feedback policy given a well-defined initial condition, i.e., given a belief provide a control at every time step with no other external manipulation of the policy.

In the development that follows, we will use a number of concepts from information theory. A standard reference text on information theory is [28]. The *differential entropy* (which will subsequently be referred to as entropy) is a measure of the uncertainty associated to a random variable

$$\mathrm{H}(\boldsymbol{x}) = -\int_{\mathcal{X}} b_t(x) \log_2 b_t(x) dx.$$

The KL divergence is used in this paper as a pseudo-metric of distance between two pdf's

$$\mathrm{KL}(b^i || b^j) = \int_{\mathcal{X}} b^i(x) \log \frac{b^i(x)}{b^j(x)} dx \tag{2}$$

as is common in the literature.

III. Method

Our solution to the minimum uncertainty navigation problem involves utilizing a set of local policies that specify multi-stage trajectories through the hyperbelief space. To avoid the computational intractability of optimizing the policy one control at time, we optimize the policy in a coarse fashion. Specifically, we use a switched policy whose modes and switching conditions are defined by a set of local policies and optimize the switching law. However, since we are using closed-loop feedback control policies at every stage, this is not a straightforward conversion of the original POMDP to a temporally abstracted POMDP. In this section, we begin by describing the set of information-driven local policies. We then demonstrate how we can modify the canonical use of the stochastic Bellman equation into a form suitable for optimizing a switched policy.

A. Local Policies

We use two kinds of local policies: those that are designed to decrease uncertainty in the robot's state, and those that draw the robot toward the goal state. To reduce uncertainty in the robot's state, we use the policy that minimizes the next-stage expected entropy. This local policy will proceed until entropy has reached a local minimum, or a maximum number of stages has elapsed. To implement target seeking behavior for both the goal configuration and sampled configurations, we use a policy that minimizes the KL divergence to a sampled target, specified by a Dirac delta function centered on a state space target. We will use this policy to draw the robot towards the goal as well as other sampled target points, to aid in obstacle avoidance. This policy executes until progress can no longer be made, or again, a pre-specified maximum number of stages elapses.

In general, these strategies cannot be computed in closed form, but numerical approaches such as Monte Carlo simulation can be applied. In this paper, we have computed the policies using a numerical brute force approach. However, one can often do better by considering the approximated structure of a belief, e.g., a set of particles combined with kernel functions to represent the next-stage expected pdf. A interesting direction of future research will be considering general principles by which one can use this structure, i.e., mixture of kernels, to approximate information-theoretic heuristic policies in a efficient manner.

B. Optimization

The POMDP planning objective can be stated as computing the optimal policy and the cost under that policy. A common tool for performing this optimization is the *policy value function* $V : \mathcal{P}_b \times \Pi \to \mathbb{R}$, where the value function encodes the expected cost-to-go from every belief in the belief space. Thus, $V(b, \pi)$ is the expected cumulative cost if the process starts from b using π. The function V satisfies

$$V(b, \pi) = \mathrm{E}_{\boldsymbol{y}} \left[l(b, u_\pi) + V(\phi_{\boldsymbol{y}, u_\pi} b, \pi) \right] \tag{3}$$

where l is the one stage cost function and $\phi_{y,u_\pi} b$ is the belief generated by moving one stage forward from b with the control from policy π. In principle, this equation could be used to synthesize an optimal policy using policy or value iteration [29]. Unfortunately, this procedure is typically too expensive to compute in practice when computing optimal controls one stage at a time.

We instead modify (3) to allow computation of the value function over the evolution of a local policy, and use a policy improvement algorithm to optimize the switching law of our policy. Let the cost-to-go of an individual sample path for t stages be denoted by j_t, and let $\bar{t}$ denote the next switching time of the switched policy for a particular sample path. Define $\alpha : \mathcal{P}_b \to \Psi$ to be the switching law that chooses the next local policy to use based on the current belief. Then, (3) can be rewritten

$$\begin{aligned} V(b, \overline{\pi}) &= \mathrm{E} \left[j_{\bar{t}}(\boldsymbol{b}_{\bar{t}}, \pi_{\alpha(b)}) + V(\boldsymbol{b}_{\bar{t}}, \overline{\pi}) \right] \\ &= J_{\alpha(b)}(b, \pi_{\alpha(b)}) + \int_{\mathcal{P}_b} \beta_{\bar{t}}(b_{\bar{t}}) V(b_{\bar{t}}, \overline{\pi}) \; db_{\bar{t}} \end{aligned} \tag{4}$$

where $J_{\alpha(b)}(b, \pi_{\alpha(b)})$ denotes the expected cost-to-go until the next switch and he functional $\beta_{\bar{t}}(b_{\bar{t}})$ is a pdf over the belief where the next switch occurs.

The two most computationally expensive parts of the optimization procedure are simulating the system to compute the effect of using a particular policy, and performing the backup procedure over a very large search graph. Rewriting (3) as (4) is useful because, if we can compute $J_{\alpha(b)}(b, \pi_{\alpha(b)})$ and $\beta_{\bar{t}}(b_{\bar{t}})$, we can build a data structure that represents a much coarser view of the system's evolution. This form is also particularly important because we can separate approximation of the system evolution from computing the value function. Thus, our gains in performance come firstly from reducing the complexity of the search graph over which we optimize by considering system evolution at a coarser level, but also from being able to utilize approximation methods that are standard in the robotics community, e.g., particle filters.

A natural value iteration-like algorithm is characterized by satisfying the equation

$$V(b, \overline{\pi}^{i+1}) = \min_{\psi \in \Psi} \left\{ J_\psi(b, \pi_\psi) + \int_{\mathcal{P}_b} \beta_{\bar{t}}(b) V(b, \overline{\pi}^{i+1}) db \right\} \tag{5}$$

for every $b \in \mathcal{P}_b$ and then choosing the optimal switching law

$$\alpha^{i+1}(b) = \arg\min_{\psi \in \Psi} \left\{ J_\psi(b, \pi_\psi) + \int_{\mathcal{P}_b} \beta_{\bar{t}}(b) V(b, \overline{\pi}^{i+1}) db \right\} \tag{6}$$

where α^i and $\overline{\pi}^i$ refer to the switching law and switched policy at the i^{th} planning iteration[3]. Since we minimize over ψ, we are essentially choosing the best local policy to be switched to at every belief and using that policy in the switching law for $\overline{\pi}$. Unfortunately, this procedure is computationally infeasible. In the next section, we discuss a sampling-based, anytime approach to approximate this algorithm.

IV. Approximation Algorithms

In a real implementation, we must approximate the method of (5)-(6) in several ways Firstly, we turn to a particle filtering approximation [4] of the belief. A standard particle filter approximates a function with a finite number of weighted support points. However, for some operations, e.g. computing collision probability, differential entropy, and KL divergence, we require a continuous probability distribution. To reconcile these two techniques, we convert that set of particles to a continuous approximation of the belief using regularization kernels [4]. Another problem is computing the exact cost and hyperbelief evolution of the POMDP for the duration of a local policy. This operation requires exponentially increasing resources as the number of stages for which the local policy executes increases. We can approximate these quantities using sequential Monte Carlo simulation or hyper-particle filtering [30]. Finally, we cannot compute expansions (defined in Section IV-B) and value for every belief in a continuous space. We again use a sampling-based method to slowly vary the policy, and improve it via an anytime approach. The remainder of this section addresses the approaches discussed briefly here.

A. Filtering

We use a modified version of a standard particle filter algorithm to track collision free trajectories of the system, and simultaneously estimate the probability of collision. Thus, we modify the canonical algorithm to estimate probability of collision, given an approximated pdf over the state space, and then resample from the collision free portion of that pdf. The steps of the method are shown in Algorithm 1.

We begin by taking an approximate belief $\hat{b}_t$ as input, which is represented as a set of particles. Each particle $(x_t^{(i)}, w_t^{(i)})$ contains a state and associated weight. The first task is to approximate Bayesian prediction by propagating existing particles through the system's process model and to estimate the probability the system will collide with workspace obstacles. Because the set of particles approximates a continuous distribution, but all the probability mass is located on a finite number of support points, we have observed that integrating this density over a projection onto $\mathcal{Q}_{\text{obst}}$, the subset of the configuration space where collisions occur, is often a poor approximation of the probability of collision unless the process model is sampled with an extremely

[3]Previously, we have used subscripts to notate time stage indices of the process (and conceptually related quantities). Here, we will use superscripts to denote the evolution of the policy with respect to iterations of the optimization algorithm.

Algorithm 1: Filtering Algorithm

Input: $\hat{b}_t := \{(x_t^{(i)}, w_t^{(i)})|_{i=1}^N\}$, u_t, y_t
Output: $\hat{b}_{t+1} := \{(x_{t+1}^{(i)}, w_{t+1}^{(i)}|_{i=1}^N)\}$, p_{collide}

// Propagate process model probability distribution
$\hat{b}_{t+1|t} \leftarrow \emptyset$
foreach $(x_t^{(i)}, w_t^{(i)})$ *in* $\hat{b}_t$ **do**
  $\bar{x}_{t+1|t}^{(i)} \leftarrow f(x_t^{(i)}, u_t, \mathrm{E}[\boldsymbol{n}_t])$
  $\hat{b} \leftarrow \hat{b} \cup \{(\bar{x}_{t+1|t}^{(i)}, w_t^{(i)}\}$
// Evaluate collisions and sample process noise
$p_{\text{collide}} \leftarrow 0$, $\hat{b}_{t+1|t} \leftarrow \emptyset$
foreach $(\bar{x}_{t+1|t}^{(i)}, \bar{w}_{t+1|t}^{(i)})$ *in* $\hat{b}$ **do**
  $\Gamma_{\boldsymbol{q}_{t|t+1}}(\cdot) \leftarrow$ projection of $\kappa(\bar{x}_{t+1|t}^{(i)}, u_t, \cdot)$ onto $\mathcal{Q}$
  $c \leftarrow \int_{\mathcal{Q}_{\text{obst}}} \Gamma_{\boldsymbol{q}_{t|t+1}}(q)dq$
  $p_{\text{collide}} \leftarrow p_{\text{collide}} + \bar{w}_{t+1|t}^{(i)} \cdot c$
  $w_{t+1|t}^{(i)} = \bar{w}_{t+1|t}^{(i)} \cdot (1-c)$
  Sample $x_{t+1|t}^{(i)}$ according to $\kappa(\bar{x}_{t+1|t}^{(i)}, u_t, \cdot)$
  while $x_{t+1|t}^{(i)}$ *is in collision* **do**
    Sample $x_{t+1|t}^{(i)}$ according to $\kappa(\bar{x}_{t+1|t}^{(i)}, u_t, \cdot)$
  $\hat{b}_{t+1|t} \leftarrow \hat{b}_{t+1|t} \cup \{(x_{t+1|t}^{(i)}, w_{t+1|t}^{(i)})\}$
// Update using observation model
foreach $(x_{t+1|t}^{(i)}, w_{t+1|t}^{(i)})$ *in* $\hat{b}_{t+1|t}$ **do**
  $w_{t+1|t}^{(i)} \leftarrow w_{t+1|t}^{(i)} \cdot \mathrm{P}\left[y_t | x_{t+1|t}^{(i)}\right]$
// Resample to avoid degeneracy
$\hat{b}_{t+1} \leftarrow \emptyset$
for *i = 1:N* **do**
  Sample $x_{t+1}^{(i)}$ from $\hat{b}_{t+1|t}$ w.p. $w_{t+1|t}^{(i)}$
  $\hat{b}_{t+1} \leftarrow \hat{b}_{t+1} \cup \{(x_{t+1}^{(i)}, \frac{1}{N})\}$

large number of particles (and then condensed down to a number of particles appropriate for filtering). Alternatively, we have had increased success by transforming the particle set to a continuous approximation of the pdf using kernel functions. Given a particle set $\hat{b} := \{(x^{(i)}, w^{(i)})|_{i=1}^N\}$, the approximate pdf is

$$\hat{\mathrm{f}}_{\boldsymbol{x}_{t|t+1}}(\hat{b}, u, x) = \sum_{i=1}^{N} \kappa(x^{(i)}, u, x) \cdot w^{(i)}$$

where $\kappa(\overline{x}, u, \cdot)$ is a kernel representing the one-step transition pdf of the system. Ideally, $\kappa(\overline{x}, u, \cdot)$ is the exact pdf of the random vector that is the result of the transformation $f(\overline{x}, u, \boldsymbol{n}_t)$. It is important to note that κ will only meet the requirements of a true regularization kernel under certain fairly general assumptions on the transition probabilities of the robot system model.

We can project $\hat{\mathrm{f}}_{\boldsymbol{x}_{t|t+1}}(b_{t|t+1}, u_t, \cdot)$ onto the configuration space and integrate over $\mathcal{Q}_{\text{obst}}$. In practice, $\hat{\mathrm{f}}_{\boldsymbol{x}_{t|t+1}}(b_{t|t+1}, \cdot)$ is an approximation and the integration will be computed numerically, so we have only an approximate probability of collision. We then re-weight each particle, removing from it the portion of the particle's weight corresponding to configurations in collision with obstacles. We then resample each particle from within the collision free portion of its kernel function. This leaves us with a predicted estimate of the robot's state from the set of collision free trajectories. The rest of the filtering procedure is standard.

B. Expansion Approximation

We define an *expansion* to be the transformation

$$\vartheta : \mathcal{P}_b \times \Psi \rightarrow \mathbb{R}^+ \times \mathcal{P}_\beta$$

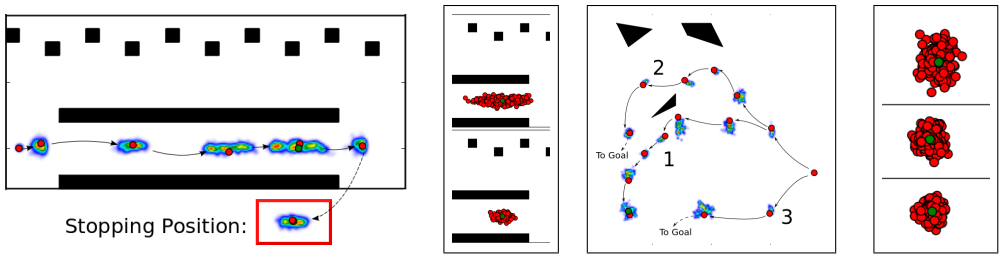
that maps every $b \in \mathcal{P}_b$ and $\psi \in \Psi$ to the $J_\psi(b, \pi)$ and $\beta_{\bar{t}}$ thats describes the evolution of the POMDP. Determining the values of an expansion exactly is not practical, since the value of the hyperbelief at a point is a continuous functional over an infinite dimensional space, and it typically cannot be represented with a finite number of parameters. Furthermore, there is an exponentially growing number of observations that must be considered as the number of stages to termination increases. Thus, we turn to a sampling-based approximation of the hyperbelief.

To curb the exponential growth of support points in $\mathcal{P}_\beta$, we represent β_t with an approximated $\hat{\beta}_t$ with a finite number of weighted support points. To approximate J_ψ and $\beta_{\bar{t}}$, either sequential Monte Carlo or particle filtering on the belief space, e.g., hyper-particle filtering [30], can be used. Sequential MC may be particularly useful if the approximation will be computed incrementally, rather than in a single shot. Hyper-particle filtering has benefits if the approximation will be computed using a fixed number of sample paths, as it can enforce some dissimilarity between parallel sample paths. Extra algorithmic bookkeeping must surround either or these algorithms as we must keep track of the likelihood of collision with obstacles along the multi-stage evolution and local policies may evolve over an unequal number of stages.

C. Planning Approximation Algorithm

Now that we have presented the techniques for filtering and approximating expansions, the underlying approach to main optimization algorithm will be discussed. For a discussion with more implementation details on the POMDP optimization algorithm, see [3]. The core idea is to implement a version of the algorithm specified by Equations (5)-(6) that can be computed in an incremental, sampled manner.

We store of the set of beliefs where a switch may occur in the set $\mathcal{B}^i$ (with initial condition $\mathcal{B}^0 = \{b_0\}$), and a collection of sets of expansions started from each belief $\mathcal{E}^i(b)$. At iteration $i+1$, we sample a belief from $\mathcal{B}^i$ and an expansion starting from b not currently explored in $\mathcal{E}^i(b)$. We then compute an approximation of ϑ and store the values in a graph structure, where the nodes correspond to beliefs, edges correspond to sample paths, and bundles of edges correspond to expansions. The edges are augmented with information on likelihood of the sample path the edge represents, and probability of collision along the sample path.



(a) (b) (c) (d)

Fig. 1. Plots of Planned Paths and Stopping Points for Sample Path Simulations

New stopping beliefs for this expansion are added to $\mathcal{B}^{i+1}$ and the new expansion to $\mathcal{E}^{i+1}(b)$. This data structure can be used to optimize the switching law of the switched policy since it represents the evolution of the POMDP process, given particular policy decisions (switching laws) are chosen. We optimize over the graph using dynamic programming. Bellman-Ford, or a variety of other graph search algorithms can also be used. The switching function is possibly improved, and the sampling algorithm continues iterating in this fashion until an external signal is sent to stop optimization.

At some point, optimization will terminate and the robot will use the switched policy. A switching function is constructed by mapping every $b \in \mathcal{B}$ to a local policy in Ψ. The robot then operates by choosing the belief in the graph closest to the current filtered belief of the robot. The robot chooses among the local policies sampled at this point by using the one corresponding to the expansion with the lowest expected cost. This local policy is used until its stopping condition is met. The same procedure is again performed until the robot reaches a belief during a switching operation where the terminal cost is smaller than the expected cost of any expansion emanating from that point. At this point, the robot stops.

V. Experiments

We use a simple example to present simulation results that demonstrate the planner is computing intuitively useful policies We consider a point robot in a planar environment, characterized by the process model

$$x_{t+1} = x_t + u_t + \boldsymbol{n}_t$$

where u_t is bounded, and $\{\boldsymbol{n}_t\}$ is an iid sequence of random vectors drawn from a mean zero, truncated normal distribution. We will consider several nonlinear sensing models that are models of range sensors.

Consider the example environment shown in Figure 1(a)-(b). In this simulation, we consider a system with a fixed orientation range sensor, oriented along the $+y$ axis (the "up" direction) of the plot. The sensor reports a noisy distance bearing to the closest obstacle. The noise is mean zero and Gaussian, and the variance increases as the distance to the obstacle the robot is measuring increases. Since there is little localization information from the sensor once entering between the two large obstacles, the robot (displayed in red) in Figure 1(a) is able to decrease uncertainty by moving past the goal (displayed in green), localizing at the edge of the obstacle, and then returning to the goal[4]. The heat map displayed around the true position of the robot is a plot of the continuous belief approximation (pdf of the robot's state) used at that stage as feedback. Figure 1(b) shows a plot of stopping positions for 200 trials using the policy generated by our method (lower plot) versus a plot of 200 trials using the policy that moves directly to the goal. Clearly, the policy generated by our method is significantly better at placing the robot near the goal position. We can quantify this by examining the Frobenius norm of the sample covariance matrix of the stopping position of the trials, which is 0.132 for our method versus 4.19655 for the comparison policy. Another measure of the quality of the method is the average entropy of the filtered belief at the stopping time. Our method produced an average entropy of 7.428 versus 9.080.

We also present a second environment in Figure 1(c)-(d). This model used two range sensors that point along the $+x$ and $+y$ axes of the plot. Both sensors have fixed noise variance, but saturate at a maximum distance measurement. Results on this environment are plotted in Figure 1(c)-(d) similarly to manner used for Environment 1.

We discuss three types of sample path characteristically generated by three policies evaluated by the planner on a particular optimization trial. The policy labeled 3 in Figure 1(c) is the most straightforward route to the goal. However, little localization information is available on this route. The policy labeled 2 moves between the obstacles, and receives the most localization information. The average uncertainty along these paths tends to be smallest. However, the planning algorithm determines that enough information is available localizing on the boundaries of the bottommost obstacle. This in concert with the smaller distance traveled from the last point where a discontinuity in the observation model is present makes the policy labeled 1 the policy our method predicts as optimum. Sample path simulations confirm this, and Figure 1(d) shows a plot of stopping positions for 200 trials for each policy, with policy 1 on the bottom, 2 in the middle, and 3 on top. The Frobenius norm of the sample covariance matrix and average entropies for policies 3, 2, and 1 were (0.228, 7.684), (0.0597, 7.382), and (0.026, 7.038), respectively.

Planning times vary significantly based on the environments and system models tested, and less significantly from

[4]The final belief state of the robot in Figure 1(a) is broken out of the plot so it can be examined separately from the first time the robot passes the goal.

experiment to experiment mainly due to randomized sampling. The optimization algorithm, executing on a standard desktop workstation, frequently found good policies within a matter of minutes. Though we make no claims about convergence to an optimal policy, the paths shown above were characteristic of paths generated by policies found on nearly all planning runs on these problems, i.e., the algorithm consistently produced paths that demonstrated the same high level behavior.

VI. Discussion and Future Directions

We have presented a minimum uncertainty planner for stochastic robot models. We use a POMDP framework and an optimization routine that performs a coarse optimization using a switched policy. The local modes of the switched policy are guided by information-theoretic quantities, which heuristically find paths that may lead to a belief-feedback policy that minimizes a criterion based on the uncertainty of the robot at the goal and probability of collision along the path to the goal. Though the optimization procedure is expensive and cannot be performed in real time on today's workstation hardware, we have demonstrated that good policies can often be computed on the scale of minutes on typical desktop hardware. Furthermore, although formulated in this paper as a single-query planning method, a multiple-query planning method can also be formulated which could potentially drastically speed up individual planning queries.

Our goal is to optimize a switching law over the belief space, which is continuous and infinite dimensional. We condense nearby beliefs to a single belief point for the purpose of filling volume within the domain of the switched policy. This approximation works well in most cases, but can be improved by the addition of sensitivity functions along the hyperbelief trajectory. This would essentially convert a computation that provides a point-wise view of the value function to a view of a small piece of the value function. This process can be thought of as constructing "tubes" around simulated trajectories inside which the system will remain with high likelihood at execution time. Efficient computation of sensitivity functions and integration into the algorithm is an additional avenue of future investigation.

References

[1] R. He, E. Brunskill, and N. Roy, "PUMA: Planning under uncertainty with macro-actions," in *Proceedings of the American Association for Artificial Intelligence*, 2010, pp. 1089–1095.

[2] H. Kurniawati, Y. Du, D. Hsu, and W. Lee, "Motion planning under uncertainty for robotic tasks with long time horizons," in *International Symposium on Robotics Research*, 2009.

[3] S. Candido, J. Davidson, and S. Hutchinson, "Exploiting domain knowledge in planning for uncertain robot systems modeled as POMDPs," in *IEEE International Conference on Robotics and Automation*, Anchorage, AK, USA, May 2010, pp. 3596–3603.

[4] A. Doucet, N. de Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice*. New York, NY: Springer Verlag, 2001.

[5] G. Chirikjian, *Stochastic Models, Information Theory, and Lie Groups, Volume I: Classical Results and Geometric Methods*. Boston, MA: Birkhauser, 2009.

[6] N. Roy and S. Thrun, "Coastal navigation with mobile robots," *Advances in Neural Information Processing Systems*, vol. 12, no. 12, pp. 1043–1049, 1999.

[7] H. Feder, J. Leonard, and C. Smith, "Adaptive mobile robot navigation and mapping," *International Journal of Robotics Research*, vol. 18, no. 7, p. 650, 1999.

[8] A. Makarenko, S. Williams, F. Bourgault, and H. Durrant-Whyte, "An experiment in integrated exploration," in *IEEE International Conference on Intelligent Robots and Systems*, 2002, pp. 534–539.

[9] R. He, S. Prentice, and N. Roy, "Planning in information space for a quadrotor helicopter in a gps-denied environment," in *IEEE International Conference on Robotics and Automation*, 2008.

[10] M. Apaydin, D. Brutlag, C. Guestrin, D. Hsu, J. Latombe, and C. Varm, "Stochastic roadmap simulation: An efficient representation and algorithm for analyzing molecular motion," *Journal of Computational Biology*, vol. 10, pp. 257–281, 2003.

[11] R. Alterovitz, T. Simeon, and K. Goldberg, "The stochastic motion roadmap: A sampling framework for planning with Markov motion uncertainty," in *Robotics: Science and Systems*, 2007, pp. 246–253.

[12] P. Missiuro and N. Roy, "Adapting probabilistic roadmaps to handle uncertain maps," in *IEEE International Conference on Robotics and Automation*, 2006, pp. 1261–1267.

[13] N. Melchior and R. Simmons, "Particle RRT for path planning with uncertainty," in *IEEE International Conference on Robotics and Automation*, 2007, pp. 1617–1624.

[14] S. Prentice and N. Roy, "The belief roadmap: Efficient planning in linear pomdps by factoring the covariance," in *International Symposium on Robotics Research*, 2007.

[15] J. Davidson and S. Hutchinson, "A sampling hyperbelief optimization technique for stochastic systems," in *Workshop on the Algorithmic Foundations of Robotics*, 2009, pp. 217–231.

[16] R. He, A. Bachrach, and N. Roy, "Efficient planning under uncertainty for a target-tracking micro-aerial vehicle," in *IEEE International Conference on Robotics and Automation*, 2010.

[17] W. Burgard, D. Fox, and S. Thrun, "Active mobile robot localization by entropy minimization," in *Proceedings of the Second Euromicro Workshop on Advanced Mobile Robots*, 1997, pp. 155–162.

[18] N. Roy, W. Burgard, D. Fox, and S. Thrun, "Coastal navigation: Mobile robot navigation with uncertainty in dynamic environments," in *IEEE International Conference on Robotics and Automation*, 1999, pp. 35–40.

[19] R. Smallwood and E. Sondik, "The optimal control of partially observable Markov processes over a finite horizon," *Operations Research*, pp. 1071–1088, 1973.

[20] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

[21] S. Thrun, "Monte Carlo POMDPs," *Advances in Neural Information Processing Systems*, pp. 1064–1070, 2000.

[22] T. Smith and R. Simmons, "Point-based POMDP algorithms: Improved analysis and implementation," in *Proceedings of Uncertainty in Artificial Intelligence*, 2005, pp. 542–555.

[23] J. Pineau, G. Gordon, and S. Thrun, "Point-based value iteration: An anytime algorithm for POMDPs," in *Proceedings of the International Joint Conference on Artificial Intelligence*, vol. 18, 2003, pp. 1025–1032.

[24] H. Kurniawati, D. Hsu, and W. Lee, "SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces," in *Robotics: Science and Systems*, 2008, pp. 65–72.

[25] M. Spaan and N. Vlassis, "Perseus: Randomized point-based value iteration for POMDPs," *Journal of Artificial Intelligence Research*, vol. 24, pp. 195–220, 2005.

[26] K. J. Astrom, "Optimal control of Markov decision processes with incomplete state estimation," *Journal of Mathematical Analysis and Applications*, vol. 10, pp. 174–205, 1965.

[27] S. Meyn and R. L. Tweedie, *Markov Chains and Stochastic Stability, Second Edition*. New York, NY: Cambridge University Press, 2009.

[28] T. Cover and J. Thomas, *Elements of information theory*. New York, NY: WILEY, 2006.

[29] D. P. Bertsekas, *Dynamic Programming and Optimal Control, Second Edition*. Belmont, MA: Athena Scientific, 1995.

[30] J. Davidson and S. Hutchinson, "Hyper-particle filtering for stochastic systems," in *IEEE International Conference on Robotics and Automation*, 2008, pp. 2770–2777.