Motion Planning under Uncertainty using Differential Dynamic Programming in Belief Space

Jur van den Berg, Sachin Patil, and Ron Alterovitz

Abstract We present an approach to motion planning under motion and sensing uncertainty, formally described as a continuous partially-observable Markov decision process (POMDP). Our approach is designed for non-linear dynamics and observation models, and follows the general POMDP solution framework in which we represent beliefs by Gaussian distributions, approximate the belief dynamics using an extended Kalman filter (EKF), and represent the value function by a quadratic function that is valid in the vicinity of a nominal trajectory through belief space. Using a variant of differential dynamic programming, our approach iterates with second-order convergence towards a linear control policy over the belief space that is locally-optimal with respect to a user-defined cost function. Unlike previous work, our approach does not assume maximum-likelihood observations, does not assume fixed estimator or control gains, takes into account obstacles in the environment, and does not require discretization of the belief space. The running time of the algorithm is polynomial in the dimension of the state space. We demonstrate the potential of our approach in several continuous partially-observable planning domains with obstacles for robots with non-linear dynamics and observation models.

1 Introduction

Motion planning under uncertainty, or belief-space planning, has received considerable interest in the robotics community over the past decade. The objective is to plan a path (or rather a control policy) for a robot in partially-observable state spaces with spatially varying degrees of motion and sensing uncertainty, such that the expected cost (as defined by a user-specified cost-function) is minimized. Optimal solutions lead the robot through regions of the state space where the most information on the state is gained through sensing and the least information is lost due to motion un-

The authors are with the Department of Computer Science, University of North Carolina at Chapel Hill, USA. E-mail: {berg, sachin, ron}@cs.unc.edu.

certainty in order to maximize, for instance, the probability of reaching a specified goal location while avoiding collisions with obstacles. This problem is formally described as a partially-observable Markov decision process (POMDP), on which a large body of work is available in the literature.

Solutions to POMDPs are known to be extremely complex [17], since they attempt to compute a control policy over the *belief space*, which in the most general formulation is an infinite-dimensional space of all possible probability distributions over the (finite-dimensional) state space. Solutions based on discrete or discretized state and action spaces are inherently subject to the "curse of dimensionality", and have only been successfully applied to very small and low-dimensional state spaces.

In this paper, we present a method to approximate a locally optimal solution to the POMDP problem with continuous state and action spaces and non-linear dynamics and observation models, where we assume a belief can be represented by a Gaussian distribution. Our approach uses a variant of differential dynamic programming to perform value iteration, where the value function is approximated using a quadratization around a nominal trajectory, and the belief dynamics is approximated using an extended Kalman filter. The result is a linear control policy over the belief space that is valid in the vicinity of the nominal trajectory. By executing the control policy, a new nominal trajectory is created around which a new control policy is constructed. This process continues with second-order convergence towards a locally-optimal solution to the POMDP problem. Unlike general POMDP solvers that have an exponential running time, our approach does not rely on discretizations and has a running time that is polynomial in the dimension of the state space.

Our approach builds off of and generalizes a series of previous works that have addressed the same problem of creating applicable approximations to the POMDP problem. Our work combines and extends ideas from previous work in order to overcome their key limitations. In particular, our approach (i) does not assume maximum-likelihood observations, (ii) does not assume fixed estimator or control gains, (iii) does not require discretizations of the state and action spaces, (iv) runs in polynomial time, (v) takes into account obstacles, and (vi) converges towards a locally-optimal control policy given an initial nominal trajectory. We do assume that the dynamics and observation models and cost functions are sufficiently smooth, and that the belief about the state of the robot is well described by only its mean and its variance. We show the potential of our approach in several illustrative partially-observable domains containing obstacles for robots with non-linear dynamics and observation models.

2 Previous Work

Partially observable Markov decision processes (POMDPs) [22] provide a principled mathematical framework for planning under uncertainty in partially-observable environments. They are known to be of extreme complexity [17], and can only be directly applied to problems with small and low-dimensional state spaces [14]. Re-

cently, several POMDP algorithms have been developed that use approximate value iteration with point-based updates [1, 15, 18, 16]. These have been shown to scale up to medium-sized domains. However, they rely on discretizing the state space or the action space, making them inevitably subject to the "curse of dimensionality". The methods of [21, 3, 8, 5] handle continuous state and action spaces, but maintain a global (discrete) representation of the value function over the belief space. In contrast, our approach is continuous and approximates the value function in parametric form only in the regions of the belief space that are relevant to solving the problem, allowing for a running time polynomial in the dimension of the state.

Another class of works, to which our method is directly related, assume a linear-quadratic Gaussian (LQG) framework to find approximately locally optimal feedback policies. In the basic LQG derivation [2], motion and sensing uncertainty have no impact on the resulting policy. As shown in [23], the LQG framework can be extended such that it accounts for state and control dependent motion noise, but still implicitly assumes full observation (or an independent estimator) of the state. Several approaches have been proposed to include partial and noisy observations such that the controller will actively choose actions to gain information about the state. Belief roadmaps [20] and icLQG [9] combine an iterative LQG approach with a roadmap, but this approach does not compute a (locally) optimal solution. The approaches of [19, 6, 7] incorporate the variance into an augmented state and use the LQG framework to find a locally-optimal control policy. However, these approaches assume *maximum-likelihood observations* to make the belief propagation deterministic. LQG-MP [24] removes this assumption, but only evaluates the probability of success of a given trajectory, rather than constructing an optimal one. Belief trees [4] overcome this limitation by combining a variant of LQG-MP with RRT* to find an optimal trajectory through belief space. Vitus and Tomlin [25] propose an alternative solution that involves solving a chance constrained optimal control problem. However, these approaches does not solve a POMDP as they assume fixed control gains along each section of the trajectory independent of the context. The work of [13] takes into account state and control dependent motion and observation noise by an interleaved iteration of the estimator and the controller. This approach does not allow for obstacles and converges towards a locally-optimal controller that assumes fixed estimator gains. Our approach combines and generalizes these approaches as it does not assume maximum-likelihood observations, does not assume fixed control or estimator gains, and takes into account the existence of obstacles in the environment to compute locally-optimal policies that maximize the probability of reaching a goal location while avoiding collisions.

3 Preliminaries and Definitions

We begin by defining POMDPs in their most general formulation (following [22]). Then, we specifically state the instance of the problem we discuss in this paper.

3.1 General POMDPs

Let $\mathcal{X} \subset \mathbb{R}^n$ be the space of all possible states $\mathbf{x}$ of the robot, $\mathcal{U} \subset \mathbb{R}^m$ be the space of all possible control inputs $\mathbf{u}$ of the robot, and $\mathcal{Z} \in \mathbb{R}^k$ be the space of all possible sensor measurements $\mathbf{z}$ the robot may receive. General POMDPs take as input a stochastic dynamics and observation model, here given in probabilistic notation:

$$\mathbf{x}_{t+1} \sim p[\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t], \qquad \mathbf{z}_t \sim p[\mathbf{z}_t|\mathbf{x}_t], \tag{1}$$

where $\mathbf{x}_t \in \mathcal{X}$, $\mathbf{u}_t \in \mathcal{U}$, and $\mathbf{z}_t \in \mathcal{Z}$ are the robot's state, control input, and received measurement at stage t, respectively.

The *belief* $b[\mathbf{x}_t]$ of the robot is defined as the distribution of the state $\mathbf{x}_t$ given all past control inputs and sensor measurements:

$$b[\mathbf{x}_t] = p[\mathbf{x}_t|\mathbf{u}_0, \ldots, \mathbf{u}_{t-1}, \mathbf{z}_1, \ldots, \mathbf{z}_t]. \tag{2}$$

Given a control input $\mathbf{u}_t$ and a measurement $\mathbf{z}_{t+1}$, the belief is propagated using Bayesian filtering:

$$b[\mathbf{x}_{t+1}] = \eta\, p[\mathbf{z}_{t+1}|\mathbf{x}_{t+1}] \int p[\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t]\, b[\mathbf{x}_t]\, d\mathbf{x}_t, \tag{3}$$

where η is a normalizer independent of $\mathbf{x}_{t+1}$. Denoting belief $b[\mathbf{x}_t]$ by $\mathbf{b}_t$, and the space of all possible beliefs by $\mathcal{B} \subset \{\mathcal{X} \to [0,1]\}$, the *belief dynamics* defined by Eq. (3) can be written as a function $\beta : \mathcal{B} \times \mathcal{U} \times \mathcal{Z} \to \mathcal{B}$:

$$\mathbf{b}_{t+1} = \beta[\mathbf{b}_t, \mathbf{u}_t, \mathbf{z}_{t+1}]. \tag{4}$$

Now, the challenge of the POMDP problem is to find a control policy $\pi_t : \mathcal{B} \to \mathcal{U}$ for all $0 \leq t < \ell$, where ℓ is the time horizon, such that selecting the controls $\mathbf{u}_t = \pi_t[\mathbf{b}_t]$ minimizes the objective function:

$$\mathop{\mathrm{E}}_{\mathbf{z}_1,\ldots,\mathbf{z}_\ell}\Big[c_\ell[\mathbf{b}_\ell] + \sum_{t=0}^{\ell-1} c_t[\mathbf{b}_t, \mathbf{u}_t]\Big], \tag{5}$$

for given immediate cost functions c_ℓ and c_t. The expectation is taken given the stochastic nature of the measurements.

A general solution approach uses *value iteration* [22], a backward recursion procedure, to find the control policy π_t for each stage t:

$$v_\ell[\mathbf{b}_\ell] = c_\ell[\mathbf{b}_\ell] \tag{6}$$

$$v_t[\mathbf{b}_t] = \min_{\mathbf{u}_t}(c_t[\mathbf{b}_t, \mathbf{u}_t] + \mathop{\mathrm{E}}_{\mathbf{z}_{t+1}}\big[v_{t+1}[\beta[\mathbf{b}_t, \mathbf{u}_t, \mathbf{z}_{t+1}]]\big]) \tag{7}$$

$$\pi_t[\mathbf{b}_t] = \operatorname*{argmin}_{\mathbf{u}_t}(c_t[\mathbf{b}_t, \mathbf{u}_t] + \mathop{\mathrm{E}}_{\mathbf{z}_{t+1}}\big[v_{t+1}[\beta[\mathbf{b}_t, \mathbf{u}_t, \mathbf{z}_{t+1}]]\big]), \tag{8}$$

where $v_t[\mathbf{b}_t] : \mathcal{B} \to \mathbb{R}$ is called the value function at time t.

3.2 Problem Definition

The complexity of POMDPs stems from the fact that $\mathcal{B}$, the space of all beliefs, is infinite-dimensional, and that in general the value function cannot be expressed in parametric form. We address these challenges in our approach by representing beliefs by Gaussian distributions, approximating the belief dynamics using an extended Kalman filter, and approximating the value function by a quadratization around a nominal trajectory through the belief space.

Specifically, we assume we are given a (non-linear) stochastic dynamics and observation model, here given in state-transition notation:

$$\mathbf{x}_{t+1} = \mathbf{f}[\mathbf{x}_t, \mathbf{u}_t, \mathbf{m}_t], \qquad \mathbf{m}_t \sim \mathcal{N}[\mathbf{0}, I], \tag{9}$$

$$\mathbf{z}_t = \mathbf{h}[\mathbf{x}_t, \mathbf{n}_t], \qquad \mathbf{n}_t \sim \mathcal{N}[\mathbf{0}, I], \tag{10}$$

where $\mathbf{m}_t$ is the motion noise and $\mathbf{n}_t$ is the measurement noise, each drawn from an independent Gaussian distribution with (without loss of generality) zero mean and unit variance. Note that the motion and sensing uncertainty can be state and control input dependent through manipulations on $\mathbf{m}_t$ and $\mathbf{n}_t$ within the functions $\mathbf{f}$ and $\mathbf{h}$, respectively.

The belief, denoted $\mathbf{b}_t = (\hat{\mathbf{x}}_t, \Sigma_t)$, is assumed to be defined by the mean $\hat{\mathbf{x}}_t$ and variance Σ_t of a Gaussian distribution $\mathcal{N}[\hat{\mathbf{x}}_t, \Sigma_t]$ of the state $\mathbf{x}_t$. Similar to the general case, our objective is to find a control policy $\mathbf{u}_t = \pi_t[\mathbf{b}_t]$ that minimizes the cost function $\mathrm{E}\left[c_\ell[\mathbf{b}_\ell] + \sum_{t=0}^{\ell-1} c_t[\mathbf{b}_t, \mathbf{u}_t]\right]$. In our case, we assume in addition that the Hessian matrix $\frac{\partial^2 c_\ell}{\partial \mathbf{b} \partial \mathbf{b}}[\mathbf{b}]$ is positive-semidefinite for all $\mathbf{b}$, and that the Hessian matrix $\frac{\partial^2 c_t}{\partial \mathbf{u} \partial \mathbf{u}}[\mathbf{b}, \mathbf{u}]$ is positive-definite for all $\mathbf{b}$, $\mathbf{u}$, and t. Further, we assume that the initial belief $\mathbf{b}_0 = (\hat{\mathbf{x}}_0, \Sigma_0)$ is given.

4 Approach

To approximate a locally optimal solution to the Gaussian POMDP problem as formulated above, we follow the general solution approach as sketched in Section 3.1. First, we approximate the belief dynamics using an extended Kalman filter. Second, we approximate the value function using a quadratic function that is locally valid in the vicinity of a *nominal trajectory* though the belief space. We then use a variant of differential dynamic programming to perform the value iteration, which results in a linear control policy over the belief space that is locally optimal around the nominal trajectory. We then iteratively generate new nominal trajectories by executing the control policy, and repeat the process until convergence to a locally-optimal solution to the POMDP problem. We discuss each of these steps in this section, and analyze the running time of our algorithm.

4.1 Belief Dynamics and the Extended Kalman Filter

Given a current belief $\mathbf{b}_t = (\hat{\mathbf{x}}_t, \Sigma_t)$, a control input $\mathbf{u}_t$, and a measurement $\mathbf{z}_{t+1}$, we let the belief evolve using the *extended Kalman filter* (EKF). The EKF is widely used for state estimation of non-linear systems [26], and uses the first-order approximation that for any vector-valued function $\mathbf{f}[\mathbf{x}]$ of a stochastic variable $\mathbf{x}$ we have:

$$\mathrm{E}[\mathbf{f}[\mathbf{x}]] \approx \mathbf{f}[\mathrm{E}[\mathbf{x}]], \qquad \mathrm{Var}[\mathbf{f}[\mathbf{x}]] \approx \frac{\partial \mathbf{f}}{\partial \mathbf{x}}[\mathrm{E}[\mathbf{x}]] \cdot \mathrm{Var}[\mathbf{x}] \cdot \frac{\partial \mathbf{f}}{\partial \mathbf{x}}[\mathrm{E}[\mathbf{x}]]^T. \tag{11}$$

The EKF update equations are then given by:

$$\hat{\mathbf{x}}_{t+1} = \mathbf{f}[\hat{\mathbf{x}}_t, \mathbf{u}_t, \mathbf{0}] + K_t(\mathbf{z}_{t+1} - \mathbf{h}[\mathbf{f}[\hat{\mathbf{x}}_t, \mathbf{u}_t, \mathbf{0}], \mathbf{0}]), \tag{12}$$

$$\Sigma_{t+1} = (I - K_t H_t)\Gamma_t, \tag{13}$$

where

$$\Gamma_t = A_t \Sigma_t A_t^T + M_t M_t^T, \qquad A_t = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}[\hat{\mathbf{x}}_t, \mathbf{u}_t, \mathbf{0}], \qquad M_t = \frac{\partial \mathbf{f}}{\partial \mathbf{m}}[\hat{\mathbf{x}}_t, \mathbf{u}_t, \mathbf{0}],$$

$$K_t = \Gamma_t H_t^T (H_t \Gamma_t H_t^T + N_t N_t^T)^{-1}, \quad H_t = \frac{\partial \mathbf{h}}{\partial \mathbf{x}}[\mathbf{f}[\hat{\mathbf{x}}_t, \mathbf{u}_t, \mathbf{0}], \mathbf{0}], \quad N_t = \frac{\partial \mathbf{h}}{\partial \mathbf{n}}[\mathbf{f}[\hat{\mathbf{x}}_t, \mathbf{u}_t, \mathbf{0}], \mathbf{0}].$$

Note that all of these matrices are functions of $\mathbf{b}_t$ and $\mathbf{u}_t$. Equations (12) and (13) define the (non-linear) belief dynamics. The second term of Eq. (12), called the *innovation* term, depends on the measurement $\mathbf{z}_{t+1}$. Since the measurement is unknown in advance, the belief dynamics are *stochastic*. Using the assumptions of Eq. (11), the innovation term is distributed according to $\mathcal{N}[\mathbf{0}, K_t H_t \Gamma_t]$.

Defining $\mathbf{b}_t = \left[\begin{smallmatrix} \hat{\mathbf{x}}_t \\ \mathrm{vec}[\Sigma_t] \end{smallmatrix}\right]$ as a true vector, containing the mean $\hat{\mathbf{x}}_t$ and the columns of the variance Σ_t (obviously, in our implementation we exploit the symmetry of Σ_t to eliminate the redundancy), the belief dynamics are given by:

$$\mathbf{b}_{t+1} = \mathbf{g}[\mathbf{b}_t, \mathbf{u}_t] + \mathbf{w}_t, \qquad \mathbf{w}_t \sim \mathcal{N}[\mathbf{0}, W[\mathbf{b}_t, \mathbf{u}_t]], \tag{14}$$

where

$$\mathbf{g}[\mathbf{b}_t, \mathbf{u}_t] = \begin{bmatrix} \mathbf{f}[\hat{\mathbf{x}}_t, \mathbf{u}_t, \mathbf{0}] \\ \mathrm{vec}[(I - K_t H_t)\Gamma_t] \end{bmatrix}, \qquad W[\mathbf{b}_t, \mathbf{u}_t] = \begin{bmatrix} K_t H_t \Gamma_t & 0 \\ 0 & 0 \end{bmatrix}. \tag{15}$$

4.2 Value Iteration

We perform value iteration backward in time to find a locally optimal control policy using a variant of differential dynamic programming [10]. We approximate the value function $v_t[\mathbf{b}]$ as a quadratic function of the form

$$v_t[\mathbf{b}] = \frac{1}{2}\mathbf{b}^T S_t \mathbf{b} + \mathbf{b}^T \mathbf{s}_t + s_t, \tag{16}$$

with $S_t \geq 0$, that is approximately valid around a nominal trajectory in belief space $(\bar{\mathbf{b}}_0, \bar{\mathbf{u}}_0, \ldots, \bar{\mathbf{b}}_\ell, \bar{\mathbf{u}}_\ell)$, which we assume is given (we will discuss initialization and iterative convergence of the nominal trajectory to a locally optimal trajectory in the next subsection).

For the final time $t = \ell$, the value function v_ℓ is approximated by setting $S_\ell = \frac{\partial^2 c_\ell}{\partial \mathbf{b} \partial \mathbf{b}}[\bar{\mathbf{b}}_\ell]$, $\mathbf{s}_\ell = \frac{\partial c_\ell}{\partial \mathbf{b}}[\bar{\mathbf{b}}_\ell] - S_\ell \bar{\mathbf{b}}_\ell$, and $s_\ell = c_\ell[\bar{\mathbf{b}}_n] - \bar{\mathbf{b}}_\ell^T \mathbf{s}_\ell - \frac{1}{2}\bar{\mathbf{b}}_\ell^T S_\ell \bar{\mathbf{b}}_\ell$, which amounts to a second-order Taylor expansion of c_ℓ around the point $\bar{\mathbf{b}}_\ell$. The value functions and the control policies for the stages $\ell > t \geq 0$ are computed by backward recursion – following Eq. (7), we get:

$$\begin{aligned} v_t(\mathbf{b}) &= \min_{\mathbf{u}} \Big(c_t[\mathbf{b}, \mathbf{u}] + \mathrm{E}\left[v_{t+1}[\mathbf{g}[\mathbf{b}, \mathbf{u}] + \mathbf{w}_t] \right] \Big) \\ &= \min_{\mathbf{u}} \Big(c_t[\mathbf{b}, \mathbf{u}] + \frac{1}{2}\mathbf{g}[\mathbf{b}, \mathbf{u}]^T S_{t+1} \mathbf{g}[\mathbf{b}, \mathbf{u}] + \mathbf{g}[\mathbf{b}, \mathbf{u}]^T \mathbf{s}_{t+1} + s_{t+1} + \\ &\qquad \frac{1}{2} \operatorname{tr}\left[S_{t+1} W[\mathbf{b}, \mathbf{u}] \right] \Big) \qquad (17) \\ &= \min_{\mathbf{u}} (q_t[\mathbf{b}, \mathbf{u}]), \qquad (18) \end{aligned}$$

where $q_t[\mathbf{b}, \mathbf{u}]$ groups together the terms in Eq. (17). The trace-term in Eq. (17) follows from the fact that $\mathrm{E}[\mathbf{x}^T Q \mathbf{x}] = \mathrm{E}[\mathbf{x}]^T Q \mathrm{E}[\mathbf{x}] + \operatorname{tr}[Q \operatorname{Var}[\mathbf{x}]]$ for any stochastic variable $\mathbf{x}$. It is this term that ensures that the stochastic nature of the belief dynamics is accounted for in the value iteration. To approximate the optimal value of $\mathbf{u}$ as a function of $\mathbf{b}$ we take the second-order Taylor expansion of $q_t[\mathbf{b}, \mathbf{u}]$ in $(\bar{\mathbf{b}}_t, \bar{\mathbf{u}}_t)$:

$$v_t[\mathbf{b}] \approx \min_{\tilde{\mathbf{u}}} \big(\frac{1}{2} \begin{bmatrix} \tilde{\mathbf{b}} \\ \tilde{\mathbf{u}} \end{bmatrix}^T \begin{bmatrix} C_t & E_t^T \\ E_t & D_t \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{b}} \\ \tilde{\mathbf{u}} \end{bmatrix} + \begin{bmatrix} \tilde{\mathbf{b}} \\ \tilde{\mathbf{u}} \end{bmatrix}^T \begin{bmatrix} \mathbf{c}_t \\ \mathbf{d}_t \end{bmatrix} + e_t \big), \tag{19}$$

where $\tilde{\mathbf{u}} = \mathbf{u} - \bar{\mathbf{u}}_t$ and $\tilde{\mathbf{b}} = \mathbf{b} - \bar{\mathbf{b}}_t$, and

$$\begin{aligned} C_t &= \frac{\partial^2 q_t}{\partial \mathbf{b} \partial \mathbf{b}}[\bar{\mathbf{b}}_t, \bar{\mathbf{u}}_t], & D_t &= \frac{\partial^2 q_t}{\partial \mathbf{u} \partial \mathbf{u}}[\bar{\mathbf{b}}_t, \bar{\mathbf{u}}_t], & E_t &= \frac{\partial^2 q_t}{\partial \mathbf{u} \partial \mathbf{b}}[\bar{\mathbf{b}}_t, \bar{\mathbf{u}}_t], \\ \mathbf{c}_t^T &= \frac{\partial q_t}{\partial \mathbf{b}}[\bar{\mathbf{b}}_t, \bar{\mathbf{u}}_t], & \mathbf{d}_t^T &= \frac{\partial q_t}{\partial \mathbf{u}}[\bar{\mathbf{b}}_t, \bar{\mathbf{u}}_t], & e_t &= q_t[\bar{\mathbf{b}}_t, \bar{\mathbf{u}}_t]. \end{aligned} \tag{20}$$

Equation (19) is then solved by expanding the terms, taking the derivative with respect to $\tilde{\mathbf{u}}$ and equating to 0 (for $\tilde{\mathbf{u}}$ to be actually a minimum, D_t must be positive-definite – we will discuss this issue in Section 4.4). We then get the solution:

$$\tilde{\mathbf{u}} = -D_t^{-1} E_t \tilde{\mathbf{b}} - D_t^{-1} \mathbf{d}_t. \tag{21}$$

Hence, the control policy for time t is linear and given by:

$$\mathbf{u}_t = \pi_t(\mathbf{b}_t) = L_t \mathbf{b}_t + \mathbf{l}_t, \quad L_t = -D_t^{-1} E_t, \quad \mathbf{l}_t = -D_t^{-1}(\mathbf{d}_t - E_t \bar{\mathbf{b}}_t) + \bar{\mathbf{u}}_t. \tag{22}$$

Filling Eq. (21) back into Eq. (19) gives the value function $v_t[\mathbf{b}]$ as a function of only $\mathbf{b}$ in the form of Eq. (16). Expanding and collecting terms gives:

$$S_t = C_t - E_t^T D_t^{-1} E_t, \tag{23}$$

$$\mathbf{s}_t = \mathbf{c}_t - E_t^T D_t^{-1} \mathbf{d}_t - S_t \bar{\mathbf{b}}_t, \tag{24}$$

$$s_t = e_t - \frac{1}{2} \mathbf{d}_t^T D_t^{-1} \mathbf{d}_t - \bar{\mathbf{b}}_t^T \mathbf{s}_t - \frac{1}{2} \bar{\mathbf{b}}_t^T S_t \bar{\mathbf{b}}_t. \tag{25}$$

This recursion then continues by computing a control policy for stage $t-1$.

4.3 Iteration to a Locally-Optimal Control Policy

The above value iteration gives a control policy that is valid in the vicinity of the given nominal trajectory. To let the control policy converge to a local optimum, we iteratively update the nominal trajectory using the most recent control policy, as in differential dynamic programming [10]. Given the initial belief $\mathbf{b}_0 = (\hat{\mathbf{x}}_0, \Sigma_0)$, and an (arbitrary) initial series of control inputs $\bar{\mathbf{u}}_0^{(0)}, \ldots, \bar{\mathbf{u}}_{\ell-1}^{(0)}$, which can be obtained using RRT motion planning [11], for instance, let the initial control policy be given by $L_t^{(0)} = 0$ and $\mathbf{l}_t^{(0)} = \bar{\mathbf{u}}_t^{(0)}$ for all t. We then compute the nominal trajectory $(\bar{\mathbf{b}}_t^{(i)}, \bar{\mathbf{u}}^{(i)})$ of the i'th iteration (starting with $i = 0$) by forward integrating the control policy in the deterministic (zero-noise) belief dynamics:

$$\bar{\mathbf{b}}_0^{(i)} = \mathbf{b}_0, \qquad \bar{\mathbf{u}}_t^{(i)} = L_t^{(i)} \bar{\mathbf{b}}_t^{(i)} + \mathbf{l}_t^{(i)}, \qquad \bar{\mathbf{b}}_{t+1}^{(i)} = \mathbf{g}[\bar{\mathbf{b}}_t^{(i)}, \bar{\mathbf{u}}_t^{(i)}], \tag{26}$$

Then, using the value iteration procedure as described above given the nominal trajectory of the i'th iteration, we find the control policy, i.e. the matrices $L_t^{(i+1)}$ and vectors $\mathbf{l}_t^{(i+1)}$ for the $i+1$'th iteration. We then recompute a nominal trajectory using Eq. (26), and reiterate. This lets the control policy converge to a locally optimal trajectory with a second-order convergence rate [12].

4.4 Ensuring Convergence

To ensure that the above algorithm in fact converges to a locally-optimal control policy, the algorithm must be augmented with some subtle but important changes, common to approaches based on differential dynamic programming [10, 12, 27].

First, to make sure that in each step of the value iteration we actually minimize the value function (rather than maximizing it), matrix D_t must be positive definite, which is not the case in general. In addition, to ensure that the trajectory iteration converges to a local optimum, the matrices S_t as well as the entire matrix $\left[\begin{smallmatrix} C_t & E_t^T \\ E_t & D_t \end{smallmatrix}\right]$ of Eq. (19) must be positive-semidefinite [12]. To enforce these requirements, while retaining the most amount of second-order information about the value function, we proceed as follows. Let $R_t = \frac{\partial^2 c_t}{\partial \mathbf{u} \partial \mathbf{u}}[\bar{\mathbf{b}}_t, \bar{\mathbf{u}}_t]$, which is by definition positive-definite (see Section 3.2). Note that $c_t[\mathbf{b}, \mathbf{u}]$ is one of the terms of $q_t[\mathbf{b}, \mathbf{u}]$, of which D_t

is the Hessian with respect to $\mathbf{u}$. Then we decompose the matrix $Q_t = \begin{bmatrix} C_t & E_t^T \\ E_t & D_t - R_t \end{bmatrix}$ into $Q_t = V \Lambda V^T$ such that Λ is a diagonal matrix containing Q_t's eigenvalues. Second, we construct $\tilde{\Lambda}$ by setting all negative elements of Λ to 0, and construct $\tilde{Q}_t = V \tilde{\Lambda} V^T$. Matrix $\tilde{Q}_t$ is now a positive-semidefinite version of Q_t. Subsequently, we let $\begin{bmatrix} \tilde{C}_t & \tilde{E}_t^T \\ \tilde{E}_t & \tilde{D}_t \end{bmatrix} = \tilde{Q}_t + \begin{bmatrix} 0 & 0 \\ 0 & R_t \end{bmatrix}$, such that $\tilde{D}_t$ is positive definite, and $\begin{bmatrix} \tilde{C}_t & \tilde{E}_t^T \\ \tilde{E}_t & \tilde{D}_t \end{bmatrix}$ is positive-semidefinite. Now, the matrices C_t, D_t, and E_t are replaced by $\tilde{C}_t$, $\tilde{D}_t$, and $\tilde{E}_t$, respectively, in Eqs. (21)-(25). If these changes are made, it is automatically guaranteed that the matrices S_t are positive-semidefinite. Note that S_ℓ is positive-semidefinite too, since $\frac{\partial^2 c_\ell}{\partial \mathbf{b} \partial \mathbf{b}}[\bar{\mathbf{b}}_\ell]$ is positive-definite by definition (see Section 3.2).

These changes do not affect the second-order convergence characteristic of the algorithm. However, as with Newton's method, this second order convergence is only achieved if the current nominal trajectory is already close to the locally-optimal trajectory. If the current nominal trajectory is "far away" from the local optimum, using second-order approaches may overshoot local-minima, which significantly slows down convergence, or even results in divergence. To address this issue, we make an additional change to the algorithm, following [27]. We limit the increment to the control policy by adding a parameter ε to Eq. (21): $\tilde{\mathbf{u}} = -\tilde{D}_t^{-1} \tilde{E}_t \tilde{\mathbf{b}} - \varepsilon \tilde{D}_t^{-1} \mathbf{d}_t$. Initially, $\varepsilon = 1$, but each time a new control policy is computed that creates a trajectory with higher cost than the previous nominal trajectory (the cost of a trajectory is evaluated using value iteration as above without updating the control policy), the new trajectory is rejected, and ε is reduced by a factor $\beta < 1$, and the iteration continues. When a new trajectory is accepted, ε is reset to 1. This change is equivalent to using backtracking line search to limit the step size in Newton's method and guarantees convergence to a locally-optimal control policy [27].

4.5 Running Time Analysis

Let us analyze the running time of our algorithm. The dimension of the state is n, and we assume for the sake of analysis that the dimension of the control inputs and the measurements are $O(n)$. As the belief contains the covariance matrix of the state, the dimension of a belief is $O(n^2)$. Hence, the matrix C_t of Eq. (20) contains $O(n^4)$ entries. Each of these entries is computed using numerical differentiation, and requires multiplying matrices of size $n \times n$ within the belief propagation, taking $O(n^3)$ time. Hence, the total running time of a single step of the value iteration is $O(n^7)$. A complete cycle of value iteration takes ℓ steps (ℓ being the time horizon), bringing the complexity to $O(\ell n^7)$. The number of such cycles needed to obtain convergence cannot be expressed in terms of n or ℓ, but as noted before, our algorithm converges with a second-order rate to a local optimum.

5 Environments with Obstacles

We presented our approach above for general immediate cost functions $c_\ell[\mathbf{b}]$ and $c_t[\mathbf{b}, \mathbf{u}]$. In typical LQG-style cost functions, the existence of obstacles in the en-

vironment is not incorporated, while we may want to minimize the probability of colliding with them. We incorporate obstacles into the cost functions as follows.

Let $\mathcal{O} \subset \mathcal{X}$ be the region of the state space that is occupied by obstacles. Given a belief $\mathbf{b}_t = (\hat{\mathbf{x}}_t, \Sigma_t)$, the probability of colliding with an obstacle is given by the integral over $\mathcal{O}$ of the probability-density function of $\mathcal{N}[\hat{\mathbf{x}}_t, \Sigma_t]$. As described in [24], this probability can be approximated by using a collision-checker to compute the number $\sigma[\mathbf{b}_t]$ of standard-deviations one may deviate from the mean before an obstacle is hit. A lower-bound on the probability of *not* colliding is then given by $\gamma[n/2, \sigma[\mathbf{b}_t]^2/2]$, where γ is the regularized gamma function, and n the dimension of the state. A lower-bound on the total probability of not colliding along a trajectory is subsequently computed as $\prod_{t=0}^{\ell-1} \gamma[n/2, \sigma[\mathbf{b}_t]^2/2]$, and this number should be *maximized*. To fit this objective within the minimizing and additive nature of the POMDP objective function, we note that maximizing a product is equivalent to minimizing the sum of the negative logarithms of the factors. Hence, we add to $c_t[\mathbf{b}, \mathbf{u}]$ the term $f[\sigma[\mathbf{b}]] = -\log \gamma[n/2, \sigma[\mathbf{b}]^2/2]$ to account for the probability of colliding with obstacles (note that $f[\sigma[\mathbf{b}]] \geq 0$), potentially multiplied by a scaling factor to allow trading-off with respect to other costs (such as the magnitude of the control input).

While the above approach works well, it should be noted that in order to compute the Hessian of $q_t[\mathbf{b}, \mathbf{u}]$ at $\bar{\mathbf{b}}_t$ (as is done in Eq. (20)), a total of $O(n^4)$ collision-checks with respect to the obstacles need to be performed, since the obstacle term $f[\sigma[\mathbf{b}]]$ is part of $q_t[\mathbf{b}, \mathbf{u}]$. As this can be prohibitively costly, we can instead approximate the Hessian of $f[\sigma[\mathbf{b}]]$ using linearizations, which involves only $O(n^2)$ collision checks. To this end, let us approximate $f[\sigma]$ by a second-order Taylor expansion:

$$f[\sigma[\mathbf{b}]] \approx \frac{1}{2} a (\sigma[\mathbf{b}] - \sigma[\bar{\mathbf{b}}_t])^2 + b(\sigma[\mathbf{b}] - \sigma[\bar{\mathbf{b}}_t]) + f[\sigma[\bar{\mathbf{b}}_t]], \tag{27}$$

where $a = \frac{\partial^2 f}{\partial \sigma \partial \sigma}[\sigma[\bar{\mathbf{b}}_t]]$ and $b = \frac{\partial f}{\partial \sigma}[\sigma[\bar{\mathbf{b}}_t]]$ (note that this requires only one collision-check). Now, we approximate $(\sigma[\mathbf{b}] - \sigma[\bar{\mathbf{b}}_t])$ using a first-order Taylor expansion:

$$\sigma[\mathbf{b}] \approx (\mathbf{b} - \bar{\mathbf{b}}_t)^T \mathbf{a} + \sigma[\bar{\mathbf{b}}_t] \quad \Leftrightarrow \quad \sigma[\mathbf{b}] - \sigma[\bar{\mathbf{b}}_t] \approx \tilde{\mathbf{b}}^T \mathbf{a}, \tag{28}$$

where $\mathbf{a}^T = \frac{\partial \sigma}{\partial \mathbf{b}}[\bar{\mathbf{b}}_t]$ (note that this requires $O(n^2)$ collision-checks). By substituting Eq. (28) in Eq. (27), we get

$$f[\sigma[\mathbf{b}]] \approx \frac{1}{2} \tilde{\mathbf{b}}^T (a \mathbf{a}\mathbf{a}^T) \tilde{\mathbf{b}} + \tilde{\mathbf{b}}^T (b\mathbf{a}) + f[\sigma[\bar{\mathbf{b}}_t]]. \tag{29}$$

Hence, $a\mathbf{a}\mathbf{a}^T$ is an approximate Hessian of the obstacle term $f[\sigma[\mathbf{b}]]$ of $q_t[\mathbf{b}, \mathbf{u}]$ that requires only $O(n^2)$ collision-checks to compute.

6 Results

We evaluate our approach in two scenarios with obstacles: a point robot with linear dynamics that is navigating in a 2-D light-dark environment (adapted from Bry and Roy [4]) and a non-holonomic car-like robot with second-order dynamics moving in a partially-observable environment with spatially varying sensing capabilities.

Our method takes as input a collision-free trajectory to the goal. A naïve trajectory computed using an uncertainty-unaware planner might stray very close to the obstacles in the environment and accumulates considerable uncertainty during execution. We show that our method improves the input trajectory to compute a locally-optimal trajectory and a corresponding control policy that safely guides the robot to the goal, even in the presence of large motion and measurement noise.

6.1 Light-Dark Environment

We consider the case of a point robot moving in a 2-D environment with obstacles shown in Fig. 1. The robot localizes itself using measurements from sensors in the environment, the reliability of which varies continuously as a function of the horizontal coordinate of the robot's position. The experiment is set up such that the robot needs to move away from the goal in order to better localize itself before moving through the narrow passage and reaching the goal.

We assume the following linear dynamics model with control-dependent noise:

$$\mathbf{x}_{t+1} = \mathbf{f}[\mathbf{x}_t, \mathbf{u}_t, \mathbf{m}_t] = \mathbf{x}_t + \mathbf{u}_t + M[\mathbf{u}_t] \cdot \mathbf{m}_t, \tag{30}$$

where the state $\mathbf{x}_t = (x, y) \in \mathbb{R}^2$ is the robot's position, the control input $\mathbf{u}_t \in \mathbb{R}^2$ is the robot's velocity, and the matrix $M[\mathbf{u}_t]$ scales the motion noise $\mathbf{m}_t$ proportional to the control input $\mathbf{u}_t$. We assume the following linear observation model with state-dependent noise:

$$\mathbf{z}_t = \mathbf{h}[\mathbf{x}_t, \mathbf{n}_t] = \mathbf{x}_t + N[\mathbf{x}_t] \cdot \mathbf{n}_t, \tag{31}$$

where the measurement vector $\mathbf{z}_t \in \mathbb{R}^2$ consists of noisy measurements of the robot's position and the matrix $N[\mathbf{x}_t]$ scales the measurement noise based on a sigmoid function of the horizontal coordinate of the robot's position x (as shown in Fig. 1). The robot is able to obtain reliable measurements in the bright region of the environment, but the measurements become noisier as the robot moves in to the dark regions.

We use the following definitions of $c_\ell[\mathbf{b}_\ell]$ and $c_t[\mathbf{b}_t, \mathbf{u}_t]$ in the cost function to be minimized (Eq. (5)):

$$c_\ell[\mathbf{b}_\ell] = \hat{\mathbf{x}}_\ell^T Q_\ell \hat{\mathbf{x}}_\ell + \mathrm{tr}[Q_\ell \Sigma_\ell], \qquad c_t[\mathbf{b}_t, \mathbf{u}_t] = \mathbf{u}_t^T R_t \mathbf{u}_t + \mathrm{tr}[Q_t \Sigma_t] + f[\sigma[\mathbf{b}_t]], \tag{32}$$

where the term $\hat{\mathbf{x}}_\ell^T Q_\ell \hat{\mathbf{x}}_\ell + \mathrm{tr}[Q_\ell \Sigma_\ell] = \mathrm{E}[\mathbf{x}_\ell^T Q_\ell \mathbf{x}_\ell]$ encodes the final cost of arriving at the goal, $\mathbf{u}_t^T R_t \mathbf{u}_t$ penalizes the control effort along the trajectory, $\mathrm{tr}[Q_t \Sigma_t]$ penalizes the uncertainty, and $f[\sigma[\mathbf{b}_t]]$ encodes the obstacle cost term. We use recurring state and control cost matrices of $Q_t = I$ and $R_t = I$ and the final cost matrix, $Q_\ell = 10\ell I$ in our experiments.

Results and discussion: In our experiment, we provide a collision-free initial trajectory computed using an RRT planner [11] (Fig. 1(a)) as input to our method. The control policy convergence took 2.75 seconds on a 3.33 Ghz Intel® i7™ PC. Fig. 1(b) shows the nominal trajectory and associated beliefs of the solution computed by our method. The robot's trajectory visits the region of the environment with reliable sensing for better localization before moving through the narrow passage.

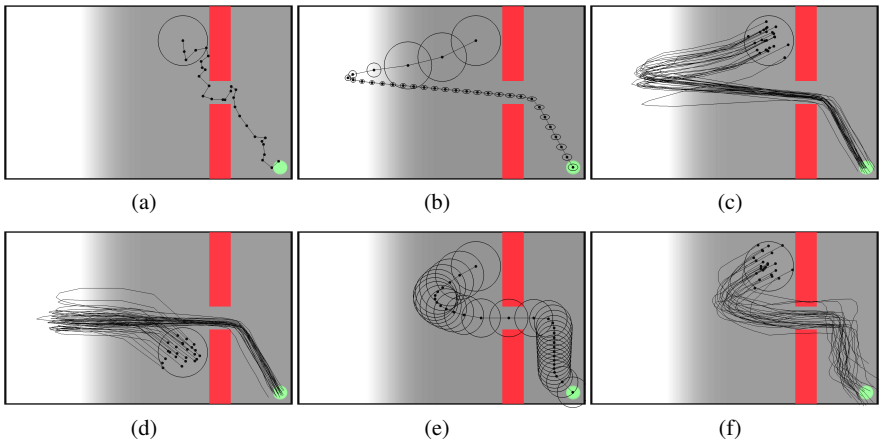

Fig. 1 Point robot moving in a 2-D environment with obstacles. (a) An initial collision-free trajectory is computed using an RRT planner. (b) Nominal trajectory and the associated beliefs of solution computed using our method. The robot moves away from the goal to better localize itself before reaching the goal with significantly reduced uncertainty. Execution traces of the robot's true state starting from the initial belief (c) and a different initial belief (d), while following the computed control policy. (e) Nominal trajectory computed by ignoring the innovation term in the belief dynamics. The optimization is unable to progress sufficiently to the region of the environment with reliable sensing, resulting in considerable uncertainty in the robot state near the obstacles and at the goal. (f) Execution traces of the robot's true state starting from the initial belief and ignoring the innovation term are much noisier as compared to the execution traces shown in (c).

We simulated the robot's execution of the computed control policy using the given dynamics and measurement models with synthetic noise. Fig. 1(c) shows the traces of the true state of the robot $\mathbf{x}$ across 25 simulations where the initial state of the robot $\mathbf{x}_0$ is sampled from the initial belief $\mathbf{b}_0$. We also initialized the robot state from a different initial belief to evaluate the robustness of the control policy. The 25 execution traces from these runs are shown in Fig. 1(d). Even if the initial belief is far away from the nominal trajectory, the control policy is able to safely guide the robot to the goal. We also evaluated our method quantitatively by computing the percentage of executions in which the robot was able to avoid obstacles across 1000 simulation executions for 10 random initial beliefs. In our experiments, in 93% (standard deviation: 3%) of the executions, the robot was able to safely traverse the narrow passage without colliding with obstacles.

Fig. 1(e) shows the nominal trajectory computed by ignoring the innovation term in Eq. (12), i.e. making the assumption that all future observations will obtain their maximum-likelihood measurement estimates. Under this assumption, the optimization is unable to progress sufficiently to the region of the environment with reliable sensing, which results in considerable uncertainty in the robot state near the obstacles and the goal. As expected, the execution traces from 25 simulations (Fig. 1(f)) are considerably noisier as compared to the execution traces obtained using our method. The expected cost of the solution found using our method is 19.9 units while the expected cost of a solution that ignores the innovation term is 143.0 units

for the parameters suggested above. This indicates that it is important to take into account the true belief of the robot while computing the control policy and ignoring the innovation term can lead to sub-optimal policies.

Our solution also agrees with the solution found by Bry and Roy [4] for this experiment. Our method directly optimizes the path rather than relying on RRT*, resulting in an order of magnitude faster computation times.

6.2 Non-holonomic car-like robot

We consider the case of a non-holonomic car-like robot navigating in a 2-D environment with obstacles shown in Fig. 2. We initialize our method with a collision-free trajectory to the goal which is computed using an RRT planner [11].

The state $\mathbf{x} = (x, y, \theta, v) \in \mathbb{R}^4$ of the robot consists of its position (x, y), its orientation θ and speed v. The control input vector $\mathbf{u} = (a, \phi)$ consists of an acceleration a and the steering wheel angle ϕ. The motion uncertainty is scaled by a constant matrix M. This gives the following non-linear dynamics model:

$$\mathbf{x}_{t+1} = \mathbf{f}[\mathbf{x}_t, \mathbf{u}_t, \mathbf{m}_t] = \begin{bmatrix} x_t + \tau v_t \cos\theta_t \\ y_t + \tau v_t \sin\theta_t \\ \theta_t + v_t \tan(\phi)/d \\ v_t + \tau a \end{bmatrix} + M\mathbf{m}_t, \tag{33}$$

where τ is the time step and d is the length of the car-like robot.

The robot localizes itself using signal measurements from two beacons b_1 and b_2 placed in the environment at locations $(\check{x}_1, \check{y}_1)$ and $(\check{x}_2, \check{y}_2)$ respectively. The strength of the signal decays quadratically with the distance to the beacon. The robot also measures its current speed using an on-board speedometer. The measurement uncertainty is scaled by a constant matrix N. This gives us the following non-linear observation model:

$$\mathbf{z}_t = \mathbf{h}[\mathbf{x}_t, \mathbf{n}_t] = \begin{bmatrix} 1/((x_t - \check{x}_1)^2 + (y_t - \check{y}_1)^2 + 1) \\ 1/((x_t - \check{x}_2)^2 + (y_t - \check{y}_2)^2 + 1) \\ v_t \end{bmatrix} + N\mathbf{n}_t, \tag{34}$$

where the observation vector $\mathbf{z}_t \in \mathbb{R}^3$ consists of two readings of signal strengths from the beacons and a speed measurement from the speedometer. Fig. 2(a) visually illustrates the quadratic decay in the beacon signal strengths in the environment. The robot is able to obtain very reliable measurements in the bright regions of the environment, but the measurements become noisier as the robot moves in to the dark regions due to the decreased signal-to-noise ratio.

We consider a similar cost function as Eq. (32) for this experiment and use recurring state and control input cost matrices of $Q_t = I$ and $R_t = I$ and the final cost matrix, $Q_\ell = 10\ell I$, where ℓ is the number of sections along the initial RRT trajectory.

Results and discussion: The control policy computation took 15.3 seconds on a 3.33 Ghz Intel® i7™ PC. Fig. 2(b) shows the nominal trajectory and associated beliefs of the solution computed by our method. The robot moves closer to the beacon

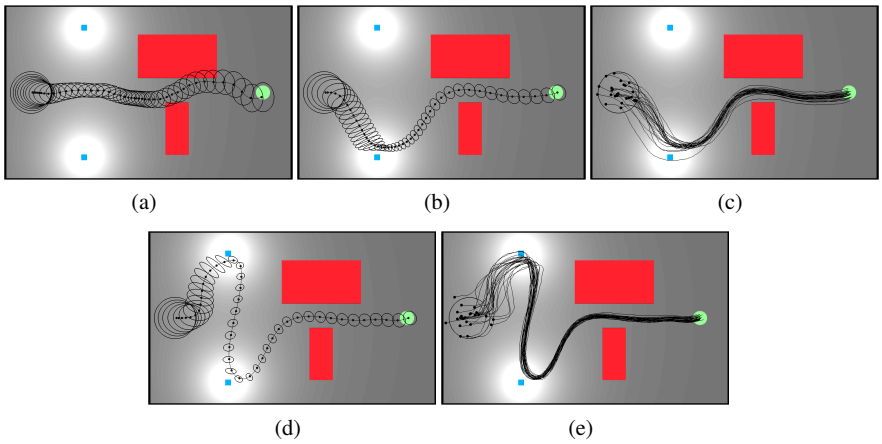

Fig. 2 A car-like robot with second order dynamics moving in a 2-D environment with obstacles. The robot obtains measurements from two beacons (marked by blue squares) and an on-board speedometer. (a) An initial collision-free trajectory is computed using an RRT planner. (b) Nominal trajectory computed using our method. Notice how the car-like robot localizes itself by moving closer to the beacon before reaching the goal. (c) Execution traces of the robot's true state starting from the initial belief for the control policy computed in (b). The jaggedness of the paths is due to the large amount of artificial motion and measurement noise introduced in the simulation. The control policy is safely able to guide the robot to the goal, in spite of the large amount of noise. (d) Nominal trajectory computed by varying the cost matrices ($Q_t = 10I$). The robot tries to reduce the uncertainty in its state by visiting both the beacons. (e) Execution traces of the robot's true state starting from the initial belief for the control policies computed in (d).

for better localization before reaching the goal. In contrast to the initial trajectory (Fig. 2(a)), the locally-optimal trajectory also moves away from the obstacles and takes a safer path to the goal.

Fig. 2(c) shows the traces of the true state of the robot $\mathbf{x}$ across 25 simulations where the initial state of the robot $\mathbf{x}_0$ is sampled from the a-priori belief $\mathbf{b}_0$. We evaluated our method quantitatively by computing the percentage of executions in which the robot was able to avoid obstacles across 1000 simulation executions where the initial state of the robot $\mathbf{x}_0$ is sampled from the a-priori belief $\mathbf{b}_0$. In our experiments, 96% of the executions were collision-free. The results indicate that the computed control policy is safely able to guide the robot to the goal region in spite of the large amount of motion and measurement noise encountered during execution.

The cost matrices Q_t and R_t determine the relative weighting between minimizing uncertainty in the robot state and minimizing control effort in the objective function. Fig. 2(d) shows the nominal trajectory of the solution computed by changing the cost matrix $Q_t = 10I$. Notice that the trajectory visits both the beacons for better localization and minimizing uncertainty, at the expense of additional control effort. We simulated 1000 execution runs using the new control policy, of which 98% were collision-free.

Fig. 3 shows the nominal trajectory when a different initial trajectory is provided as input to our method. The presence of obstacles in the environment forces our

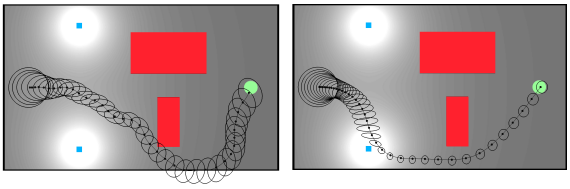

Fig. 3 A different initial trajectory results in a different locally-optimal solution. Our method is able to improve trajectories within a single homotopy class.

method to locally optimize trajectories within a single homotopy class. In contrast to considering a large number of initial candidate trajectories for evaluation as in LQG-MP [24], our method would only require trajectory initializations within each homotopy class to compute a globally optimal solution.

7 Conclusion and Future Work

We presented a general approach to motion planning under uncertainty by computing locally-optimal solutions to continuous POMDP problems in environments with obstacles. Our approach generalizes earlier work on Gaussian-based POMDPs by removing several key limiting assumptions, and overcomes the main drawback of approaches based on discretizations of the state space by having a running time that is polynomial ($O(n^7)$) rather than exponential in the dimension of the state.

Our approach has several limitations. First, we represent beliefs using Gaussian distributions. This may not be an acceptable approximation in some applications, for instance ones where multi-modal beliefs are expected to appear. However, the class of problems where Gaussian distributions are applicable is large, as is proven by the widespread use of the extended Kalman filter for state estimation, for instance in mobile robotics. Our approach should be applicable in any such application. Second, we require the dynamics, observation, and cost functions to be smooth, since our method relies on gradients to iterate towards a locally-optimal solution. Our approach would therefore not work directly in some experimental domains shown in previous work where there are abrupt boundaries between sensing regimes (e.g. inside or outside the field of view of a camera).

Subjects of ongoing and future work include improving the running time of the algorithm. While $O(n^7)$ is polynomial, it may still be too high for robots with complex dynamics and high-dimensional state spaces. The running time can potentially be brought down to $O(n^5)$ if we can avoid computing Hessian matrices. A derivation of our approach based on a quasi-Newton variant of differential dynamic programming [27] may achieve this, and may allow for the direct application of our approach to real-world domains involving complex dynamics such as autonomous quadrotor flight, medical needle steering, or even manipulation of deformable tissue.

Acknowledgments

This research was supported in part by the National Science Foundation (NSF) under grant #IIS-0905344 and by the National Institutes of Health (NIH) under grant #R21EB011628.

References

1. H. Bai, D. Hsu, W. Lee, V. Ngo. Monte Carlo value iteration for continuous state POMDPs. *Workshop on the Algorithmic Foundations of Robotics*, 2010.
2. D. Bertsekas. *Dynamic programming and optimal control*. Athena Scientific, 2001.
3. A. Brooks, A. Makarendo, S. Williams, H. Durrant-Whyte. Parametric POMDPs for planning in continuous state spaces. *Robotics and Autonomous Systems* 54(11):887–897, 2006.
4. A. Bry, N. Roy. Rapidly-exploring random belief trees for motion planning under uncertainty. *IEEE Int. Conf. on Robotics and Automation*, 2011.
5. S. Candido, S. Hutchinson. Minimum Uncertainty Robot Navigation Using Information-guided POMDP Planning. *IEEE Int. Conf. on Robotics and Automation*, 2011.
6. N. Du Toit, J. Burdick. Robotic motion planning in dynamic, cluttered, uncertain environments. *IEEE Int. Conf. on Robotics and Automation*, 2010.
7. T. Erez, W. D. Smart. A Scalable Method for Solving High-Dimensional Continuous POMDPs Using Local Approximation. *Conf. on Uncertainty in Artificial Intelligence*, 2010.
8. K. Hauser. Randomized belief-space replanning in partially-observable continuous spaces. *Workshop on the Algorithmic Foundations of Robotics*, 2010.
9. V. Huynh, N. Roy. icLQG: combining local and global optimization for control in information space. *IEEE Int. Conf. on Robotics and Automation*, 2009.
10. D. Jacobson, D. Mayne. *Differential Dynamic Programming*. American Elsevier Publishing Company, Inc., New York, 1970.
11. S. LaValle, J. Kuffner. Randomized kinodynamic planning. *Int. Journal on Robotics Research* 20(5):378–400, 2001.
12. L.-Z. Liao, C. Shoemaker. Convergence in unconstrained discrete-time differential dynamic programming. *IEEE Trans. on Automatic Control* 36(6):692–706, 1991.
13. W. Li, E. Todorov. Iterative linearization methods for approximately optimal control and estimation of non-linear stochastic system. *Int. Journal of Control* 80(9):1439–1453, 2007.
14. L. Kaelbling, M. Littman, A. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101(1-2):99–134, 1998.
15. H. Kurniawati, D. Hsu, W. Lee. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. *Robotics: Science and Systems*, 2008. configuration spaces. *IEEE Trans. on Robotics and Automation* 12:4(566–580), 1996.
16. S. Ong, S. Png, D. Hsu, W. Lee. Planning under uncertainty for robotic tasks with mixed observability. *Int. J. of Robotics Research* 29(8):1053–1068, 2010.
17. C. Papadimitriou, J. Tsisiklis. The complexity of Markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.
18. J. Porta, N. Vlassis, M. Spaan, P. Poupart. Point-based value iteration for continuous POMDPs. *Journal of Machine Learning Research* 7:2329–2367, 2006.
19. R. Platt, R. Tedrake, L. Kaelbling, T. Lozano-Perez. Belief space planning assuming maximum likelihood observations. *Robotics: Science and Systems*, 2010.
20. S. Prentice, N. Roy. The belief roadmap: Efficient planning in belief space by factoring the covariance. *Int. J. of Robotics Research* 28(1112):1448-1465, 2009.
21. S. Thrun. Monte Carlo POMDPs. *Advances in Neural Information Processing Systems*. The MIT Press, 2000.
22. S. Thrun, W. Burgard, D. Fox. *Probabilistic Robotics*, MIT Press, 2005.
23. E. Todorov, W. Li. A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems. *American Control Conference*, 2005.
24. J. van den Berg, P. Abbeel, K. Goldberg. LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information. *Robotics: Science and Systems*, 2010.
25. M. P. Vitus, C. J. Tomlin. Closed-Loop Belief Space Planning for Linear, Gaussian Systems. *IEEE Int. Conf. on Robotics and Automation*, 2011.
26. G. Welch, G. Bishop. An introduction to the Kalman filter. *Tech. Report TR 95-041*, University of North Carolina at Chapel Hill, 2006.
27. S. Yakowitz. Algorithms and computational techniques in differential dynamic programming. *Control and Dynamic Systems* 31:75–91, 1989.