Motion Planning under Uncertainty using Iterative Local Optimization in Belief Space

Jur van den Berg[1] Sachin Patil[2] Ron Alterovitz[2]
[1]*School of Computing, University of Utah, berg@cs.utah.edu.*
[2]*Dept. of Computer Science, University of North Carolina at Chapel Hill, {sachin, ron}@cs.unc.edu.*

Abstract

We present a new approach to motion planning under sensing and motion uncertainty by computing a locally optimal solution to a continuous partially observable Markov decision process (POMDP). Our approach represent beliefs (the distributions of the robot's state estimate) by Gaussian distributions and is applicable to robot systems with non-linear dynamics and observation models. The method follows the general POMDP solution framework in which we approximate the belief dynamics using an extended Kalman filter and represent the value function by a quadratic function that is valid in the vicinity of a nominal trajectory through belief space. Using a belief space variant of iterative LQG (iLQG), 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 state and action spaces. The running time of the algorithm is polynomial ($O[n^6]$) in the dimension n of the state space. We demonstrate the potential of our approach in simulation for holonomic and nonholonomic robots maneuvering through environments with obstacles with noisy and partial sensing and with non-linear dynamics and observation models.

1 Introduction

As a robot moves through an environment to accomplish a task, uncertainty may arise in (1) the robot's motion due to unmodeled or unpredictable external forces, and (2) the robot's sensing of its state due to noisy or incomplete sensor measurements. These forms of uncertainty are common in a variety of practical robotics tasks, including guiding aerial vehicles in turbulent conditions, maneuvering mobile robots in unfamiliar terrain, and robotically steering flexible medical needles to clinical targets in soft tissues. Explicitly considering motion and sensing uncertainty when computing motion plans can improve the quality of computed plans. The objective of motion planning under uncertainty is to plan motions for a robot such that the expected cost (as defined by a user-specified cost-function) is minimized. Optimal plans typically limit the information that is lost due to motion uncertainty and move the robot through regions of the state space where information on the state is gained. Optimal solutions will maximize, for instance, the probability of reaching a specified goal location while avoiding collisions with obstacles.

To fully consider the impact of uncertainty in motion and sensing, a motion planner should not merely compute a static path through the robot's configuration space but rather a control policy that defines the motion to perform given any current state information. A key challenge is that the robot often cannot directly observe its current state but instead estimates a distribution

over the set of possible states (i.e., its belief state) based on sensor measurements that are both noisy and partial (i.e., only a subset of the state vector can be sensed). The problem of computing a control policy over the space of belief states 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 [19], since the belief space (over which a control policy is to be computed) is in the most general formulation 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 that takes as input a feasible trajectory and improves it by computing a locally optimal trajectory and a corresponding control policy that together minimize the expected value of a user-specified cost metric in the presence of motion and sensing uncertainty. To accomplish this, our method computes a locally optimal solution to a 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. This POMDP formulation is applicable to a wide range of robot motion planning problems. Our approach uses a belief space variant of iterative linear-quadratic Gaussian (iLQG) 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 (any non-linear Gaussian filter can in fact be used). 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 ($O[n^6]$) in the dimension n of the state space.

Our approach combines, generalizes, and overcomes the limitations of previous work that has addressed the same problem of creating applicable approximations to the POMDP problem. Most previous work on POMDPs assumes *maximum-likelihood observations* to enable or simplify computing a control policy. This assumption has no formal justification, yet seems to produce reasonable results. Our approach does *not* assume maximum-likelihood observations, but can relatively easily be adapted such that it does. We use this to study the impact of the maximum-likelihood observation assumption on the resulting control policies and discuss the impact on plans computed using iterative local optimization. Our results indicate that not making this assumption results, on average, in better control policies (i.e., they have lower expected cost).

Furthermore, our approach does not assume fixed estimator or control gains, and takes into account obstacles in the environment. 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 scenarios involving robots with non-linear dynamics and observation models moving through environments containing obstacles and relying on limited and partial sensing.

2 Previous Work

Partially observable Markov decision processes (POMDPs) [24] provide a principled mathematical framework for planning under uncertainty. They are known to be of extreme complexity [19], and can only be directly applied to problems with small and low-dimensional state spaces [16]. Recently, several POMDP algorithms have been developed that use approximate value iteration with point-based updates [1, 17, 20, 18]. 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 [23, 4, 9, 6] 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, assumes a linear-quadratic Gaussian (LQG) framework to find locally optimal feedback policies. In the basic LQG derivation [2], motion and sensing uncertainty have no impact on the resulting policy. As shown in [25], 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 [22] and icLQG [10] combine an iterative LQG approach with a roadmap, but this approach does not result in (locally) optimal solutions. The approaches of [21, 7, 8] are similar to ours and incorporate the variance into an augmented state and use the LQG framework to find a locally optimal control policy. The main difference is that these approaches assume *maximum-likelihood observations* to make the belief propagation deterministic. LQG-MP [26] removes this assumption, but only evaluates the probability of success of a given trajectory, rather than constructing an optimal one. Belief trees [5] overcome this limitation by combining a variant of LQG-MP with RRT* to find an optimal trajectory through belief space. A great advantage of this approach is that it finds a globally optimal solution. Vitus and Tomlin [31] propose an alternative solution that involves solving a chance constrained optimal control problem. However, these approaches do not really solve a POMDP as they assume fixed control gains along each section of the trajectory independent of the context. The work of [15] takes into account state and control dependent motion and observation noise by an interleaved iteration of the estimator and the controller, converging to a local optimum. While this approach is asymptotically faster than ours, it does not allow for obstacles in the environment and results in a controller that is optimal only under the assumption of 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 minimize the expected value of a user-defined cost function.

This paper is an extended version of a preliminary paper presented by the authors in [28], which used stochastic differential dynamic programming (sDDP) rather than iLQG for the value iteration, but otherwise presents the same global approach. Also, to improve numerical stability compared to [28], in this paper we use the principal square root of the variance, rather than the variance itself, in the definition of the belief. Qualitatively, iLQG is asymptotically faster than sDDP ($O[n^6]$ rather than $O[n^7]$) and numerically more stable (regularization of matrices to maintain positive-semidefiniteness of the value function is not necessary with iLQG). Our experimental results include a quantitative comparison between the two approaches.

3 Preliminaries and Definitions

We begin by defining POMDPs in their most general formulation (following [24]). 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 time step 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 $\mathcal{B} \subset \{\mathcal{X} \to \mathbb{R}\}$, 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 (i.e. the index of the final time step), 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 because the measurements are stochastic.

A general solution approach uses *value iteration* [24], a backward recursion procedure, to find the control policy π_t for each time step 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 step 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 $\mathbf{b}_t = (\hat{\mathbf{x}}_t, \sqrt{\Sigma_t})$, is assumed to be defined by the mean $\hat{\mathbf{x}}_t$ and the *principal square root* $\sqrt{\Sigma_t}$ of the variance Σ_t of a Gaussian distribution $\mathcal{N}[\hat{\mathbf{x}}_t, \Sigma_t]$ of the state $\mathbf{x}_t$. We use the square root for numerical robustness of the algorithm we present below. Similar to the general POMDP 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 positive-(semi)definiteness for the Hessian matrices of the immediate cost functions c_t:

$$\frac{\partial^2 c_\ell}{\partial \mathbf{b} \partial \mathbf{b}}[\mathbf{b}] \geq 0, \qquad \frac{\partial^2 c_t}{\partial \mathbf{u} \partial \mathbf{u}}[\mathbf{b}, \mathbf{u}] > 0, \qquad \begin{bmatrix} \frac{\partial^2 c_t}{\partial \mathbf{b} \partial \mathbf{b}}[\mathbf{b}, \mathbf{u}] & \frac{\partial^2 c_t}{\partial \mathbf{b} \partial \mathbf{u}}[\mathbf{b}, \mathbf{u}] \\ \frac{\partial^2 c_t}{\partial \mathbf{u} \partial \mathbf{b}}[\mathbf{b}, \mathbf{u}] & \frac{\partial^2 c_t}{\partial \mathbf{u} \partial \mathbf{u}}[\mathbf{b}, \mathbf{u}] \end{bmatrix} \geq 0, \tag{11}$$

for all $\mathbf{b}$, $\mathbf{u}$ and t. Further, we assume that the initial belief $\mathbf{b}_0 = (\hat{\mathbf{x}}_0, \sqrt{\Sigma_0})$ is given.

4 Approach

To compute 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 belief-space variant of iterative LQG 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 Bayesian Filter and Belief Dynamics

Given a current belief $\mathbf{b}_t = (\hat{\mathbf{x}}_t, \sqrt{\Sigma_t})$, a control input $\mathbf{u}_t$, and a measurement $\mathbf{z}_{t+1}$, the belief evolves using a *Bayesian filter*. We approximate the Bayesian filter by an extended Kalman filter (EKF), which is applicable to Gaussian beliefs (we note that any other non-linear Gaussian filter, such as the unscented Kalman filter [12], can be used as well). The EKF is widely used for state estimation of non-linear systems [32], and uses the first-order approximation that for any vector-valued function $\mathbf{f}[\mathbf{x}]$ of a stochastic variable $\mathbf{x}$ we have:

$$\mathrm{E}[\mathbf{f}[\mathbf{x}]] \approx \mathbf{f}[\mathrm{E}[\mathbf{x}]], \qquad \mathrm{Var}[\mathbf{f}[\mathbf{x}]] \approx \frac{\partial \mathbf{f}}{\partial \mathbf{x}}[\mathrm{E}[\mathbf{x}]] \cdot \mathrm{Var}[\mathbf{x}] \cdot \frac{\partial \mathbf{f}}{\partial \mathbf{x}}[\mathrm{E}[\mathbf{x}]]^T. \tag{12}$$

Given $\hat{\mathbf{x}}_t$ and $\sqrt{\Sigma_t}$ that define the current belief, the EKF update equations are then given by:

$$\hat{\mathbf{x}}_{t+1} = \mathbf{f}[\hat{\mathbf{x}}_t, \mathbf{u}_t, \mathbf{0}] + K_t(\mathbf{z}_{t+1} - \mathbf{h}[\mathbf{f}[\hat{\mathbf{x}}_t, \mathbf{u}_t, \mathbf{0}], \mathbf{0}]), \tag{13}$$

$$\sqrt{\Sigma_{t+1}} = \sqrt{\Gamma_t - K_t H_t \Gamma_t}, \tag{14}$$

where

$$\Gamma_t = A_t \sqrt{\Sigma_t} (A_t \sqrt{\Sigma_t})^T + M_t M_t^T, \quad A_t = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}[\hat{\mathbf{x}}_t, \mathbf{u}_t, \mathbf{0}], \qquad M_t = \frac{\partial \mathbf{f}}{\partial \mathbf{m}}[\hat{\mathbf{x}}_t, \mathbf{u}_t, \mathbf{0}], \tag{15}$$

$$K_t = \Gamma_t H_t^T (H_t \Gamma_t H_t^T + N_t N_t^T)^{-1}, \quad H_t = \frac{\partial \mathbf{h}}{\partial \mathbf{x}}[\mathbf{f}[\hat{\mathbf{x}}_t, \mathbf{u}_t, \mathbf{0}], \mathbf{0}], \quad N_t = \frac{\partial \mathbf{h}}{\partial \mathbf{n}}[\mathbf{f}[\hat{\mathbf{x}}_t, \mathbf{u}_t, \mathbf{0}], \mathbf{0}]. \tag{16}$$

Note that all of these matrices are functions of $\mathbf{b}_t$ and $\mathbf{u}_t$. Equations (13) and (14) define the (non-linear) belief dynamics. The second term of Eq. (13), 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 Eq. (10) and the assumptions of Eq. (12), the innovation term is distributed according to $\mathcal{N}[\mathbf{0}, K_t H_t \Gamma_t]$.

We define the belief $\mathbf{b}_t = \left[\begin{smallmatrix} \hat{\mathbf{x}}_t \\ \mathrm{vec}[\sqrt{\Sigma_t}] \end{smallmatrix}\right]$ as a true vector, containing the mean $\hat{\mathbf{x}}_t$ and the columns of $\sqrt{\Sigma_t}$. Obviously, in our implementation we exploit the symmetry of $\sqrt{\Sigma_t}$ to eliminate the redundancy. Then, the stochastic belief dynamics are given by:

$$\mathbf{b}_{t+1} = \mathbf{g}[\mathbf{b}_t, \mathbf{u}_t] + W[\mathbf{b}_t, \mathbf{u}_t]\mathbf{w}_t, \qquad \mathbf{w}_t \sim \mathcal{N}[\mathbf{0}, I_n], \tag{17}$$

where n is the dimension $\dim[\mathbf{x}]$ of the state, and:

$$\mathbf{g}[\mathbf{b}_t, \mathbf{u}_t] = \begin{bmatrix} \mathbf{f}[\hat{\mathbf{x}}_t, \mathbf{u}_t, \mathbf{0}] \\ \mathrm{vec}[\sqrt{\Gamma_t - K_t H_t \Gamma_t}] \end{bmatrix}, \qquad W[\mathbf{b}_t, \mathbf{u}_t] = \begin{bmatrix} \sqrt{K_t H_t \Gamma_t} \\ 0 \end{bmatrix}. \tag{18}$$

4.2 Value Iteration

We perform value iteration backward in time to find a locally optimal control policy. When using value iteration (dynamic programming) over discrete states one usually stores the value of each possible state. In the case of a continuous state this is not possible. Instead, we assume that we have an initial (nominal) trajectory given. For each time step t we calculate an approximation of the value function around the state the robot is in at time step t when following the nominal trajectory. As the value function at time step t depends on the value function at time step $t+1$, this is done in a backward iterative process starting at the final time step ℓ. Using the approximated value function, we can also calculate an optimal policy for each time step. Using this optimal policy we generate a new nominal trajectory by starting at the initial state and applying this optimal policy forward in time. The process in then repeated using the new nominal trajectory, and ultimately converges to a locally optimal solution.

We use a belief-space variant of iterative LQG [25] to perform the value iteration. We approximate the value function $v_t[\mathbf{b}]$ as a quadratic function that is approximately valid around a given nominal trajectory in belief space. Let the nominal trajectory be given as a series of beliefs and control inputs $(\bar{\mathbf{b}}_0, \bar{\mathbf{u}}_0, \ldots, \bar{\mathbf{b}}_{\ell-1}, \bar{\mathbf{u}}_{\ell-1}, \bar{\mathbf{b}}_\ell)$ such that $\bar{\mathbf{b}}_{t+1} = \mathbf{g}[\bar{\mathbf{b}}_t, \bar{\mathbf{u}}_t]$ for $t \in 0 \ldots \ell-1$ (we will discuss initialization and iterative convergence of the nominal trajectory to a locally optimal trajectory in the next subsection). The value function is then approximated as:

$$v_t[\mathbf{b}] \approx \frac{1}{2}(\mathbf{b} - \bar{\mathbf{b}}_t)^T S_t (\mathbf{b} - \bar{\mathbf{b}}_t) + (\mathbf{b} - \bar{\mathbf{b}}_t)^T \mathbf{s}_t + s_t, \tag{19}$$

with $S_t \geq 0$.

For the final time step $t = \ell$, the value function v_ℓ (see Eq. (6)) is approximated by setting

$$S_\ell = \frac{\partial^2 c_\ell}{\partial \mathbf{b} \partial \mathbf{b}}[\bar{\mathbf{b}}_\ell], \qquad \mathbf{s}_\ell = \frac{\partial c_\ell}{\partial \mathbf{b}}[\bar{\mathbf{b}}_\ell], \qquad s_\ell = c_\ell[\bar{\mathbf{b}}_\ell], \tag{20}$$

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 time steps $\ell > t \geq 0$ are computed by backward recursion.

We proceed by combining Eqs. (7), (17), and (19):

$$
\begin{aligned}
v_t[\mathbf{b}] &= \min_{\mathbf{u}} \Big(c_t[\mathbf{b},\mathbf{u}] + \mathrm{E}\left[v_{t+1}[\mathbf{g}[\mathbf{b},\mathbf{u}] + W[\mathbf{b},\mathbf{u}]\mathbf{w}_t] \right] \Big) \\
&= \min_{\mathbf{u}} \Big(c_t[\mathbf{b},\mathbf{u}] + \mathrm{E}\left[\frac{1}{2}(\mathbf{g}[\mathbf{b},\mathbf{u}] + W[\mathbf{b},\mathbf{u}]\mathbf{w}_t - \bar{\mathbf{b}}_{t+1})^T S_{t+1}(\mathbf{g}[\mathbf{b},\mathbf{u}] + W[\mathbf{b},\mathbf{u}]\mathbf{w}_t - \bar{\mathbf{b}}_{t+1}) + \right. \\
&\qquad \left. (\mathbf{g}[\mathbf{b},\mathbf{u}] + W[\mathbf{b},\mathbf{u}]\mathbf{w}_t - \bar{\mathbf{b}}_{t+1})^T \mathbf{s}_{t+1} + s_{t+1} \right] \Big) \\
&= \min_{\mathbf{u}} \Big(c_t[\mathbf{b},\mathbf{u}] + \frac{1}{2}(\mathbf{g}[\mathbf{b},\mathbf{u}] - \bar{\mathbf{b}}_{t+1})^T S_{t+1}(\mathbf{g}[\mathbf{b},\mathbf{u}] - \bar{\mathbf{b}}_{t+1}) + (\mathbf{g}[\mathbf{b},\mathbf{u}] - \bar{\mathbf{b}}_{t+1})^T \mathbf{s}_{t+1} + \\
&\qquad s_{t+1} + \frac{1}{2} \operatorname{tr}\left[W[\mathbf{b},\mathbf{u}]^T S_{t+1} W[\mathbf{b},\mathbf{u}] \right] \Big) \qquad (21) \\
&= \min_{\mathbf{u}} \Big(c_t[\mathbf{b},\mathbf{u}] + \frac{1}{2}(\mathbf{g}[\mathbf{b},\mathbf{u}] - \bar{\mathbf{b}}_{t+1})^T S_{t+1}(\mathbf{g}[\mathbf{b},\mathbf{u}] - \bar{\mathbf{b}}_{t+1}) + (\mathbf{g}[\mathbf{b},\mathbf{u}] - \bar{\mathbf{b}}_{t+1})^T \mathbf{s}_{t+1} + \\
&\qquad s_{t+1} + \frac{1}{2} \sum_{i=1}^{n} W_{(i)}[\mathbf{b},\mathbf{u}]^T S_{t+1} W_{(i)}[\mathbf{b},\mathbf{u}] \Big), \qquad (22)
\end{aligned}
$$

where $W_{(i)}[\mathbf{b},\mathbf{u}]$ refers to the i'th column of matrix $W[\mathbf{b},\mathbf{u}]$ (note that $W[\mathbf{b},\mathbf{u}]$ has n columns, where n is the dimension of the state). The trace-term in Eq. (21) 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 $\mathbf{x}$, and that $\operatorname{tr}[QXX^T] = \operatorname{tr}[X^T Q X]$. It is this term that ensures that the stochastic nature of the belief dynamics is accounted for in the value iteration. Eq. (22) follows from the fact that $\operatorname{tr}[X^T Q X] = \sum_i {X_{(i)}}^T Q X_{(i)}$.

To approximate the optimal value of $\mathbf{u}$ as a function of $\mathbf{b}$ we linearize the belief dynamics and each of the columns of $W[\mathbf{b},\mathbf{u}]$ about the belief $\bar{\mathbf{b}}_t$ and control input $\bar{\mathbf{u}}_t$ of the nominal trajectory. Given that $\bar{\mathbf{b}}_{t+1} = \mathbf{g}[\bar{\mathbf{b}}_t, \bar{\mathbf{u}}_t]$, we get:

$$
\begin{aligned}
\mathbf{g}[\mathbf{b},\mathbf{u}] - \bar{\mathbf{b}}_{t+1} &\approx F_t(\mathbf{b} - \bar{\mathbf{b}}_t) + G_t(\mathbf{u} - \bar{\mathbf{u}}_t), \qquad (23) \\
W_{(i)}[\mathbf{b},\mathbf{u}] &\approx \mathbf{e}_t^i + F_t^i(\mathbf{b} - \bar{\mathbf{b}}_t) + G_t^i(\mathbf{u} - \bar{\mathbf{u}}_t), \qquad (24)
\end{aligned}
$$

where

$$
\begin{aligned}
F_t &= \frac{\partial \mathbf{g}}{\partial \mathbf{b}}[\bar{\mathbf{b}}_t, \bar{\mathbf{u}}_t], & G_t &= \frac{\partial \mathbf{g}}{\partial \mathbf{u}}[\bar{\mathbf{b}}_t, \bar{\mathbf{u}}_t], & & (25) \\
\mathbf{e}_t^i &= W_{(i)}[\bar{\mathbf{b}}_t, \bar{\mathbf{u}}_t], & F_t^i &= \frac{\partial W_{(i)}}{\partial \mathbf{b}}[\bar{\mathbf{b}}_t, \bar{\mathbf{u}}_t], & G_t^i &= \frac{\partial W_{(i)}}{\partial \mathbf{u}}[\bar{\mathbf{b}}_t, \bar{\mathbf{u}}_t]. \qquad (26)
\end{aligned}
$$

The immediate cost function $c_t[\mathbf{b},\mathbf{u}]$ is quadratized about $\bar{\mathbf{b}}_t$ and $\bar{\mathbf{u}}_t$:

$$
c_t[\mathbf{b},\mathbf{u}] \approx \frac{1}{2} \begin{bmatrix} \mathbf{b} - \bar{\mathbf{b}}_t \\ \mathbf{u} - \bar{\mathbf{u}}_t \end{bmatrix}^T \begin{bmatrix} Q_t & P_t^T \\ P_t & R_t \end{bmatrix} \begin{bmatrix} \mathbf{b} - \bar{\mathbf{b}}_t \\ \mathbf{u} - \bar{\mathbf{u}}_t \end{bmatrix} + \begin{bmatrix} \mathbf{b} - \bar{\mathbf{b}}_t \\ \mathbf{u} - \bar{\mathbf{u}}_t \end{bmatrix}^T \begin{bmatrix} \mathbf{q}_t \\ \mathbf{r}_t \end{bmatrix} + p_t, \qquad (27)
$$

where

$$
\begin{aligned}
Q_t &= \frac{\partial^2 c_t}{\partial \mathbf{b} \partial \mathbf{b}}[\bar{\mathbf{b}}_t, \bar{\mathbf{u}}_t], & R_t &= \frac{\partial^2 c_t}{\partial \mathbf{u} \partial \mathbf{u}}[\bar{\mathbf{b}}_t, \bar{\mathbf{u}}_t], & P_t &= \frac{\partial^2 c_t}{\partial \mathbf{u} \partial \mathbf{b}}[\bar{\mathbf{b}}_t, \bar{\mathbf{u}}_t], \\
\mathbf{q}_t^T &= \frac{\partial c_t}{\partial \mathbf{b}}[\bar{\mathbf{b}}_t, \bar{\mathbf{u}}_t], & \mathbf{r}_t^T &= \frac{\partial c_t}{\partial \mathbf{u}}[\bar{\mathbf{b}}_t, \bar{\mathbf{u}}_t], & p_t &= c_t[\bar{\mathbf{b}}, \bar{\mathbf{u}}]. \qquad (28)
\end{aligned}
$$

Filling in Eqs. (23), (24), and (27) into Eq. (22), we get:

$$
\begin{aligned}
v_t[\mathbf{b}] \approx \min_{\mathbf{u}} \Big(& \frac{1}{2} \begin{bmatrix} \mathbf{b} - \bar{\mathbf{b}}_t \\ \mathbf{u} - \bar{\mathbf{u}}_t \end{bmatrix}^T \begin{bmatrix} Q_t & P_t^T \\ P_t & R_t \end{bmatrix} \begin{bmatrix} \mathbf{b} - \bar{\mathbf{b}}_t \\ \mathbf{u} - \bar{\mathbf{u}}_t \end{bmatrix} + \begin{bmatrix} \mathbf{b} - \bar{\mathbf{b}}_t \\ \mathbf{u} - \bar{\mathbf{u}}_t \end{bmatrix}^T \begin{bmatrix} \mathbf{q}_t \\ \mathbf{r}_t \end{bmatrix} + p_t + \\
& \frac{1}{2}(F_t(\mathbf{b} - \bar{\mathbf{b}}_t) + G_t(\mathbf{u} - \bar{\mathbf{u}}_t))^T S_{t+1}(F_t(\mathbf{b} - \bar{\mathbf{b}}_t) + G_t(\mathbf{u} - \bar{\mathbf{u}}_t)) + \\
& (F_t(\mathbf{b} - \bar{\mathbf{b}}_t) + G_t(\mathbf{u} - \bar{\mathbf{u}}_t))^T \mathbf{s}_{t+1} + s_{t+1} + \\
& \frac{1}{2} \sum_{i=1}^{n} (\mathbf{e}_t^i + F_t^i(\mathbf{b} - \bar{\mathbf{b}}_t) + G_t^i(\mathbf{u} - \bar{\mathbf{u}}_t))^T S_{t+1}(\mathbf{e}_t^i + F_t^i(\mathbf{b} - \bar{\mathbf{b}}_t) + G_t^i(\mathbf{u} - \bar{\mathbf{u}}_t)) \Big) \\
= \min_{\mathbf{u}} \big(& \frac{1}{2} \begin{bmatrix} \mathbf{b} - \bar{\mathbf{b}}_t \\ \mathbf{u} - \bar{\mathbf{u}}_t \end{bmatrix}^T \begin{bmatrix} C_t & E_t^T \\ E_t & D_t \end{bmatrix} \begin{bmatrix} \mathbf{b} - \bar{\mathbf{b}}_t \\ \mathbf{u} - \bar{\mathbf{u}}_t \end{bmatrix} + \begin{bmatrix} \mathbf{b} - \bar{\mathbf{b}}_t \\ \mathbf{u} - \bar{\mathbf{u}}_t \end{bmatrix}^T \begin{bmatrix} \mathbf{c}_t \\ \mathbf{d}_t \end{bmatrix} + e_t \big), \qquad (29)
\end{aligned}
$$

where

$$
C_t = Q_t + F_t^T S_{t+1} F_t + \sum_{i=1}^{n} {F_t^i}^T S_{t+1} F_t^i, \qquad \mathbf{c}_t = \mathbf{q}_t + F_t^T \mathbf{s}_{t+1} + \sum_{i=1}^{n} {F_t^i}^T S_{t+1} \mathbf{e}_t^i, \qquad (30)
$$

$$
D_t = R_t + G_t^T S_{t+1} G_t + \sum_{i=1}^{n} {G_t^i}^T S_{t+1} G_t^i, \qquad \mathbf{d}_t = \mathbf{r}_t + G_t^T \mathbf{s}_{t+1} + \sum_{i=1}^{n} {G_t^i}^T S_{t+1} \mathbf{e}_t^i, \qquad (31)
$$

$$
E_t = P_t + G_t^T S_{t+1} F_t + \sum_{i=1}^{n} {G_t^i}^T S_{t+1} F_t^i, \qquad e_t = p_t + s_{t+1} + \frac{1}{2} \sum_{i=1}^{n} {\mathbf{e}_t^i}^T S_{t+1} \mathbf{e}_t^i. \qquad (32)
$$

Equation (29) is then solved by expanding the terms, taking the derivative with respect to $\mathbf{u}$ and equating to 0 (for $\mathbf{u}$ to be actually a minimum, D_t must be positive-definite. Given the assumptions of Eq. (11), this is necessarily the case). We then get the solution:

$$
\mathbf{u} - \bar{\mathbf{u}}_t = -D_t^{-1} E_t (\mathbf{b} - \bar{\mathbf{b}}_t) - D_t^{-1} \mathbf{d}_t. \qquad (33)
$$

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

$$
\mathbf{u}_t = \pi_t[\mathbf{b}_t] = L_t(\mathbf{b}_t - \bar{\mathbf{b}}_t) + \mathbf{l}_t + \bar{\mathbf{u}}_t, \qquad L_t = -D_t^{-1} E_t, \qquad \mathbf{l}_t = -D_t^{-1} \mathbf{d}_t. \qquad (34)
$$

Filling Eq. (33) back into Eq. (29) gives the value function $v_t[\mathbf{b}]$ as a function of only $\mathbf{b}$ in the form of Eq. (19). Expanding and collecting terms gives:

$$
S_t = C_t - E_t^T D_t^{-1} E_t, \qquad \mathbf{s}_t = \mathbf{c}_t - E_t^T D_t^{-1} \mathbf{d}_t, \qquad s_t = e_t - \frac{1}{2} \mathbf{d}_t^T D_t^{-1} \mathbf{d}_t. \qquad (35)
$$

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

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 [11]. Given the initial belief $\mathbf{b}_0 = (\hat{\mathbf{x}}_0, \sqrt{\Sigma_0})$, and an (arbitrary) initial nominal trajectory $(\bar{\mathbf{b}}_0^{(0)}, \bar{\mathbf{u}}_0^{(0)}, \ldots, \bar{\mathbf{b}}_{\ell-1}^{(0)}, \bar{\mathbf{u}}_{\ell-1}^{(0)}, \bar{\mathbf{b}}_{\ell}^{(0)})$ (such that $\bar{\mathbf{b}}_0^{(0)} = \mathbf{b}_0$ and $\bar{\mathbf{b}}_{t+1}^{(0)} = \mathbf{g}[\bar{\mathbf{b}}_t^{(0)}, \bar{\mathbf{u}}_t^{(0)}]$ for $t \in 0 \ldots \ell - 1$), which can be obtained using RRT motion planning [13], for instance, we proceed as follows.

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 $L_t^{(i)}$ and vectors $\mathbf{l}_t^{(i)}$ for the i'th iteration. We then compute the nominal trajectory $(\bar{\mathbf{b}}_t^{(i+1)}, \bar{\mathbf{u}}_t^{(i+1)})$ of the $i + 1$'th iteration

(starting with $i = 0$) by forward integrating the control policy in the deterministic (zero-noise) belief dynamics:

$$\bar{\mathbf{b}}_0^{(i+1)} = \mathbf{b}_0, \quad \bar{\mathbf{u}}_t^{(i+1)} = L_t^{(i)}(\bar{\mathbf{b}}_t^{(i+1)} - \bar{\mathbf{b}}_t^{(i)}) + \mathbf{l}_t^{(i)} + \bar{\mathbf{u}}_t^{(i)}, \quad \bar{\mathbf{b}}_{t+1}^{(i+1)} = \mathbf{g}[\bar{\mathbf{b}}_t^{(i+1)}, \bar{\mathbf{u}}_t^{(i+1)}], \tag{36}$$

We then recompute the control policy, and reiterate. This lets the control policy converge to a locally optimal trajectory with a second-order convergence rate [14].

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 *line search*. As with Newton's method for finding roots of a function, second order convergence of the above algorithm 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, the approach may overshoot local-minima, which significantly slows down convergence, or even results in divergence. To address this issue, we subtly change the algorithm following [33]. We limit the increment to the control policy by adding a parameter ε to Eq. (33): $(\mathbf{u} - \bar{\mathbf{u}}_t) = L_t(\mathbf{b} - \bar{\mathbf{b}}_t) + \varepsilon \mathbf{l}_t$. Initially, $\varepsilon = 1$, but each time that the control policy creates a trajectory with higher expected cost than the previous nominal trajectory, the trajectory is rejected, ε is divided in half, and a new trajectory is created. 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 [33].

An issue that remains is how to compute the expected cost of a given nominal trajectory. In deterministic iLQG, one simply evaluate its cost using the given immediate cost functions $c_t[\mathbf{b}, \mathbf{u}]$. In our case however, the dynamics are stochastic, so one has to compute the expected cost. We do this as follows. Let $L_t^{(i)}$ and $\varepsilon \mathbf{l}_t^{(i)}$ define the control policy in the i'th iteration. A candidate nominal trajectory for iteration $i+1$ is now generated by applying this control policy with respect to the nominal trajectory of iteration i, according to Eq. (36). We have:

$$\bar{\mathbf{u}}_t^{(i+1)} - \bar{\mathbf{u}}_t^{(i)} = L_t^{(i)}(\bar{\mathbf{b}}_t^{(i+1)} - \bar{\mathbf{b}}_t^{(i)}) + \varepsilon \mathbf{l}_t^{(i)}. \tag{37}$$

The control policy of iteration i itself is defined as

$$\begin{aligned}
\mathbf{u} - \bar{\mathbf{u}}_t^{(i)} &= L_t^{(i)}(\mathbf{b} - \bar{\mathbf{b}}_t^{(i)}) + \varepsilon \mathbf{l}_t^{(i)}, \\
\Rightarrow (\mathbf{u} - \bar{\mathbf{u}}_t^{(i+1)}) + (\bar{\mathbf{u}}_t^{(i+1)} - \bar{\mathbf{u}}_t^{(i)}) &= L_t^{(i)}((\mathbf{b} - \bar{\mathbf{b}}_t^{(i+1)}) + (\bar{\mathbf{b}}_t^{(i+1)} - \bar{\mathbf{b}}_t^{(i)})) + \varepsilon \mathbf{l}_t^{(i)}, \\
\Rightarrow (\mathbf{u} - \bar{\mathbf{u}}_t^{(i+1)}) + L_t^{(i)}(\bar{\mathbf{b}}_t^{(i+1)} - \bar{\mathbf{b}}_t^{(i)}) + \varepsilon \mathbf{l}_t^{(i)} &= L_t^{(i)}((\mathbf{b} - \bar{\mathbf{b}}_t^{(i+1)}) + (\bar{\mathbf{b}}_t^{(i+1)} - \bar{\mathbf{b}}_t^{(i)})) + \varepsilon \mathbf{l}_t^{(i)}, \\
\Rightarrow \mathbf{u} - \bar{\mathbf{u}}_t^{(i+1)} &= L_t^{(i)}(\mathbf{b} - \bar{\mathbf{b}}_t^{(i+1)}).
\end{aligned} \tag{38}$$

Hence, Eq. (38) gives the control policy of iteration i relative to a candidate trajectory of iteration $i+1$.

We now compute the expected cost of the candidate nominal trajectory $(\bar{\mathbf{b}}_t^{(i+1)}, \bar{\mathbf{u}}_t^{(i+1)})$ as follows. Quadratizing the immediate cost functions and linearizing the belief dynamics about the candidate trajectory of iteration $i+1$ according to Eqs. (23) to (28), in combination with the control policy of Eq. (38), allows us to to recursively update the value function along the

candidate trajectory as:

$$S_t = Q_t + L_t^T R_t L_t + L_t^T P_t + P_t^T L_t + (F_t + G_t L_t)^T S_{t+1}(F_t + G_t L_t) + \sum_{j=1}^{n}(F_t^j + G_t^j L_t)^T S_{t+1}(F_t^j + G_t^j L_t), \tag{39}$$

$$\mathbf{s}_t = \mathbf{q}_t + L_t^T \mathbf{r}_t + (F_t + G_t L_t)^T \mathbf{s}_{t+1} + \sum_{j=1}^{n}(F_t^j + G_t^j L_t)^T S_{t+1}\mathbf{e}_t^j, \tag{40}$$

$$s_t = p_t + s_{t+1} + \frac{1}{2}\sum_{j=1}^{n} {\mathbf{e}_t^j}^T S_{t+1}\mathbf{e}_t^j, \tag{41}$$

where $L_t = L_t^{(i)}$. The value s_0 now gives the expected cost of the candidate nominal trajectory with respect to the control policy of the current nominal trajectory (note that the $\mathbf{s}_t$'s are inconsequential for the expected cost, and need not be computed). If this expected cost is lower than the expected cost of the current nominal trajectory, the candidate nominal trajectory is accepted, ε is reset to 1, and the iteration continues. Otherwise, ε is divided in half, and the search for a new nominal trajectory continues. Since the vectors $\mathbf{l}_t$ point in the direction of the gradient of the expected cost, a positive ε that generates a new trajectory with lower expected cost will always be found.

When the magnitude of the $\mathbf{l}_t$'s vanish (or drop below a preset small value), the iteration stops and the current nominal trajectory and its control policy is a locally optimal solution.

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 (square root of the) covariance matrix of the state, the dimension of a belief is $O[n^2]$.

The bottleneck of the running time lies in the computation of the matrix C_t in Eq. (30). Evaluating the product $F_t^T S_{t+1} F_t$ in Eq. (30) of matrices of $O[n^2] \times O[n^2]$ dimension takes $O[n^6]$ time. Also, computing the matrix Q_t of Eq. (28), which contains $O[n^4]$ entries, using numerical differentiation (central differences) can be done in $O[n^6]$ time assuming that $c_t[\mathbf{b}, \mathbf{u}]$ can be evaluated in $O[n^2]$ time. Further, the product ${F_t^i}^T S_{t+1} F_t^i$ is evaluated n times, but each can be evaluated in $O[n^5]$ time, since each F_t^i only contains non-zero entries in the upper $n \times O[n^2]$ block of the matrix (see the definition of $W[\mathbf{b}, \mathbf{u}]$ in Eq. (18)). Note that linearizing the belief dynamics, i.e. computing the matrices F_t, G_t, F_t^i and G_t^i using numerical differentiation (central differences) can be done in $O[n^5]$ time, as it involves evaluating the belief dynamics (which takes $O[n^3]$ time for the EKF (and also for the UKF)) $O[n^2]$ times. Hence, this does not form a bottleneck of the computation.

A complete cycle of value iteration takes ℓ steps (ℓ being the index of the final time step), bringing the complexity to $O[\ell n^6]$. 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 $c_t[\mathbf{b}, \mathbf{u}]$ (with the assumptions of Eq. (11)). In typical LQG-style cost functions, the existence of obstacles in the environment is not incorporated, while we may want to minimize the probability of colliding with them. We incorporate obstacles into the cost functions as follows.

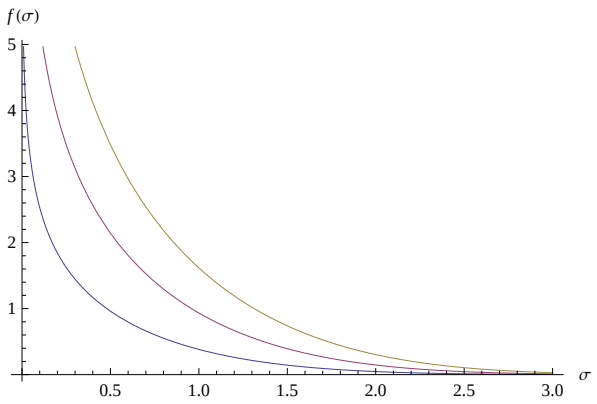


Figure 1: *Plots of the function* $f[\sigma] = -\log\gamma[n/2, \sigma^2/2]$ *for* $n = \{1, 2, 3\}$.

Let $\mathcal{O} \subset \mathcal{X}$ be the region of the state space that is occupied by obstacles. Given a belief $\mathbf{b}_t = (\hat{\mathbf{x}}_t, \sqrt{\Sigma_t})$, 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 [26], 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 (it takes one geometric distance computation to compute this number, and does not involve Monte Carlo sampling). 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 $f[\sigma[\mathbf{b}]] > 0$ and $\frac{\partial^2 f}{\partial\sigma\partial\sigma} > 0$; see Fig. 1), 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 $c_t[\mathbf{b}, \mathbf{u}]$ at $\bar{\mathbf{b}}_t$ (i.e. computing the matrix Q_t as is done in Eq. (28)), 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 $c_t[\mathbf{b}, \mathbf{u}]$. 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 about $\sigma[\bar{\mathbf{b}}_t]$:

$$f[\sigma[\mathbf{b}]] \approx \frac{1}{2}a(\sigma[\mathbf{b}] - \sigma[\bar{\mathbf{b}}_t])^2 + b(\sigma[\mathbf{b}] - \sigma[\bar{\mathbf{b}}_t]) + f[\sigma[\bar{\mathbf{b}}_t]], \tag{42}$$

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 about $\bar{\mathbf{b}}_t$:

$$\sigma[\mathbf{b}] - \sigma[\bar{\mathbf{b}}_t] \approx (\mathbf{b} - \bar{\mathbf{b}}_t)^T\mathbf{a} \tag{43}$$

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. (43) in Eq. (42), we get

$$f[\sigma[\mathbf{b}]] \approx \frac{1}{2}(\mathbf{b} - \bar{\mathbf{b}}_t)^T(a\mathbf{a}\mathbf{a}^T)(\mathbf{b} - \bar{\mathbf{b}}_t) + (\mathbf{b} - \bar{\mathbf{b}}_t)^T(b\mathbf{a}) + f[\sigma[\bar{\mathbf{b}}_t]]. \tag{44}$$

Hence, $a\mathbf{a}\mathbf{a}^T$ is an approximate Hessian of the obstacle term $f[\sigma[\mathbf{b}]]$ of $c_t[\mathbf{b}, \mathbf{u}]$ that requires only $O[n^2]$ collision-checks to compute. In addition, since $a > 0$, this Hessian is guaranteed to be positive-semidefinite, as mandated by Eq. (11).

6 Results

We evaluate our approach in simulation applied to robot motion planning scenarios involving stochastic dynamics, measurement models with state and control-dependent noise, and spatially-varying sensing capabilities. We consider three scenarios: (i) a 2-D point robot with linear dynamics, (ii) a non-holonomic, car-like robot with second-order dynamics, and (iii) an aircraft-like robot navigating in a 3-D environment.

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 accumulate 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 uncertainty and measurement noise.

In each of the following experiments, 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)):

$$c_\ell[\mathbf{b}_\ell] = \hat{\mathbf{x}}_\ell^T Q_\ell \hat{\mathbf{x}}_\ell + \mathrm{tr}[\sqrt{\Sigma_\ell} Q_\ell \sqrt{\Sigma_\ell}], \tag{45}$$

$$c_t[\mathbf{b}_t, \mathbf{u}_t] = \mathbf{u}_t^T R_t \mathbf{u}_t + \mathrm{tr}[\sqrt{\Sigma_\ell} Q_t \sqrt{\Sigma_t}] + f[\sigma[\mathbf{b}_t]], \tag{46}$$

for given $Q_t \geq 0$ and $R_t > 0$. The term $\hat{\mathbf{x}}_\ell^T Q_\ell \hat{\mathbf{x}}_\ell + \mathrm{tr}[\sqrt{\Sigma_\ell} Q_\ell \sqrt{\Sigma_\ell}] = \mathrm{E}[\mathbf{x}_\ell^T Q_\ell \mathbf{x}_\ell]$ 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, $\mathrm{tr}[\sqrt{\Sigma_t} Q_t \sqrt{\Sigma_t}]$ penalizes the uncertainty, and $f[\sigma[\mathbf{b}_t]]$ encodes the obstacle cost term (if applicable). Using the approximation of Eq. (44) for $f[\sigma[\mathbf{b}_t]]$, the above cost functions are in accordance with the assumptions of Eq. (11), and their Hessians can be constructed in $O[n^4]$ time, so it does not present a bottleneck for the running time.

All the performance results presented in this section are based on a C++ implementation running on a 3.33 Ghz Intel® i7™ PC. For each scenario, we evaluate the performance of our approach and the quality of the computed control policy. We also separately consider environments with and without obstacles to demonstrate that our approach can handle both types of environments. We compare and analyze the performance and convergence characteristics of the approach presented in this paper to our preliminary approach based on stochastic differential dynamic programming (sDDP) [28]. We also analyze the effect of assuming maximum-likelihood observations [21, 7, 8] on the computed locally optimal trajectory and corresponding control policy.

6.1 2-D Point Robot

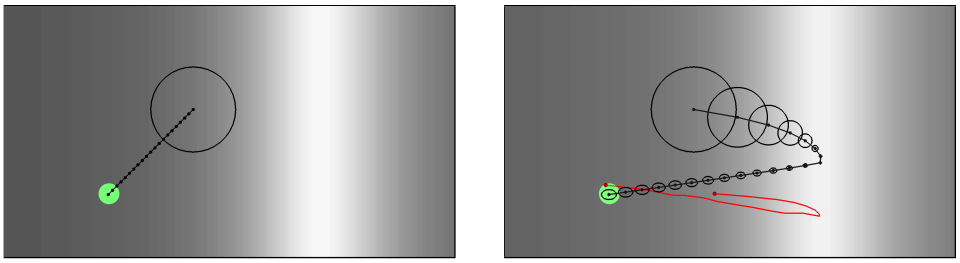
We consider the case of a point robot moving in a 2-D environment with the following linear dynamics model with control-dependent motion noise:

$$\mathbf{x}_{t+1} = \mathbf{f}[\mathbf{x}_t, \mathbf{u}_t, \mathbf{m}_t] = \mathbf{x}_t + \tau \mathbf{u}_t + M[\mathbf{u}_t] \cdot \mathbf{m}_t, \tag{47}$$

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, τ is the duration of a time step, and the matrix $M[\mathbf{u}_t]$ scales the motion noise $\mathbf{m}_t$ proportional to the control input $\mathbf{u}_t$.

The robot localizes itself using noisy measurements from sensors in the environment, the reliability of which varies as a function of the robot's position $\mathbf{x}$. 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. This gives the following linear observation model with spatially-varying noise:

$$\mathbf{z}_t = \mathbf{h}[\mathbf{x}_t, \mathbf{n}_t] = \mathbf{x}_t + N[\mathbf{x}_t] \cdot \mathbf{n}_t, \tag{48}$$



(a) Initial trajectory. (b) Locally optimal solution.

Figure 2: *Point robot moving in a 2-D light-dark domain without obstacles (adapted from Platt et al. [21]). (a) The method is initialized with a naïve straight line trajectory to the goal. (b) The nominal trajectory and associated beliefs of the solution (shown in black), and the trajectory obtained by applying the computed feedback policy to a robot with an initial belief that is considerably different than the initial belief used for computing the control policy (shown in red).*

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 function of the robot's position.

We use state and control cost matrices of $Q_t = I$, $R_t = I$, and the final cost matrix, $Q_\ell = 10\ell I$ in our experiments, where ℓ is the number of sections in the initial trajectory. The method converges when the difference between the expected costs between successive iterations falls below a user-specified epsilon threshold.

6.1.1 Light-Dark Domain (No Obstacles)

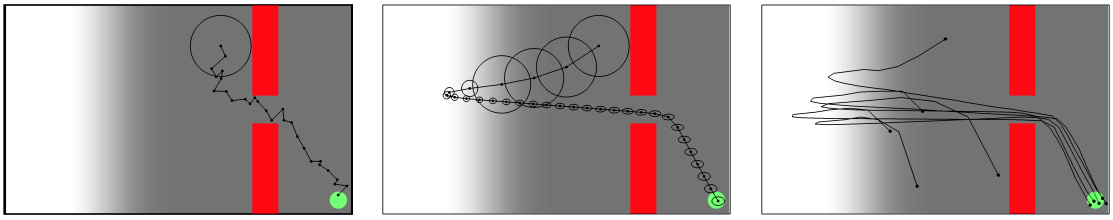
We consider the light-dark domain scenario suggested by Platt et al. [21]. The measurement noise (modeled by the matrix $N[\mathbf{x}_t]$) varies as a quadratic function of the robot's horizontal coordinate x (as shown in Fig. 2). We initialize our method with a naïve straight line trajectory from the initial position to the goal (Fig. 2(a)).

Fig. 2(b) shows the nominal trajectory and the associated beliefs of the solution computed using our approach. The locally optimal nominal trajectory leads the robot to the horizontal coordinate where the measurement noise is minimum, in order to better localize itself, before proceeding to the goal. For this example, the initial nominal trajectory has an expected cost of 49.7, and the trajectory converges to a (local) optimum with an expected cost of 9.61 in 42 iterations, requiring a total computation time of 0.094 seconds. To evaluate the quality of the computed control policy, we also computed the actual expected costs across 10000 simulation runs that use the computed feedback policy to compensate for artificial motion and measurement noise. The actual expected cost for the computed control policy was 9.46 units.

To demonstrate the effectiveness of the control policy computed by our method, we apply the computed feedback policy to a robot with a belief that is considerably different than the belief with which our method is initialized. The resulting trajectory is indicated in red in Fig. 2(b). The computed policy initially leads the robot towards the light region, it quickly rectifies the trajectory after a better estimate of the belief is obtained in the light region. The basin of attraction of the control policy is wide enough to avoid the need for replanning.

6.1.2 Light-Dark Domain (With Obstacles)

We consider the light-dark domain scenario with obstacles as suggested in Bry and Roy [5]. In this scenario, the measurement noise (modeled by the matrix $N[\mathbf{x}_t]$) varies as a sigmoid function



(a) Initial trajectory. (b) Locally optimal solution. (c) Execution traces.

Figure 3: *Point robot moving in a 2-D light-dark domain 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 approach. The robot moves away from the goal to better localize itself before reaching the goal with significantly reduced uncertainty. (c) Execution traces of the robot's true state drawn from different initial beliefs while following the computed control policy.*

of the robot's horizontal coordinate x (as shown in Fig. 3). We initialize our method with a collision-free initial trajectory computed using an RRT planner [13] (Fig. 3(a)).

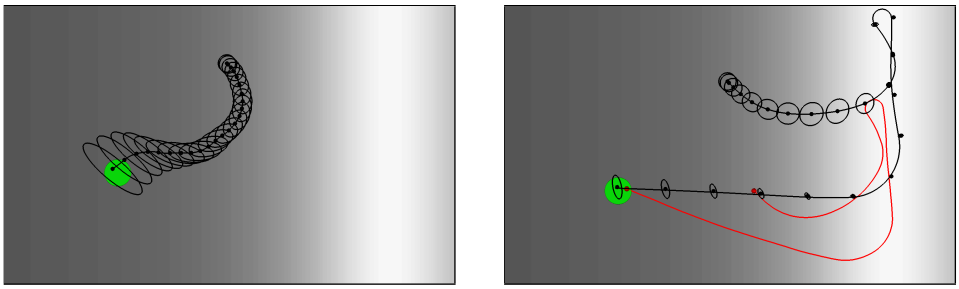
Fig. 3(b) shows the nominal trajectory and the associated beliefs of the solution computed using our approach. The nominal trajectory leads the robot to the region of the environment with reliable sensing for better localization, before moving the robot through the narrow passage to arrive at the goal. For this example, the initial trajectory has an expected cost of 144.9 and the trajectory converges to a local optimum with expected cost of 14.08 in 66 iterations, which requires a total computation time of 3.657 seconds. To evaluate the quality of the computed control policy, we also computed the actual expected costs across 10000 simulation runs that use the computed feedback policy to compensate for artificial motion and measurement noise. The actual expected cost was 13.8 units.

To demonstrate the effectiveness of the control policy computed by our method, we apply the computed feedback policy to a robot with a belief that is considerably different than the belief with which our method is initialized. Fig. 3(c) shows traces of the true state of the robot $\mathbf{x}$ across 5 simulations where the initial state of the robot $\mathbf{x}_0$ is sampled from a different initial belief to evaluate the robustness of the control policy. Even if the initial belief is considerably different from the initial belief used to compute the solution, 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.

Our solution also agrees with the solution found by Bry and Roy [5] for this experiment. Our method directly optimizes the trajectory rather than relying on an optimal sampling-based planner in belief space, resulting in an order of magnitude faster computation times. Our method also does not assume fixed control gains along each along each section of the nominal trajectory. However, the method of Bry and Roy is able to find a globally-optimal solution (given the fixed control gains), whereas our method computes a locally optimal solution given an initial trajectory.

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 noisy and partial sensing of the robot's state. 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 $\mathbf{u} = (a, \phi)$



(a) Initial trajectory. (b) Locally optimal solution.

Figure 4: *Car-like robot moving in a 2-D light-dark domain without obstacles (adapted from Platt et al. [21]). (a) The method is initialized with a naïve trajectory to the goal using a RRT planner. (b) The nominal trajectory and associated beliefs of the solution computed using our approach (shown in black), and the trajectory obtained by applying the computed feedback policy to a robot with a belief that is considerably different than the belief used for method initialization (red).*

consists of an acceleration a and the steering wheel angle ϕ. This gives the following non-linear dynamics model:

$$\mathbf{x}_{t+1} = \mathbf{f}[\mathbf{x}_t, \mathbf{u}_t, \mathbf{m}_t] = \begin{bmatrix} x_t + \tau v_t \cos\theta_t \\ y_t + \tau v_t \sin\theta_t \\ \theta_t + \tau v_t \tan[\phi]/d \\ v_t + \tau a \end{bmatrix} + M[\mathbf{u}_t] \cdot \mathbf{m}_t, \tag{49}$$

where τ is the duration of a time step, d is the length of the car-like robot, and $M[\mathbf{u}_t]$ scales the motion noise $\mathbf{m}_t$ proportional to the control input $\mathbf{u}_t$.

6.2.1 Light-Dark Domain (No Obstacles)

We again consider the light-dark domain scenario suggested by Platt et al. [21]. In this scenario, the robot's ability to sense its state is both partial (the robot is only capable of sensing its position but not its velocity or orientation) and noisy. The measurement noise (modeled by the matrix $N[\mathbf{x}_t]$) varies as a quadratic function of the robot's horizontal coordinate x (as shown in Fig. 4). This gives the following observation model with spatially-varying noise:

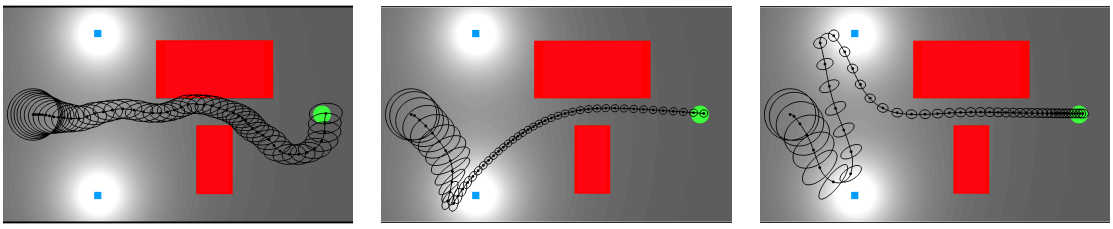
$$\mathbf{z}_t = \mathbf{h}[\mathbf{x}_t, \mathbf{n}_t] = \left[\begin{smallmatrix} x_t \\ y_t \end{smallmatrix}\right] + N[x_t] \cdot \mathbf{n}_t, \tag{50}$$

where the measurement vector $\mathbf{z}_t \in \mathbb{R}^2$ consists of noisy measurements of the robot's position, and the matrix $N[x_t]$ scales the measurement noise based on a function of the robot's horizontal coordinate x.

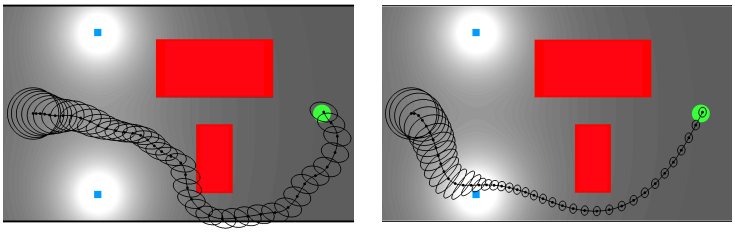
We initialize our method with a naïve trajectory to the goal computed using a RRT planner [13] (Fig. 4(a)). We use state and control cost matrices of $Q_t = I$, $R_t = I$, and the final cost matrix, $Q_\ell = 10\ell I$ in our experiments, where ℓ is the number of sections in the initial trajectory.

Fig. 4(b) shows the nominal trajectory and the associated beliefs of the solution computed using our approach. The locally optimal nominal trajectory leads the robot to the horizontal coordinate where the measurement noise is minimum, in order to better localize itself, before proceeding to the goal. For this example, the initial trajectory has an expected cost of 25.76 and the trajectory converges to a local-optimum with an expected cost of 7.6 in 81 iterations, which requires a total computation time of 2.07 seconds.

We also apply the computed feedback policy to a robot with a belief that is considerably different that the belief with which our method is initialized. The resulting trajectory is shown



(a) Initial trajectory. (b) Locally optimal solution. (c) Solution for $Q_t = 10I\ell$.



(d) Initial trajectory. (e) Locally optimal solution.

Figure 5: *A car-like robot moving in a 2-D light-dark domain 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 and the associated beliefs of solution computed using our approach. The robot localizes itself by moving closer to the beacon(s) before reaching the goal. The final nominal trajectory also follow the medial axis between the narrow passage to minimize the possibility of colliding with obstacles. (c) 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. (d) A different initial trajectory results in a different locally optimal solution. (e) Our method is able to improve trajectories within a single homotopy class.*

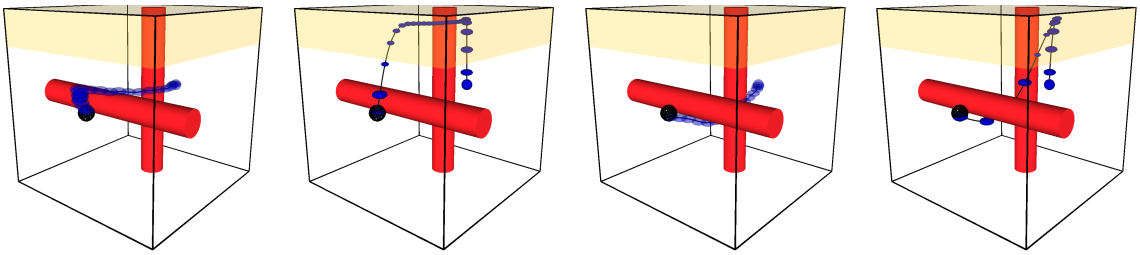
in red in Fig. 4(b). Since the belief is considerably different from the assumed belief used for method initialization, the control policy leads the robot to mimic the computed nominal trajectory, but once the robot has localized itself in the light region of the environment, the control policy reliably leads the robot to the goal.

6.2.2 Domain With Spatially Varying Sensing (With Obstacles)

We also consider a scenario in which the car-like robot estimates its location 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:

$$\mathbf{z}_t = \mathbf{h}[\mathbf{x}_t, \mathbf{n}_t] = \begin{bmatrix} 1/((x_t - \check{x}_1)^2 + (y_t - \check{y}_1)^2 + 1) \\ 1/((x_t - \check{x}_2)^2 + (y_t - \check{y}_2)^2 + 1) \\ v_t \end{bmatrix} + N\mathbf{n}_t, \tag{51}$$

where the 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. 5(a) visually illustrates the quadratic decay in the beacon signal strengths in the environment. The robot is able to obtain reliable measurements in the bright regions of the environment, but the measurements become relatively noisier as the robot moves in to the dark regions due to the decreased signal-to-noise ratio.



(a) Initial trajectory. (b) Locally optimal solution. (c) Different trajectory. (d) Locally optimal solution.

Figure 6: *An aircraft-like robot with omni-directional acceleration moving in a 3-D environment with obstacles with partial and noisy sensing. The motion uncertainty is considerably lower at higher altitudes (indicated by the yellow region). (a) An initial collision-free trajectory is computed using an RRT planner. (b) Nominal trajectory and the associated beliefs of solution computed using our approach. The nominal trajectory is locally optimized such that the robot spends a large proportion of the trajectory at higher altitudes to reduce uncertainty, before reaching the goal. (c) A different trajectory initialization results in local improvement within its initial homotopy class, resulting in a locally optimal nominal trajectory (d).*

We initialize our method with a collision-free trajectory to the goal computed using a RRT planner [13] (Fig. 5(a)). We use state and control cost matrices of $Q_t = I$, $R_t = I$, and the final cost matrix, $Q_\ell = 10\ell I$ in our experiments.

Fig. 5(b) shows the nominal trajectory and the associated beliefs of the solution computed using our approach. The nominal trajectory leads the robot to the region of the environment with reliable sensing for better localization, before moving the robot through the narrow passage to arrive at the goal. In contrast to the initial trajectory (Fig. 5(a)), the locally optimal trajectory also moves away from the obstacles and takes a safer path to the goal. For this example, the initial trajectory has an expected cost of 101.65 and the trajectory converges to a local-optimum with an expected cost of 20.57 in 19 iterations, which requires a total computation time of 9.57 seconds.

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. 5(c) 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. Figs. 5(d) and 5(e) shows the nominal trajectory when a different initial trajectory is provided as input. The presence of obstacles in the environment forces our method to locally optimize trajectories within a single homotopy class.

6.3 3-D Aircraft

We consider the case of an aircraft-like robot with partial and noisy sensing maneuvering in a 3-D environment with obstacles. We consider a simplified model of an aircraft that has omni-directional acceleration. This model can be used to approximate the kinematic constraints on the aircraft as long as the aircraft is moving with non-zero speed [27]. The state $\mathbf{x} = (x, y, z, v_x, v_y, v_z) \in \mathbb{R}^6$ of the robot consists of its position $\mathbf{p} = (x, y, z)$ and its velocity $\mathbf{v} = (v_x, v_y, v_z)$. The control input vector $\mathbf{u} = (a_x, a_y, a_z)$ comprises of the omni-directional acceleration applied to the robot. This gives the following dynamics model:

$$\mathbf{x}_{t+1} = \mathbf{f}[\mathbf{x}_t, \mathbf{u}_t, \mathbf{m}_t] = \begin{bmatrix} \mathbf{p}_t + \tau \mathbf{v}_t + \frac{1}{2}\tau^2 \mathbf{u}_t \\ \mathbf{v}_t + \tau \mathbf{u}_t \end{bmatrix} + M[\mathbf{p}_t] \cdot \mathbf{m}_t, \tag{52}$$

where τ is the duration of a time step, and $M[\mathbf{p}_t]$ scales the motion noise $\mathbf{m}_t$ proportional to the robot's position $\mathbf{p}_t$. We set motion uncertainty to be much lower at higher altitudes, approximately modeling the effect of atmospheric and weather conditions on the robot motion. The uncertainty steadily increases as the altitude of the robot decreases (Fig. 6).

We also assume the following stochastic observation model based on partial and noisy sensing:

$$\mathbf{z}_t = \mathbf{h}[\mathbf{x}_t, \mathbf{n}_t] = \mathbf{p}_t + N \cdot \mathbf{n}_t, \tag{53}$$

where the measurement vector $\mathbf{z}_t \in \mathbb{R}^2$ consists of noisy measurements of the robot's position, and the measurement noise is scaled by a constant matrix N.

We initialize our method with a collision-free trajectory to the goal computed using a RRT planner [13] (Fig. 6(a)). Fig. 6(b) shows the nominal trajectory and the associated beliefs of the solutions computed using our approach. The robot spends a considerable proportion of the nominal trajectory at higher altitudes in order to reduce the uncertainty, before arriving at the goal. In contrast to the initial trajectory (Fig. 6(a)), the locally optimal trajectory is also smoother in terms of the applied control inputs and stays away from the obstacles to take a safer path to the goal. For this example, the initial trajectory has an expected cost of 4539.3 and the trajectory converges to a local optimum with a considerably lower expected cost of 705.96 in 47 iterations, which requires a total computation time of 41.8 seconds.

Figs. 6(c) and 6(d) show the nominal trajectory when a different initial trajectory is provided as input. The presence of obstacles in the environment forces our method to locally optimize trajectories within a single homotopy class. Our method is still able to locally force the robot to ascend to a higher altitude to reduce the uncertainty, before descending below and going around the obstacle to arrive at the goal.

6.4 Comparison between iLQG and sDDP

We quantitatively compared our approach with value iteration based on iLQG with our preliminary approach with value iteration based on stochastic differential dynamic programming (sDDP) [28]. In Table 1, we compare the number of iterations required for convergence and the optimal expected cost for each of the considered scenarios for both methods. Qualitatively, the iLQG-based method is asymptotically faster than the sDDP-based method ($O[n^6]$ rather than $O[n^7]$) and numerically more stable even when the sDDP method is implemented with the square root of the variance in the belief (sDDP requires regularization of matrices to maintain positive-semidefiniteness of the value function).

As expected, each iteration of the iLQG method ($O[n^6]$) takes less time than an equivalent sDDP iteration ($O[n^7]$). The differences are more pronounced as the dimensionality of the belief space increases, as is evident in the aircraft scenario. On the other hand, sDDP converges in fewer iterations then iLQG. This is because sDDP uses direct computation of the Hessians of the value function, while iLQG computes the Hessians based on a linearization of the belief dynamics (which truncates some second-order terms compared to sDDP).

In all experiments, iLQG and sDDP yield almost identical solutions, whose difference is visually hardly appreciable, and the optimal expected cost that both iLQG and sDDP converge to are almost identical. To evaluate the difference in the two methods, we also compute the actual expected costs across 10000 simulation runs that use the computed feedback policy to compensate for artificially simulated motion uncertainty and measurement noise. The differences in the actual expected costs are minimal, which alludes to the fact that the control policies computed by the two methods are similar. This is what one would expect; the slight differences that do appear are a result of numerical variations between the methods, and in a few cases this causes the approaches to converge to different local optima.

Overall, our experiments indicate that iLQG is preferable over sDDP because it scales better to higher dimensional problems and is numerically more stable since the iLQG method

Scenario	Initial exp cost	Method	Num. iter	Time (s)	Time per iter (s)	Optimal exp cost	Actual exp cost
Point (no obs)	49.69	iLQG	42	0.09	0.002	9.61	9.46
		sDDP	13	0.125	0.009	9.72	9.52
Point (obs)	144.9	iLQG	66	3.66	0.055	14.08	13.8
		sDDP	51	3.14	0.062	14.08	13.98
Car (no obs)	25.76	iLQG	81	2.07	0.025	7.67	7.39
		sDDP	55	10.79	0.196	7.27	7.03
Car (obs)	101.65	iLQG	19	9.57	0.5	20.57	20.3
		sDDP	16	12.1	0.76	20.71	20.38
aircraft	4539.3	iLQG	47	41.8	0.89	705.96	703.45
		sDDP	35	136.59	3.9	705.84	703.72

Table 1: *Comparison of iLQG and sDDP.*

does not require regularization to ensure that the Hessians are positive semi-definite [28]. The inherent complexity of the method is still too high for robots with complex dynamics and high-dimensional state spaces, and algorithmic improvements in the method and efficient implementations thereof present interesting research directions.

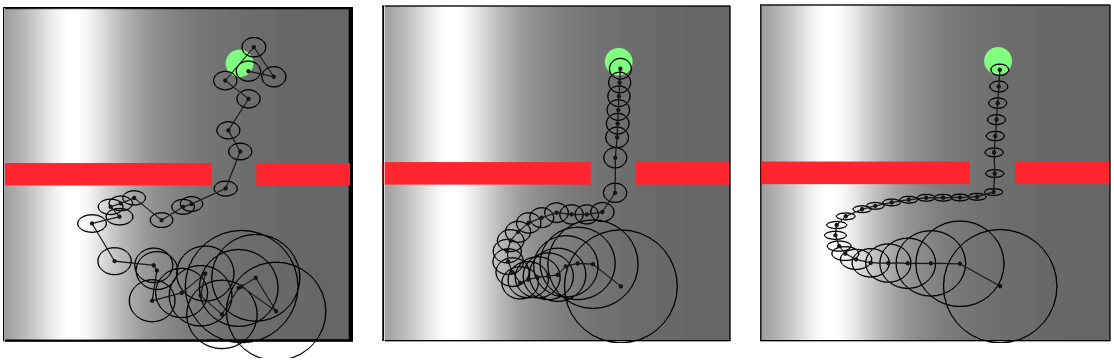
6.5 Effect Of Assuming Maximum-Likelihood Observations

We analyze the effect of assuming maximum-likelihood observations made in prior work [21, 7, 8] on the computed locally optimal trajectory and corresponding control policy. We reproduce this assumption in our method by ignoring all the terms in the value iteration that pertain to the matrix $W[\mathbf{b}, \mathbf{u}]$, which determines the stochastic nature of the belief dynamics given by Eq. (17). More specifically, we can reproduce the assumption by removing the terms containing the sum-quantifiers in Eqs. (30), (31), and (32). This has the net result of considering deterministic belief dynamics as is the case when maximum-likelihood observations are assumed.

We consider an illustrative example that considers a point robot moving in a 2-D domain with obstacles, as shown in Fig. 7(a). We consider the same stochastic dynamics model for the robot as in Sec. 6.1. We also consider the light-dark domain scenario suggested by Platt et al. [21] where the measurement noise varies as a quadratic function of the robot's horizontal coordinate x (as shown in Fig. 7(a)). We use state and control cost matrices of $Q_t = I$, $R_t = 3I$, and the final cost matrix, $Q_\ell = 10I$ in our experiments.

We computed 100 random trajectories using an RRT planner [13] and used the trajectories to initialize our method with and without assuming maximum-likelihood observations. In the case of maximum-likelihood observations, the mean initial cost is 107.2 units with a standard deviation of 35 units. The mean final cost at convergence is 17.7 units with a standard deviation of 1.5 units. It is important to note that the final cost is based on deterministic belief dynamics and is exactly known. We also computed the final expected cost of the computed control policy using value iteration assuming stochastic belief dynamics, as outlined in Sec. 4.2. The mean expected cost of the policy at convergence is 23.1 units with a standard deviation of 2 units. This indicates that there is a mismatch in the final cost assuming deterministic belief dynamics and the actual expected cost of the computed policies when executed under motion and sensing uncertainty.

We also ran our method on the same 100 trajectories without assuming maximum-likelihood observations. The mean initial cost is $35,371$ units with a standard deviation of $41,522$ units, while the mean expected cost at convergence is 21.3 units with a standard deviation of 1.9 units. For this scenario, our method which does *not* assume maximum-likelihood observations yielded an average expected cost 8.5% better than the method making the maximum-likelihood



(a) Initial trajectory.

(b) With maximum-likelihood assumption.

(c) Without maximum-likelihood assumption.

Figure 7: *An illustrative example that considers a point robot moving in a 2-D domain 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 under the assumption of maximum-likelihood observations. The optimization results in a nominal trajectory that does not lead the robot all the way to the horizontal coordinate where the measurement noise is minimum. (c) Solution computed without making the maximum-likelihood observation assumption. The optimization is able to find a different locally optimal trajectory and policy that allows the robot to localize itself with certainty before arriving at the goal region with reduced uncertainty.*

assumption.

To demonstrate the effectiveness of the control policy computed with and without assuming maximum-likelihood observations, we evaluated each control policy quantitatively by computing the percentage of executions in which the robot was able to avoid obstacles across 10000 simulation executions assuming artificial motion and measurement noise. In our experiments, the control policies computed assuming maximum-likelihood observations result in an average of 324 collisions (standard deviation: 87) while the control policies computed by our method result in an average of 252 collisions (standard deviation: 72). This demonstrates that not assuming maximum-likelihood observations reduces the number of collisions by approximately 25% for the considered scenario.

We visualize the difference in the two cases in Figs. 7(b) and 7(c) using an illustrative example from the 100 random scenarios considered in our experiments. As shown in Fig. 7(b), the nominal trajectory for the case in which we assume maximum-likelihood observations does not lead the robot all the way to the horizontal coordinate where the measurement noise is minimum. This results in a higher expected cost of 24.4 units at convergence and higher uncertainty in the state of the robot as the robot traverses the narrow passage. In contrast, the solution computed without making the maximum-likelihood observation assumption is able to find a different locally optimal trajectory and policy that allows the robot to localize itself with greater certainty before arriving at the goal region with reduced uncertainty (see Fig. 7(c)). The expected cost at convergence in this case is 16.9 units. We note that a lower expected cost is not guaranteed: among the 100 random initial trajectories there are also cases in which the solution computed with the maximum likelihood assumption has a better expected cost than the solution computed without the assumption. As in the scenario of the figure, this is very likely the result of both methods converging to a different local optimum.

Overall, our results indicate that *not* making the maximum-likelihood assumption gives, on average, better control policies. However, depending on the application, the impact of the assumption may be relatively limited. This raises the question of whether the assumption can be formally justified and its negative impact bounded. In the case of our method, making the

assumption does not improve the (asymptotic) running time of the algorithm, implying the maximum likelihood assumption should not be used. But in other contexts, e.g. in the case of non-Gaussian beliefs, the assumption may greatly simplify the computations or even enable finding a solution that would otherwise be intractable. Finding a formal justification for the assumption, even if only for the Gaussian case, would greatly benefit research on continuous POMDPs and motion planning under uncertainty.

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 ($O[n^6]$) 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 and unscented Kalman filters 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 $O[n^6]$ is polynomial, it may still be too high for robots with complex dynamics and high-dimensional state spaces or for real-time application. Recent preliminary work by the authors [29] suggests that the running time can be brought down to $O[n^4]$ when approximating the value function by a function that is quadratic in the mean, but linear in the variance. This seems to come at the expense of convergence rate however, and the resulting control policy operates on only the mean and not the entire belief. Further, we are exploring the use of different optimization methods on belief spaces, such as direct collocation and sequential quadratic programming methods [3, 30]. We also want to apply our method to real-world domains involving complex dynamics such as autonomous quadrotor flight, medical needle steering, or manipulation of deformable tissue.

Acknowledgments

This research was supported in part by the National Science Foundation (NSF) under awards #IIS-0905344, #IIS-1117127, and #IIS-1149965 and by the National Institutes of Health (NIH) under award #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] J. Betts. Practical Methods for Optimal Control Using Nonlinear Programming. *Proc. SIAM Advances in Design and Control*, 2001.

[4] 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.

[5] A. Bry, N. Roy. Rapidly-exploring random belief trees for motion planning under uncertainty. *IEEE Int. Conf. on Robotics and Automation*, 2011.

[6] S. Candido, S. Hutchinson. Minimum Uncertainty Robot Navigation Using Information-guided POMDP Planning. *IEEE Int. Conf. on Robotics and Automation*, 2011.

[7] N. Du Toit, J. Burdick. Robotic motion planning in dynamic, cluttered, uncertain environments. *IEEE Int. Conf. on Robotics and Automation*, 2010.

[8] T. Erez, W. D. Smart. A Scalable Method for Solving High-Dimensional Continuous POMDPs Using Local Approximation. *Conf. on Uncertainty in Artificial Intelligence*, 2010.

[9] K. Hauser. Randomized belief-space replanning in partially-observable continuous spaces. *Workshop on the Algorithmic Foundations of Robotics*, 2010.

[10] V. Huynh, N. Roy. icLQG: combining local and global optimization for control in information space. *IEEE Int. Conf. on Robotics and Automation*, 2009.

[11] D. Jacobson, D. Mayne. *Differential Dynamic Programming*. American Elsevier Publishing Company, Inc., New York, 1970.

[12] S. Julier, J. Uhlmann. Unscented filtering and nonlinear estimation. *Proc. IEEE* 92(3):401–422, 2004.

[13] S. LaValle, J. Kuffner. Randomized kinodynamic planning. *Int. Journal on Robotics Research* 20(5):378–400, 2001.

[14] L.-Z. Liao, C. Shoemaker. Convergence in unconstrained discrete-time differential dynamic programming. *IEEE Trans. on Automatic Control* 36(6):692–706, 1991.

[15] 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.

[16] L. Kaelbling, M. Littman, A. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101(1–2):99–134, 1998.

[17] 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.

[18] 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.

[19] C. Papadimitriou, J. Tsisiklis. The complexity of Markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.

[20] J. Porta, N. Vlassis, M. Spaan, P. Poupart. Point-based value iteration for continuous POMDPs. *Journal of Machine Learning Research* 7:2329–2367, 2006.

[21] R. Platt, R. Tedrake, L. Kaelbling, T. Lozano-Perez. Belief space planning assuming maximum likelihood observations. *Robotics: Science and Systems*, 2010.

[22] S. Prentice, N. Roy. The belief roadmap: Efficient planning in belief space by factoring the covariance. *Int. J. of Robotics Research* 28(11–12):1448–1465, 2009.

[23] S. Thrun. Monte Carlo POMDPs. *Advances in Neural Information Processing Systems*. The MIT Press, 2000.

[24] S. Thrun, W. Burgard, D. Fox. *Probabilistic Robotics*, MIT Press, 2005.

[25] E. Todorov, W. Li. A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems. *American Control Conference*, 2005.

[26] 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.

[27] J. van den Berg, J. Snape, S. J. Guy, D. Manocha. Reciprocal collision avoidance with acceleration-velocity obstacles. *IEEE Int. Conf. on Robotics and Automation*, 2011.

[28] J. van den Berg, S. Patil, R. Alterovitz. Motion Planning Under Uncertainty Using Differential Dynamic Programming in Belief Space. *Proc. Int. Symposium of Robotics Research (ISRR)*, 2011.

[29] J. van den Berg, S. Patil, R. Alterovitz. Efficient Approximate Value Iteration for Continuous Gaussian POMDPs. *Proc. AAAI Conf. on Artificial Intelligence*, 2012.

[30] O. von Stryk. Numerical solution of optimal control problems by direct collocation. *Optimal Control* (International Series in Numerical Mathematics 111), pp. 129-143, 1993.

[31] M. P. Vitus, C. J. Tomlin. Closed-Loop Belief Space Planning for Linear, Gaussian Systems. *IEEE Int. Conf. on Robotics and Automation*, 2011.

[32] G. Welch, G. Bishop. An introduction to the Kalman filter. *Tech. Report TR 95-041*, University of North Carolina at Chapel Hill, 2006.

[33] S. Yakowitz. Algorithms and computational techniques in differential dynamic programming. *Control and Dynamic Systems* 31:75–91, 1989.