Efficient Approximate Value Iteration for Continuous Gaussian POMDPs

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 introduce a highly efficient method for solving continuous partially-observable Markov decision processes (POMDPs) in which beliefs can be modeled using Gaussian distributions over the state space. Our method enables fast solutions to sequential decision making under uncertainty for a variety of problems involving noisy or incomplete observations and stochastic actions. We present an efficient approach to compute locally-valid approximations to the value function over continuous spaces in time polynomial ($O[n^4]$) in the dimension n of the state space. To directly tackle the intractability of solving general POMDPs, we leverage the assumption that beliefs are Gaussian distributions over the state space, approximate the belief update using an extended Kalman filter (EKF), and represent the value function by a function that is quadratic in the mean and linear in the variance of the belief. Our approach iterates towards a linear control policy over the state space that is locally-optimal with respect to a user defined cost function, and is approximately valid in the vicinity of a nominal trajectory through belief space. We demonstrate the scalability and potential of our approach on problems inspired by robot navigation under uncertainty for state spaces of up to 128 dimensions.

1 Introduction

Partially-observable Markov decision processes (POMDPs) (Kaelbling, Littman, and Cassandra 1998) provide a principled framework for sequential decision making under uncertainty for a variety of tasks involving noisy or incomplete observations and stochastic actions. The objective is to compute a policy defined as an optimal action for each possible belief in the state space such that the expected cost for completing the task is minimized. In this paper, we introduce a highly efficient method for solving continuous POMDPs in which beliefs can be modeled using Gaussian distributions over the state space. Problems that fall in this subclass of POMDPs span a variety of real-world applications, including modeling eye-hand coordination (Erez and Smart 2010) and navigating a robot toward a goal in the presence of motion uncertainty and noisy, incomplete sensing.

Computing exact solutions to general POMDPs, which has been shown to be PSPACE complete (C. Papadimitriou 1987), requires computing a policy over an infinite-dimensional *belief space*, the space of probability distributions over the (finite-dimensional) state space. For problems involving discrete state, action, and/or observation spaces, algorithms have been developed that use approximate value iteration with point-based updates (Pineau, Gordon, and Thrun 2003; Smith and Simmons 2004; Porta et al. 2006; Kurniawati, Hsu, and Lee 2008; Ong et al. 2010; Silver and Veness 2010; Kurniawati et al. 2011; Bai et al. 2011). For problems more naturally defined over continuous spaces (e.g. robot navigation), discretizing the problem and using the aforementioned approaches leads to an exponential growth in the number of states, inherently subjecting these problems to the "curse of dimensionality".

When considering continuous state and actions spaces, a key challenge is creating an efficient representation of the value function. The methods of (Thrun 2000; Brooks et al. 2006; Hauser 2011) handle continuous state and action spaces, but maintain a global (discrete) representation of the value function over the belief space, which limits their applicability to small to medium sized domains. Another class of methods avoids computation in the belief space by evaluating a large number of candidate trajectories in the state space (Prentice and Roy 2009; van den Berg, Abbeel, and Goldberg 2011; Bry and Roy 2011). These methods are not ideal because planning in state space will not yield optimal plans in the belief space.

For problems in which it is reasonable to model beliefs using Gaussian distributions, beliefs can be represented in parameterized form (Miller, Harris, and Chong 2009; Erez and Smart 2010; Platt et al. 2010; van den Berg, Patil, and Alterovitz 2011). These methods handle continuous state, action, and observation spaces and approximately compute the value function in parametric form only in local regions of the belief space, allowing for a running time polynomial in the dimension n of the state space. These methods consider a value function quadratic in the belief, leading to an $O[n^7]$ running time when a local optimization method such as differential dynamic programming (DDP) is used in belief space (van den Berg, Patil, and Alterovitz 2011). Local approaches can be extended to non-Gaussian beliefs (Platt et al. 2011) by using particle filters.

In this paper, we present a new method for computing locally optimal solutions to continuous POMDP's in which belief is modeled using Gaussian distributions. Our approach performs approximate value iteration over the belief space with a running time that is only $O[n^4]$ in the dimension n of the state space, which considerably improves upon prior work mentioned above and enables solving problems of higher dimensionality. The key insight that enables this is the representation of the value function by a function that is quadratic in the mean and *linear* in the variance of the belief. This representation naturally aligns with cost functions that are quadratic in the state since the expected cost is then quadratic in the mean and linear in the variance of the state (which follows from the identity $\mathrm{E}[\mathbf{x}^T Q \mathbf{x}] = \mathrm{E}[\mathbf{x}]^T Q \, \mathrm{E}[\mathbf{x}] + \mathrm{tr}[Q \, \mathrm{Var}[\mathbf{x}]]$ for any stochastic variable $\mathbf{x}$). In addition, our approach does not make the (erroneous) assumption that *maximum-likelihood observations* are received (which previous approaches do to obtain deterministic belief dynamics) and accounts for *stochastic* belief dynamics in the value iteration, resulting in more accurate solutions to continuous POMDPs. In addition, our approach handles hard constraints, e.g. obstacles in the environment.

We demonstrate the scalability and potential of our approach on problems with state spaces of up to 128 dimensions, and on a problem inspired by robotics involving a non-holonomic car-like robot navigating among obstacles.

2 Preliminaries and Definitions

We begin by defining POMDPs in their most general formulation. Then, we specifically state the instance of the problem we discuss in this paper.

General POMDPs

Let $\mathbb{X}$ be the space of all possible states $\mathbf{x}$ of the agent, $\mathbb{U}$ be the space of all possible control inputs $\mathbf{u}$ of the agent, and $\mathbb{Z}$ be the space of all possible sensor measurements $\mathbf{z}$ the agent may receive. The *belief* $\mathcal{X}_t$ is defined as the distribution of the state $\mathbf{x}_t$ at stage t given all past control inputs and sensor measurements. Let $\mathbb{B}$ denote the space of all possible beliefs. Given a control input $\mathbf{u}_t$ and a measurement $\mathbf{z}_{t+1}$, the belief is propagated using a *Bayesian filter*, which defines the *belief dynamics* written as a function $\beta : \mathbb{B} \times \mathbb{U} \times \mathbb{Z} \rightarrow \mathbb{B}$:

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

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

$$\mathrm{E}_{\mathbf{z}_1,\ldots,\mathbf{z}_\ell}[c_\ell[\mathcal{X}_\ell] + \textstyle\sum_{t=0}^{\ell-1} c_t[\mathcal{X}_t, \mathbf{u}_t]], \tag{2}$$

for given immediate cost functions $c_\ell : \mathbb{B} \rightarrow \mathbb{R}$ and $c_t : \mathbb{B} \times \mathbb{U} \rightarrow \mathbb{R}$. The expectation is taken given the stochastic nature of the measurements.

A general solution approach uses *value iteration* (Thrun, Burgard, and Fox 2005), a backward recursion procedure, to find the control policy π_t for each stage t:

$$v_\ell[\mathcal{X}] = c_\ell[\mathcal{X}], \tag{3}$$

$$v_t[\mathcal{X}] = \min_{\mathbf{u}}(c_t[\mathcal{X}, \mathbf{u}] + \mathop{\mathrm{E}}_{\mathbf{z}}[v_{t+1}[\beta[\mathcal{X}, \mathbf{u}, \mathbf{z}]]]), \tag{4}$$

where $v_t : \mathbb{B} \rightarrow \mathbb{R}$ is called the value function for stage t. The control policy $\pi_t[\mathcal{X}]$ for stage t is defined by the minimizing control input $\mathbf{u}$ in Eq. (4). Computing this control policy in practice is challenging, because in general the belief space $\mathbb{B}$ is infinite-dimensional and the value function cannot be expressed in parametric form.

Problem Definition

We will consider POMDPs in which the state, action, and observation spaces are continuous and the belief $\mathcal{X}_t = \mathcal{N}[\hat{\mathbf{x}}_t, \Sigma_t]$ is assumed to be a Gaussian distribution with mean $\hat{\mathbf{x}}_t$ and variance Σ_t. Specifically, we assume that $\mathbb{X} = \mathbb{R}^n$, $\mathbb{U} = \mathbb{R}^m$, and $\mathbb{Z} = \mathbb{R}^k$, and that we are given a (non-linear) stochastic dynamics and observation model:

$$\mathbf{x}_{t+1} = \mathbf{f}[\mathbf{x}_t, \mathbf{u}_t] + \mathbf{m}, \qquad \mathbf{m} \sim \mathcal{N}[\mathbf{0}, M[\mathbf{x}_t, \mathbf{u}_t]], \tag{5}$$

$$\mathbf{z}_t = \mathbf{h}[\mathbf{x}_t] + \mathbf{n}, \qquad \mathbf{n} \sim \mathcal{N}[\mathbf{0}, N[\mathbf{x}_t]], \tag{6}$$

where $\mathbf{m}$ and $\mathbf{n}$ are the motion and sensor noise, respectively, drawn from independent Gaussian distributions with zero mean and state and control input dependent variance.

Similar to the general POMDP case, our objective is to find control policies $\mathbf{u}_t = \pi_t[\hat{\mathbf{x}}_t, \Sigma_t]$ for all stages t that minimize the objective function of Eq. (2), for given immediate cost functions $c_\ell[\hat{\mathbf{x}}, \Sigma]$ and $c_t[\hat{\mathbf{x}}, \Sigma, \mathbf{u}]$. In our case, we require in addition positive-(semi)definiteness for the Hessian matrices of the immediate cost functions for all $0 \leq t < \ell$:

$$\frac{\partial^2 c_\ell}{\partial \hat{\mathbf{x}} \partial \hat{\mathbf{x}}} \geq 0, \quad \frac{\partial^2 c_t}{\partial \mathbf{u} \partial \mathbf{u}} > 0, \quad \begin{bmatrix} \frac{\partial^2 c_t}{\partial \hat{\mathbf{x}} \partial \hat{\mathbf{x}}} & \frac{\partial^2 c_t}{\partial \hat{\mathbf{x}} \partial \mathbf{u}} \\ \frac{\partial^2 c_t}{\partial \mathbf{u} \partial \hat{\mathbf{x}}} & \frac{\partial^2 c_t}{\partial \mathbf{u} \partial \mathbf{u}} \end{bmatrix} \geq 0, \tag{7}$$

Further, we assume that the initial belief $\mathcal{N}[\hat{\mathbf{x}}_0, \Sigma_0]$ is given.

3 Approach

Our approach computes a locally optimal solution to the continuous, Gaussian POMDP problem as formulated above. In our approach the belief dynamics are approximated using an extended Kalman filter, and the value function is approximated by a function that is quadratic in the mean and linear in the variance of the belief, and that is locally valid in the vicinity of a nominal trajectory though the belief space. A belief-space variant of LQG (linear-quadratic Gaussian) is used to perform value iteration, which results in linear control policies over the state space that are locally valid around the nominal trajectory. A locally-optimal solution to the POMDP problem is then found by iteratively generating nominal trajectories through execution of the control policies, and repeating the process until convergence. We discuss each of these steps in this section, and analyze the running time of our algorithm.

Belief Dynamics and the Extended Kalman Filter

Given a current belief $\mathcal{N}[\hat{\mathbf{x}}_t, \Sigma_t]$, 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. The EKF is widely used for state estimation of non-linear systems (Welch and Bishop 2006), and uses the first-order approximation that for any vector-valued function $\mathbf{f}[\mathbf{x}]$ of a stochastic variable $\mathbf{x}$ we have:

$$\begin{aligned} \mathrm{E}[\mathbf{f}[\mathbf{x}]] &\approx \mathbf{f}[\mathrm{E}[\mathbf{x}]], \\ \mathrm{Var}[\mathbf{f}[\mathbf{x}]] &\approx \frac{\partial \mathbf{f}}{\partial \mathbf{x}}[\mathrm{E}[\mathbf{x}]] \cdot \mathrm{Var}[\mathbf{x}] \cdot \frac{\partial \mathbf{f}}{\partial \mathbf{x}}[\mathrm{E}[\mathbf{x}]]^T. \end{aligned} \tag{8}$$

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

$$\hat{\mathbf{x}}_{t+1} = \mathbf{f}[\hat{\mathbf{x}}_t, \mathbf{u}_t] + K_t(\mathbf{z}_{t+1} - \mathbf{h}[\mathbf{f}[\hat{\mathbf{x}}_t, \mathbf{u}_t]]), \tag{9}$$

$$\Sigma_{t+1} = \Gamma_t - K_t H_t \Gamma_t, \tag{10}$$

where

$$\begin{aligned} \Gamma_t &= A_t \Sigma_t A_t^T + M[\hat{\mathbf{x}}_t, \mathbf{u}_t], \\ K_t &= \Gamma_t H_t^T (H_t \Gamma_t H_t^T + N[\mathbf{f}[\hat{\mathbf{x}}_t, \mathbf{u}_t]])^{-1}, \\ A_t &= \frac{\partial \mathbf{f}}{\partial \mathbf{x}}[\hat{\mathbf{x}}_t, \mathbf{u}_t], \quad H_t = \frac{\partial \mathbf{h}}{\partial \mathbf{x}}[\mathbf{f}[\hat{\mathbf{x}}_t, \mathbf{u}_t]]. \end{aligned}$$

Equations (9) and (10) define the (non-linear) belief dynamics. The second term of Eq. (9), 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. (6) and the assumptions of Eq. (8), the innovation term is distributed according to $\mathcal{N}[\mathbf{0}, K_t H_t \Gamma_t]$.

In summary, the stochastic belief dynamics are given by:

$$\hat{\mathbf{x}}_{t+1} = \mathbf{f}[\hat{\mathbf{x}}_t, \mathbf{u}_t] + \mathbf{w}, \quad \mathbf{w} \sim \mathcal{N}[\mathbf{0}, W[\hat{\mathbf{x}}_t, \Sigma_t, \mathbf{u}_t]], \tag{11}$$

$$\Sigma_{t+1} = \Phi[\hat{\mathbf{x}}_t, \Sigma_t, \mathbf{u}_t], \tag{12}$$

where:

$$W[\hat{\mathbf{x}}_t, \Sigma_t, \mathbf{u}_t] = K_t H_t \Gamma_t, \quad \Phi[\hat{\mathbf{x}}_t, \Sigma_t, \mathbf{u}_t] = \Gamma_t - K_t H_t \Gamma_t.$$

Note that in the deterministic part of the belief dynamics, the mean $\hat{\mathbf{x}}_{t+1}$ is not a function of the variance Σ_t. Also, only the evolution of the mean in the belief dynamics is stochastic. These precise conditions allow us to maintain the invariant that the value function is quadratic in the mean, and linear in the variance, as we see below.

Value Iteration

We perform value iteration backward in time to find a locally optimal control policy. We use a belief-space variant of LQG for doing so, and approximate the value function $v_t[\hat{\mathbf{x}}, \Sigma]$ as a function that is quadratic in the mean $\hat{\mathbf{x}}$ and linear in the variance Σ of the belief, and is approximately valid around a given nominal trajectory in belief space. Let the nominal trajectory be given as a series of beliefs (means and variances) and control inputs $(\bar{\mathbf{x}}_0, \bar{\Sigma}_0, \bar{\mathbf{u}}_0, \ldots, \bar{\mathbf{x}}_\ell, \bar{\Sigma}_\ell, \bar{\mathbf{u}}_\ell)$ such that $\bar{\mathbf{x}}_{t+1} = \mathbf{f}[\bar{\mathbf{x}}_t, \bar{\mathbf{u}}_t]$ and $\bar{\Sigma}_{t+1} = \Phi[\bar{\mathbf{x}}_t, \bar{\Sigma}_t, \bar{\mathbf{u}}_t]$ 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 represented by a function of the following form:

$$v_t[\hat{\mathbf{x}}, \Sigma] \approx s_t + \tfrac{1}{2}(\hat{\mathbf{x}} - \bar{\mathbf{x}}_t)^T S_t (\hat{\mathbf{x}} - \bar{\mathbf{x}}_t) + \mathbf{s}_t^T (\hat{\mathbf{x}} - \bar{\mathbf{x}}_t) + \mathbf{t}_t^T \operatorname{vec}[\Sigma - \bar{\Sigma}_t], \tag{13}$$

with $S_t \geq 0$. The notation $\operatorname{vec}[X]$ refers to the vector formed by stacking the columns of the matrix X. The value function is of a similar form as the cost-to-go function of traditional LQG controllers, with the key difference that a linear term in the variance of the state is added.

For the final time $t = \ell$, the value function v_ℓ (see Eq. (3)) is approximated by setting

$$s_\ell = c_\ell[\bar{\mathbf{x}}_\ell, \bar{\Sigma}_\ell], \qquad S_\ell = \frac{\partial^2 c_\ell}{\partial \hat{\mathbf{x}} \partial \hat{\mathbf{x}}}[\bar{\mathbf{x}}_\ell, \bar{\Sigma}_\ell],$$
$$\mathbf{s}_\ell^T = \frac{\partial c_\ell}{\partial \hat{\mathbf{x}}}[\bar{\mathbf{x}}_\ell, \bar{\Sigma}_\ell], \qquad \mathbf{t}_\ell^T = \frac{\partial c_\ell}{\partial \operatorname{vec}[\Sigma]}[\bar{\mathbf{x}}_\ell, \bar{\Sigma}_\ell], \tag{14}$$

which amounts to a second-order Taylor expansion for the mean and a first-order Taylor expansion for the variance of c_ℓ around the endpoint of the nominal trajectory. The value functions and the control policies for the stages $\ell > t \geq 0$ are computed by backward recursion; combining Eqs. (4), (11), (12), and (13) gives:

$$\begin{aligned}
v_t[\hat{\mathbf{x}}, \Sigma] &= \min_{\mathbf{u}} \big(c_t[\hat{\mathbf{x}}, \Sigma, \mathbf{u}] + \mathrm{E}_{\mathbf{w}}[s_{t+1} + \\
&\tfrac{1}{2}(\mathbf{f}[\hat{\mathbf{x}}, \mathbf{u}] + \mathbf{w} - \bar{\mathbf{x}}_{t+1})^T S_{t+1} (\mathbf{f}[\hat{\mathbf{x}}, \mathbf{u}] + \mathbf{w} - \bar{\mathbf{x}}_{t+1}) + \\
&\mathbf{s}_{t+1}^T (\mathbf{f}[\hat{\mathbf{x}}, \mathbf{u}] + \mathbf{w} - \bar{\mathbf{x}}_{t+1}) + \mathbf{t}_{t+1}^T \operatorname{vec}[\Phi[\hat{\mathbf{x}}, \Sigma, \mathbf{u}] - \bar{\Sigma}_{t+1}]]\big) \\
&= \min_{\mathbf{u}} \big(c_t[\hat{\mathbf{x}}, \Sigma, \mathbf{u}] + s_{t+1} + \\
&\tfrac{1}{2}(\mathbf{f}[\hat{\mathbf{x}}, \mathbf{u}] - \bar{\mathbf{x}}_{t+1})^T S_{t+1} (\mathbf{f}[\hat{\mathbf{x}}, \mathbf{u}] - \bar{\mathbf{x}}_{t+1}) + \\
&\mathbf{s}_{t+1}^T (\mathbf{f}[\hat{\mathbf{x}}, \mathbf{u}] - \bar{\mathbf{x}}_{t+1}) + \mathbf{t}_{t+1}^T \operatorname{vec}[\Phi[\hat{\mathbf{x}}, \Sigma, \mathbf{u}] - \bar{\Sigma}_{t+1}] + \\
&\tfrac{1}{2} \operatorname{vec}[S_{t+1}]^T \operatorname{vec}[W[\hat{\mathbf{x}}, \Sigma, \mathbf{u}]]\big),
\end{aligned} \tag{15}$$

where the last term in Eq. (15) 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 $\operatorname{tr}[QX] = \operatorname{vec}[Q^T]^T \operatorname{vec}[X]$. It is this term that ensures that the stochastic nature of the belief dynamics is accounted for in the value iteration.

To approximate the minimizing value of $\mathbf{u}$, we linearize the belief dynamics, and quadratize (for the mean and control input) and linearize (for the variance) the immediate cost function about the nominal trajectory. Given that $\bar{\mathbf{x}}_{t+1} = \mathbf{f}[\bar{\mathbf{x}}_t, \bar{\mathbf{u}}_t]$ and $\bar{\Sigma}_{t+1} = \Phi[\bar{\mathbf{x}}_t, \bar{\Sigma}_t, \bar{\mathbf{u}}_t]$, we get:

$$\mathbf{f}[\hat{\mathbf{x}}, \mathbf{u}] - \bar{\mathbf{x}}_{t+1} \approx F_t(\hat{\mathbf{x}} - \bar{\mathbf{x}}_t) + G_t(\mathbf{u} - \bar{\mathbf{u}}_t), \tag{16}$$

$$\operatorname{vec}[\Phi[\hat{\mathbf{x}}, \Sigma, \mathbf{u}] - \bar{\Sigma}_{t+1}] \approx T_t(\hat{\mathbf{x}} - \bar{\mathbf{x}}_t) + U_t \operatorname{vec}[\Sigma - \bar{\Sigma}_t] + V_t(\mathbf{u} - \bar{\mathbf{u}}_t), \tag{17}$$

$$\operatorname{vec}[W[\hat{\mathbf{x}}, \Sigma, \mathbf{u}]] \approx \mathbf{y}_t + X_t(\hat{\mathbf{x}} - \bar{\mathbf{x}}_t) + Y_t \operatorname{vec}[\Sigma - \bar{\Sigma}_t] + Z_t(\mathbf{u} - \bar{\mathbf{u}}_t), \tag{18}$$

$$c_t[\hat{\mathbf{x}}, \Sigma, \mathbf{u}] \approx q_t + \frac{1}{2} \begin{bmatrix} \hat{\mathbf{x}} - \bar{\mathbf{x}}_t \\ \mathbf{u} - \bar{\mathbf{u}}_t \end{bmatrix}^T \begin{bmatrix} Q_t & P_t^T \\ P_t & R_t \end{bmatrix} \begin{bmatrix} \hat{\mathbf{x}} - \bar{\mathbf{x}}_t \\ \mathbf{u} - \bar{\mathbf{u}}_t \end{bmatrix} + \begin{bmatrix} \mathbf{q}_t \\ \mathbf{r}_t \end{bmatrix}^T \begin{bmatrix} \hat{\mathbf{x}} - \bar{\mathbf{x}}_t \\ \mathbf{u} - \bar{\mathbf{u}}_t \end{bmatrix} + \mathbf{p}_t^T \operatorname{vec}[\Sigma - \bar{\Sigma}_t], \tag{19}$$

where:

$$\begin{aligned}
F_t &= \frac{\partial \mathbf{f}}{\partial \hat{\mathbf{x}}}[\bar{\mathbf{x}}_t, \bar{\mathbf{u}}_t], & G_t &= \frac{\partial \mathbf{f}}{\partial \mathbf{u}}[\bar{\mathbf{x}}_t, \bar{\mathbf{u}}_t], \\
T_t &= \frac{\partial \operatorname{vec}[\Phi]}{\partial \hat{\mathbf{x}}}[\bar{\mathbf{x}}_t, \bar{\Sigma}_t, \bar{\mathbf{u}}_t], & U_t &= \frac{\partial \operatorname{vec}[\Phi]}{\partial \operatorname{vec}[\Sigma]}[\bar{\mathbf{x}}_t, \bar{\Sigma}_t, \bar{\mathbf{u}}_t], \\
V_t &= \frac{\partial \operatorname{vec}[\Phi]}{\partial \mathbf{u}}[\bar{\mathbf{x}}_t, \bar{\Sigma}_t, \bar{\mathbf{u}}_t], & X_t &= \frac{\partial \operatorname{vec}[W]}{\partial \hat{\mathbf{x}}}[\bar{\mathbf{x}}_t, \bar{\Sigma}_t, \bar{\mathbf{u}}_t], \\
Y_t &= \frac{\partial \operatorname{vec}[W]}{\partial \operatorname{vec}[\Sigma]}[\bar{\mathbf{x}}_t, \bar{\Sigma}_t, \bar{\mathbf{u}}_t], & Z_t &= \frac{\partial \operatorname{vec}[W]}{\partial \mathbf{u}}[\bar{\mathbf{x}}_t, \bar{\Sigma}_t, \bar{\mathbf{u}}_t], \\
\mathbf{y}_t &= \operatorname{vec}[W[\bar{\mathbf{x}}_t, \bar{\Sigma}_t, \bar{\mathbf{u}}_t]], & q_t &= c_t[\bar{\mathbf{x}}_t, \bar{\Sigma}_t, \bar{\mathbf{u}}_t], \\
Q_t &= \frac{\partial^2 c_t}{\partial \hat{\mathbf{x}} \partial \hat{\mathbf{x}}}[\bar{\mathbf{x}}_t, \bar{\Sigma}_t, \bar{\mathbf{u}}_t], & \mathbf{q}_t^T &= \frac{\partial c_t}{\partial \hat{\mathbf{x}}}[\bar{\mathbf{x}}_t, \bar{\Sigma}_t, \bar{\mathbf{u}}_t], \\
R_t &= \frac{\partial^2 c_t}{\partial \mathbf{u} \partial \mathbf{u}}[\bar{\mathbf{x}}_t, \bar{\Sigma}_t, \bar{\mathbf{u}}_t], & \mathbf{r}_t^T &= \frac{\partial c_t}{\partial \mathbf{u}}[\bar{\mathbf{x}}_t, \bar{\Sigma}_t, \bar{\mathbf{u}}_t], \\
P_t &= \frac{\partial^2 c_t}{\partial \mathbf{u} \partial \hat{\mathbf{x}}}[\bar{\mathbf{x}}_t, \bar{\Sigma}_t, \bar{\mathbf{u}}_t], & \mathbf{p}_t^T &= \frac{\partial c_t}{\partial \operatorname{vec}[\Sigma]}[\bar{\mathbf{x}}_t, \bar{\Sigma}_t, \bar{\mathbf{u}}_t].
\end{aligned}$$

Filling in Eqs. (16)-(19) into Eq. (15), we get:

$$v_t[\hat{\mathbf{x}}, \Sigma] \approx \min_{\mathbf{u}} \Big(e_t + \frac{1}{2} \begin{bmatrix} \hat{\mathbf{x}} - \bar{\mathbf{x}}_t \\ \mathbf{u} - \bar{\mathbf{u}}_t \end{bmatrix}^T \begin{bmatrix} C_t & E_t^T \\ E_t & D_t \end{bmatrix} \begin{bmatrix} \hat{\mathbf{x}} - \bar{\mathbf{x}}_t \\ \mathbf{u} - \bar{\mathbf{u}}_t \end{bmatrix} + \begin{bmatrix} \mathbf{c}_t \\ \mathbf{d}_t \end{bmatrix}^T \begin{bmatrix} \hat{\mathbf{x}} - \bar{\mathbf{x}}_t \\ \mathbf{u} - \bar{\mathbf{u}}_t \end{bmatrix} + \mathbf{e}_t^T \operatorname{vec}[\Sigma - \bar{\Sigma}_t] \Big), \tag{20}$$

where:

$$\begin{aligned}
C_t &= Q_t + F_t^T S_{t+1} F_t, \quad D_t = R_t + G_t^T S_{t+1} G_t,\\
E_t &= P_t + G_t^T S_{t+1} F_t, \quad e_t = q_t + s_{t+1} + \tfrac{1}{2}\operatorname{vec}[S_{t+1}]^T \mathbf{y}_t,\\
\mathbf{c}_t^T &= \mathbf{q}_t^T + \mathbf{s}_{t+1}^T F_t + \mathbf{t}_{t+1}^T T_t + \tfrac{1}{2}\operatorname{vec}[S_{t+1}]^T X_t,\\
\mathbf{d}_t^T &= \mathbf{r}_t^T + \mathbf{s}_{t+1}^T G_t + \mathbf{t}_{t+1}^T V_t + \tfrac{1}{2}\operatorname{vec}[S_{t+1}]^T Z_t,\\
\mathbf{e}_t^T &= \mathbf{p}_t^T + \mathbf{t}_{t+1}^T U_t + \tfrac{1}{2}\operatorname{vec}[S_{t+1}]^T Y_t. \qquad (21)
\end{aligned}$$

The minimizing $\mathbf{u}$ is then found by taking the derivative with respect to $\mathbf{u}$ and equating to 0. This gives the solution:

$$\mathbf{u} = L_t(\hat{\mathbf{x}} - \bar{\mathbf{x}}_t) + \mathbf{l}_t + \bar{\mathbf{u}}_t \qquad (22)$$

(where $L_t = -D_t^{-1}E_t$ and $\mathbf{l}_t = -D_t^{-1}\mathbf{d}_t$), which defines the control policy $\mathbf{u} = \pi_t[\hat{\mathbf{x}}, \Sigma]$ for stage t.

Filling Eq. (22) back into Eq. (20) and collecting terms gives the value function $v_t[\hat{\mathbf{x}}, \Sigma]$ in the form of Eq. (13):

$$\begin{aligned}
s_t &= e_t + \tfrac{1}{2}\mathbf{d}_t^T \mathbf{l}_t, & S_t &= C_t + L_t^T E_t,\\
\mathbf{s}_t^T &= \mathbf{c}_t^T + \mathbf{l}_t^T E_t, & \mathbf{t}_t^T &= \mathbf{e}_t^T. \qquad (23)
\end{aligned}$$

This recursion then continues by computing the control policy and value function for stage $t-1$.

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 (Todorov and Li 2005; Jacobson and Mayne 1970). Given the initial belief $\hat{\mathbf{x}}_0, \Sigma_0$, and an initial nominal trajectory $(\bar{\mathbf{x}}_0^{(0)}, \bar{\Sigma}_0^{(0)}, \bar{\mathbf{u}}_0^{(0)}, \ldots, \bar{\mathbf{x}}_\ell^{(0)}, \bar{\Sigma}_\ell^{(0)}, \bar{\mathbf{u}}_\ell^{(0)})$ such that $\bar{\mathbf{x}}_0^{(0)} = \hat{\mathbf{x}}_0$, $\bar{\Sigma}_0^{(0)} = \Sigma_0$, and $\bar{\mathbf{x}}_{t+1}^{(0)} = \mathbf{f}[\bar{\mathbf{x}}_t^{(0)}, \bar{\mathbf{u}}_t^{(0)}]$ and $\bar{\Sigma}_{t+1}^{(0)} = \Phi[\bar{\mathbf{x}}_t^{(0)}, \bar{\Sigma}_t^{(0)}, \bar{\mathbf{u}}_t^{(0)}]$ for $0 \le t < \ell$, we proceed as follows.

Given the nominal trajectory of the $i-1$'th iteration (starting with $i = 1$), we find the control policy, i.e. the matrices L_t and vectors $\mathbf{l}_t$, using the value iteration procedure described above. We then compute the nominal trajectory $(\bar{\mathbf{x}}_0^{(i)}, \bar{\Sigma}_0^{(i}, \bar{\mathbf{u}}_0^{(i)}, \ldots, \bar{\mathbf{x}}_\ell^{(i)}, \bar{\Sigma}_\ell^{(i)}, \bar{\mathbf{u}}_\ell^{(i)})$ of the i'th iteration by setting $\bar{\mathbf{x}}_0^{(i)} = \hat{\mathbf{x}}_0$ and $\bar{\Sigma}_0^{(i)} = \Sigma_0$ and forward integrating the deterministic belief dynamics (for $0 \le t < \ell$) using the computed control policy:

$$\begin{aligned}
\bar{\mathbf{u}}_t^{(i)} &= L_t(\bar{\mathbf{x}}_t^{(i)} - \bar{\mathbf{x}}_t^{(i-1)}) + \mathbf{l}_t + \bar{\mathbf{u}}_t^{(i-1)},\\
\bar{\mathbf{x}}_{t+1}^{(i)} &= \mathbf{f}[\bar{\mathbf{x}}_t^{(i)}, \bar{\mathbf{u}}_t^{(i)}], \quad \bar{\Sigma}_{t+1}^{(i)} = \Phi[\bar{\mathbf{x}}_t^{(i)}, \bar{\Sigma}_t^{(i)}, \bar{\mathbf{u}}_t^{(i)}]. \qquad (24)
\end{aligned}$$

We then recompute the control policy, and reiterate. This lets the control policy converge to a local optimum (Liao and Shoemaker 1991).[1] We note that upon convergence the vectors $\mathbf{l}_t$ have become $\mathbf{0}$, so the ultimate result of the algorithm is a nominal trajectory and a linear control policy of the form $\mathbf{u}_t = \pi_t[\hat{\mathbf{x}}_t] = L_t(\hat{\mathbf{x}}_t - \bar{\mathbf{x}}_t) + \bar{\mathbf{u}}_t$.

[1]To ensure that the iteration in fact converges to a locally-optimal control policy, the algorithm is augmented with *line search* such that a new nominal trajectory is only accepted if it has a lower expected cost than the current nominal trajectory.

Time and Space Complexity Analysis

Let us analyze the running time of our algorithm. The dimension of the state is n, hence the size of its variance is n^2, and we assume that the dimensions of the control input $\mathbf{u}$ and the measurement $\mathbf{z}$ are $O[n]$. Further, we assume that the functions $\mathbf{f}$ and $\mathbf{h}$ can be evaluated in $O[n^2]$ time (this is the case if they were linear), that the functions M and N can be evaluated in $O[n^3]$ time (this is the case if they were linear), and that the immediate cost functions c_t can be evaluated in $O[n^2]$ time (this is the case if they were quadratic in the mean and control input, and linear in the variance).

As a result, the functions Φ and W each take $O[n^3]$ time to evaluate: they both involve performing a step of the extended Kalman filter, in which the Jacobians A_t and H_t are computed. Using numerical differentiation, this takes $O[n^3]$ time. The EKF further involves a constant number of multiplications and inversions of matrices of dimension $O[n \times n]$, which can also be done in $O[n^3]$ time.

In each step of the value iteration, the belief dynamics are linearized and the immediate cost function is quadratized for the mean and the control input, and linearized for the variance, which involves computing the Hessian and Jacobian matrices of Eqs. (16)-(19). Using numerical differentiation, the matrices F_t and G_t can be computed in $O[n^3]$ time and T_t, V_t, X_t, Z_t, Q_t, R_t, and P_t can be computed in $O[n^4]$ time. Computing the matrices U_t and Y_t using numerical differentiation would take $O[n^5]$ time, but it turns out that for each of the n^2 columns of U_t and Y_t we can find an analytic expression that takes $O[n^2]$ to evaluate. So also U_t and Y_t take $O[n^4]$ time to compute. The vectors $\mathbf{y}_t$, $\mathbf{q}_t$, and $\mathbf{r}_t$ are computed in $O[n^3]$ time, and $\mathbf{p}_t$ takes $O[n^4]$ time to compute. Lastly, the scalar q_t is computed in $O[n^2]$ time.

Other computations in each step of the value iteration are a constant number of multiplications and inversions of matrices. The most costly operations are the multiplications of an $n^2 \times n^2$ matrix (U_t and Y_t) with a vector of dimension n^2 ($\mathbf{t}_{t+1}$ and $\operatorname{vec}[S_{t+1}]$, respectively) in Eq. (21). This takes $O[n^4]$ time. All other matrix multiplications and inversions can be computed in at most $O[n^3]$ time. Hence, the total running time for a single step of the value iteration takes $O[n^4]$ time. A complete cycle of value iteration takes ℓ steps (ℓ being the time horizon), bringing the complexity to $O[\ell n^4]$. The number of value-iteration cycles needed to obtain convergence cannot be expressed in terms of n or ℓ.

In the exposition of our algorithm, all matrices and vectors that appear are of size at most $O[n^2]$, except for the matrices T_t, V_t, X_t, and Z_t, which are of dimension $O[n^2 \times n]$ (hence size $O[n^3]$), and the matrices U_t and Y_t, which are of dimension $n^2 \times n^2$ (hence size $O[n^4]$). Since these matrices are multiplied by a vector in all computations where they appear, and since they are computed column by column (each of $O[n^2]$ size), we never need to store them in memory in their entirety, but only a column at a time. Hence, the total storage requirement of our algorithm is $O[n^2]$ per step of the value iteration, and $O[\ell n^2]$ in total, since all matrices L_t and $\bar{\Sigma}_t$ (for $0 \le t < \ell$) are stored in memory. We further note that in our implementation we exploit the symmetry of the matrices S, Σ, Φ, and W to reduce the running time and storage requirements by a constant factor.

4 State Constraints and Obstacles

We presented our approach above for general immediate cost functions $c_\ell[\hat{\mathbf{x}}, \Sigma]$ and $c_t[\hat{\mathbf{x}}, \Sigma, \mathbf{u}]$ (with the requirements of Eq. (7)). In typical LQG-style cost functions, the existence of constraints on the state (e.g. due to obstacles in the environment) is not incorporated. To consider constraints and maximize the probability of satisfying them, we incorporate constraints into the cost functions as follows.

Let $\mathbb{F} \subset \mathbb{X}$ be the valid region of the state space defining the constraints on the state. Given a belief $\hat{\mathbf{x}}, \Sigma$, the probability of satisfying the constraints, is given by the integral over $\mathbb{F}$ of the probability-density function of $\mathcal{N}[\hat{\mathbf{x}}, \Sigma]$. We approximate this probability as follows. First, we choose a maximal convex region in $\mathbb{F}$ around the current belief, resulting in a set of linear constraints $\{\mathbf{a}_i^T \mathbf{x} < b_i\}$. The probability that constraint i is satisfied is given by:

$$p[\mathbf{a}_i^T \mathbf{x} < b_i] = \mathrm{cdf}[(b_i - \mathbf{a}_i^T \hat{\mathbf{x}})/\sqrt{\mathbf{a}_i^T \Sigma \mathbf{a}_i}], \qquad (25)$$

where $\mathrm{cdf} : \mathbb{R} \to \mathbb{R}$ denotes the cumulative density function of the standard Gaussian distribution $\mathcal{N}[0, 1]$. The probability that all constraints are satisfied is approximated by:

$$p[\forall_i : \mathbf{a}_i^T \mathbf{x} < b_i] \approx \textstyle\prod_i \mathrm{cdf}[(b_i - \mathbf{a}_i^T \hat{\mathbf{x}})/\sqrt{\mathbf{a}_i^T \Sigma \mathbf{a}_i}], \quad (26)$$

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. Therefore, we add to $c_t[\hat{\mathbf{x}}, \Sigma, \mathbf{u}]$ the term

$$f[\hat{\mathbf{x}}, \Sigma] = -\textstyle\sum_i \log[\mathrm{cdf}[(b_i - \mathbf{a}_i^T \hat{\mathbf{x}})/\sqrt{\mathbf{a}_i^T \Sigma \mathbf{a}_i}]] \qquad (27)$$

to account for the probability of violating constraints. We note that $\frac{\partial^2 f}{\partial \hat{\mathbf{x}} \partial \hat{\mathbf{x}}} \geq 0$, as is required by Eq. (7).

5 Results

We evaluate our approach in simulation in robot navigation scenarios involving continuous state, action, and observation spaces with stochastic dynamics and observation models with state and control-dependent noise and spatially-varying sensing capabilities. We consider two scenarios: (i) n-D point robot with linear dynamics that uses beacon-based localization, and (ii) a 4-D under-actuated car-like robot with second-order dynamics navigating in a 2-D environment. The method was implemented in C++ and evaluated on a 3.33 Ghz Intel® i7™ PC.

In the following experiments, we define the cost functions in Eq. (2) to be:

$$c_\ell[\hat{\mathbf{x}}_\ell, \Sigma_\ell] = \hat{\mathbf{x}}_\ell^T Q_\ell \hat{\mathbf{x}}_\ell + \mathrm{tr}[Q_\ell \Sigma_\ell], \qquad (28)$$

$$c_t[\hat{\mathbf{x}}_t, \Sigma_t, \mathbf{u}_t] = \mathbf{u}_t^T R_t \mathbf{u}_t + \mathrm{tr}[Q_t \Sigma_t] + f[\hat{\mathbf{x}}_t, \Sigma_t], \qquad (29)$$

for given $Q_t \geq 0$ and $R_t > 0$. The term $\hat{\mathbf{x}}_\ell^T Q_\ell \hat{\mathbf{x}}_\ell + \mathrm{tr}[Q_\ell \Sigma_\ell] = \mathrm{E}[\mathbf{x}_\ell^T Q_\ell \mathbf{x}_\ell]$ 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 in the state, and $f[\hat{\mathbf{x}}_t, \Sigma_t]$ encodes the state constraint/obstacle cost (if applicable).

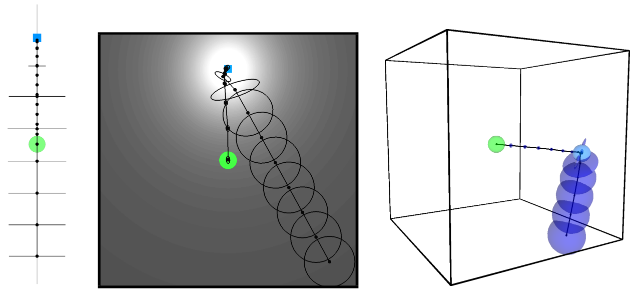

(a) (b) (c)

Figure 1: n-D point robot navigating to the origin (shown in green) in a unit hypercube with beacon-based localization (beacon shown in blue). (a) 1-D scenario where the beacon and initial state of the robot are on either side of the goal. The variance is indicated by the error bars. The robot goes to the beacon before backtracking to the goal. (b) 2-D scenario, and (c) 3-D scenario, where the robot localizes itself at the beacon before reaching the goal with significantly reduced uncertainty (variance indicated using error ellipsoids).

n-D Point Robot With Beacon-Based Localization

To evaluate the scalability of our approach, we consider a point robot with linear dynamics navigating in an n-D unit hypercube centered at the origin, using beacon-based localization. This gives the following stochastic dynamics model:

$$\mathbf{x}_{t+1} = \mathbf{f}[\mathbf{x}_t, \mathbf{u}_t] + \mathbf{m} = \mathbf{x}_t + \tau \mathbf{u}_t + \mathbf{m}, \qquad (30)$$

where $\mathbf{x}_t \in \mathbb{R}^n$ is the robot's position, $\mathbf{u}_t \in \mathbb{R}^n$ is the robot's velocity, τ is the time step, and the noise $\mathbf{m} \sim \mathcal{N}[\mathbf{0}, M[\mathbf{u}_t]]$ is scaled proportional to the control input $\mathbf{u}_t$.

We consider a partially-observable scenario where the robot localizes itself using signal measurements from a beacon placed in the environment at location $\check{\mathbf{x}}$. The signal strength decays quadratically with the distance to the beacon $||\mathbf{x} - \check{\mathbf{x}}||_2$ due to the decreased signal-to-noise ratio. For the purposes of constructing an observation model that scales to arbitrarily high dimensions, we also scale the signal strength by a normalization factor to preserve scaling of the Euclidean norm ($|| \cdot ||_2$) as the dimension of the state space increases. We chose the normalization factor to be $\sqrt{n}$, which is the length of the longest diagonal of a n-D unit hypercube. This gives us the following non-linear observation model:

$$\mathbf{z}_t = \mathbf{h}[\mathbf{x}_t] + \mathbf{n} = (\sqrt{n})^2/(1 + ||\mathbf{x}_t - \check{\mathbf{x}}||_2^2) + \mathbf{n}, \qquad (31)$$

where $\mathbf{z}_t \in \mathbb{R}$ consists of a signal strength reading from the beacon, and the observation noise $\mathbf{n} \sim \mathcal{N}[\mathbf{0}, N]$ is considered constant.

In our experiments, the goal is placed at the origin, the robot is initialized with a belief state that consists of a randomly sampled state within the unit hypercube and a variance of $0.1I$. The location of the beacon is also chosen at random. We use control and state cost matrices of $R_t = I$, $Q_t = 10I$, and $Q_\ell = 10\ell I$ where $\ell = 15$. We initialize the method with a straight line trajectory from the mean of initial belief to the goal. Due to high initial uncertainty in position, the optimal strategy for the robot is typically to move to the beacon for precise localization before moving the goal (Fig. 1(a), 1(b), and 1(c)).

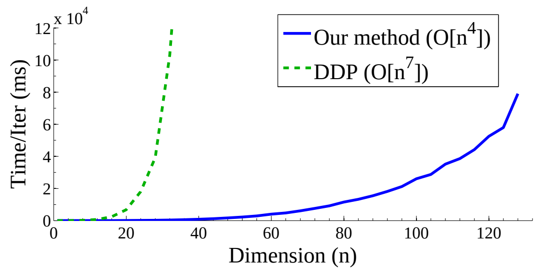


Figure 2: Variation in the average time per iteration (ms) versus the number of dimensions $(1-128)$. Our method (solid blue line) has lower computation times as compared to state of the art prior work based on DDP (van den Berg, Patil, and Alterovitz 2011) (dashed green line) and scales as the dimension increases, which enables us to solve considerably higher dimensional problems.

We evaluate the scalability of our approach with respect to the number of dimensions of the state. Fig. 2 shows the variation in the time required per iteration of our method as the dimension of the state space increases from 1 to 128, averaged over 100 trials per dimension. For each trial in each dimension, we randomly sampled the initial state of the robot and the beacon location within the unit hypercube. The growth in the average time per iteration (in milliseconds) is bound by the predicted $O[n^4]$ complexity. We also compare our method against a $O[n^7]$ approach based on differential dynamic programming (DDP) (van den Berg, Patil, and Alterovitz 2011). The average time per iteration of our method is considerably lower and scales as the state space dimension increases as compared to prior work. Our approach is able to compute locally-optimal solutions in continuous spaces within minutes on commodity hardware for up to 128 dimensional spaces.

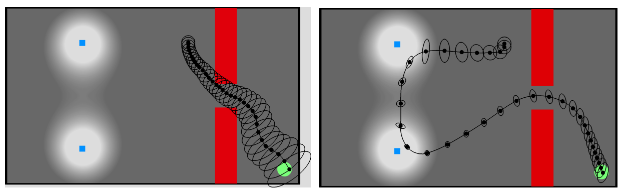
Table 1 provides more detail for some of the lower-dimensional experiments, and shows that the average number of iterations and total computation time increases as the dimensionality increases. It should be noted that even though the DDP based approach demonstrates quadratic convergence (Liao and Shoemaker 1991) and requires lesser number of iterations on average as compared to our approach, it does not scale well to higher dimensions.

	Our method ($O[n^4]$)		DDP ($O[n^7]$)	
Dim	Num iterations	Total time (ms)	Num iterations	Total time (ms)
1	13 (± 4)	9 (± 3)	7 (± 2)	28 (± 9)
2	31 (± 8)	40 (± 11)	18 (± 6)	203 (± 72)
4	48 (± 15)	146 (± 44)	20 (± 11)	817 (± 419)
8	60 (± 14)	575 (± 132)	25 (± 8)	9.4e3 (± 2.3e3)
16	72 (± 13)	3.7e3 (± 640)	29 (± 12)	1.8e5 (± 7.5e4)
32	88 (± 19)	2.9e4 (± 6.2e3)	No solution found	

Table 1: Comparison of our method with prior work over 100 trials. Standard deviations provided in parentheses.

4-D Under-Actuated Car-Like Robot

We consider the case of a non-holonomic car-like robot navigating in a partially-observable 2-D environment with obstacles. 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. This gives the following



(a) Input trajectory (b) Optimal trajectory

Figure 3: Under-actuated car-like robot navigating to the goal (shown in green) in a partially-observable environment with beacon-based localization (beacons shown in blue). (a) Naïve collision-free trajectory computed using an uncertainty-unaware planner accumulates considerable uncertainty during execution in the narrow passage. (b) The locally-optimal solution guides the robot to both beacons before it arrives at the goal through the narrow passage with considerably reduced uncertainty.

stochastic non-linear dynamics model:

$$\mathbf{x}_{t+1} = \mathbf{f}[\mathbf{x}_t, \mathbf{u}_t] + \mathbf{m} = \begin{bmatrix} x_t + \tau v_t \cos\theta_t \\ y_t + \tau v_t \sin\theta_t \\ \theta_t + \tau v_t \tan[\phi_t]/d \\ v_t + \tau a_t \end{bmatrix} + \mathbf{m}, \quad (32)$$

where τ is the time step, d is the length of the car-like robot, and $\mathbf{m} \sim \mathcal{N}[\mathbf{0}, M[\mathbf{u}_t]]$ scales the noise proportional to the control input $\mathbf{u}_t$. The measurement vector $\mathbf{z}_t \in \mathbb{R}^3$ consists of two signal measurements from two beacons placed in the environment (similar to the beacon considered above) and speed measurement from a speedometer:

$$\mathbf{z}_t = \mathbf{h}[\mathbf{x}_t] + \mathbf{n} = \begin{bmatrix} 1/(1 + ||\mathbf{x}_t - \check{\mathbf{x}}_1||_2^2) \\ 1/(1 + ||\mathbf{x}_t - \check{\mathbf{x}}_2||_2^2) \\ v_t \end{bmatrix} + \mathbf{n}, \quad (33)$$

with constant measurement noise variance $\mathbf{n} \sim \mathcal{N}[\mathbf{0}, N]$.

Our method takes as input a collision-free trajectory to the goal computed using a RRT planner (LaValle and Kuffner 2001), which only generates a path and does not consider uncertainty. If the robot were to follow this input trajectory in an open-loop fashion, noisy actuation would likely result it collision with an obstacle before reaching the goal (Fig. 3(a)). 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 (Fig. 3(b)). Notice how the optimal trajectory visits both beacons for reliable localization. The nominal trajectory traverses the narrow passage along the medial axis and the remainder of the nominal trajectory is revised to stay away from the boundaries of the environment, to minimize likelihood of colliding with obstacles. It took 4.84 seconds to converge to the locally-optimal solution over 52 iterations.

6 Conclusion and Future Work

We presented a highly efficient algorithm for solving continuous POMDPs in which beliefs can be modeled using Gaussian distributions over the state space. Our approach performs approximate value iteration over the belief space with a running time that is only $O[n^4]$ in the dimension n of the state space, which we showed enables solving problems with state spaces of up to 128 dimensions. Further, our approach generalizes earlier work on Gaussian-based POMDPs by removing the assumption that maximum-likelihood observations are received.

Our approach has several limitations. First, we represent beliefs using Gaussian distributions. This may not be an acceptable approximation in some applications, for instance 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 and our approach should be directly applicable in such applications. 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 where there are abrupt boundaries between sensing regimes (e.g. inside or outside the field of view of a camera).

In future work we plan to apply our method to real-world problems that involve complex dynamics and that would benefit from a fast continuous Gaussian POMDP solver, including autonomous quadrotor flight, medical needle steering, and manipulation of deformable tissue.

References

Bai, H.; Hsu, D.; Lee, W.; and Ngo, V. 2011. Monte Carlo Value Iteration for Continuous-State POMDPs. *Algorithmic Foundations of Robotics IX* 175–191.

Brooks, A.; Makarenko, A.; Williams, S.; and Durrant-Whyte, H. 2006. Parametric POMDPs for Planning in Continuous State Spaces. *Robotics and Autonomous Systems* 54(11):887–897.

Bry, A., and Roy, N. 2011. Rapidly-exploring Random Belief Trees for Motion Planning Under Uncertainty. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 723–730.

C. Papadimitriou, J. T. 1987. The Complexity of Markov Decision Processes. *Mathematics of Operations Research* 12(3):441–450.

Erez, T., and Smart, W. D. 2010. A Scalable Method for Solving High-Dimensional Continuous POMDPs Using Local Approximation. In *Conf. on Uncertainty in Artificial Intelligence*, 160–167.

Hauser, K. 2011. Randomized Belief-Space Replanning in Partially-Observable Continuous Spaces. *Algorithmic Foundations of Robotics IX* 193–209.

Jacobson, D., and Mayne, D. 1970. *Differential Dynamic Programming*. American Elsevier Publishing Company, Inc.

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1998. Planning and Acting in Partially Observable Stochastic Domains. *Artificial Intelligence* 101(1-2):99–134.

Kurniawati, H.; Du, Y.; Hsu, D.; and Lee, W. 2011. Motion Planning under Uncertainty for Robotic Tasks with Long Time Horizons. *Int. Journal of Robotics Research* 30(3):308–323.

Kurniawati, H.; Hsu, D.; and Lee, W. S. 2008. SARSOP: Efficient Point-based POMDP Planning by Approximating Optimally Reachable Belief Spaces. In *Robotics: Science and Systems (RSS)*.

LaValle, S., and Kuffner, J. 2001. Randomized Kinodynamic Planning. *Int. Journal of Robotics Research* 20(5):378–400.

Liao, L. Z., and Shoemaker, C. A. 1991. Convergence in Unconstrained Discrete-Time Differential Dynamic Programming. *IEEE Trans. on Automatic Control* 36(6):692–706.

Miller, S. A.; Harris, Z. A.; and Chong, E. K. P. 2009. Coordinated Guidance of Autonomous UAVs via Nominal Belief-State Optimization. In *American Control Conference (ACC)*, 2811–2818.

Ong, S.; Png, S.; Hsu, D.; and Lee, W. S. 2010. Planning under Uncertainty for Robotic Tasks with Mixed Observability. *Int. Journal of Robotics Research* 29(8):1053–1068.

Pineau, J.; Gordon, G.; and Thrun, S. 2003. Point-based Value Iteration: An Anytime Algorithm for POMDPs. In *Int. Joint Conf. on Artificial Intelligence*, 1025–1032.

Platt, R.; Tedrake, R.; Kaelbling, L.; and Lozano-Perez, T. 2010. Belief Space Planning assuming Maximum Likelihood Observations. In *Robotics: Science and Systems (RSS)*.

Platt, R.; Kaelbling, L.; Lozano-Perez, T.; and Tedrake, R. 2011. Efficient Planning in Non-Gaussian Belief Spaces and its Application to Robot Grasping. In *Int. Symp. on Robotics Research (ISRR)*.

Porta, J.; Vlassis, N.; Spaan, M.; and Poupart, P. 2006. Point-based Value Iteration for Continuous POMDPs. *Journal of Machine Learning Research* 7:2329–2367.

Prentice, S., and Roy, N. 2009. The Belief Roadmap: Efficient Planning in Belief Space by Factoring the Covariance. *Int. Journal of Robotics Research* 28(11–12):1448–1465.

Silver, D., and Veness, J. 2010. Monte-Carlo Planning in Large POMDPs. In *Advances in Neural Information Processing Systems (NIPS)*. 2164–2172.

Smith, T., and Simmons, R. 2004. Heuristic Search Value Iteration for POMDPs. In *Proc. Conf. on Uncertainty in Artificial Intelligence*, 520–527.

Thrun, S.; Burgard, W.; and Fox, D. 2005. *Probabilistic Robotics*. MIT Press.

Thrun, S. 2000. Monte Carlo POMDPs. *Advances in Neural Information Processing Systems* 12:1064–1070.

Todorov, E., and Li, W. 2005. A Generalized Iterative LQG Method for Locally-Optimal Feedback Control of Constrained Nonlinear Stochastic Systems. In *Proc. American Control Conference (ACC)*, 300–306.

van den Berg, J.; Abbeel, P.; and Goldberg, K. 2011. LQG-MP: Optimized Path Planning for Robots with Motion Uncertainty and Imperfect State Information. *Int. Journal of Robotics Research* 30(7):895–913.

van den Berg, J.; Patil, S.; and Alterovitz, R. 2011. Motion Planning under Uncertainty using Differential Dynamic Programming in Belief Space. In *Int. Symp. on Robotics Research (ISRR)*.

Welch, G., and Bishop, G. 2006. An Introduction to the Kalman Filter. Technical Report TR 95-041, Univ. North Carolina at Chapel Hill.