Stochastic Extended LQR: Optimization-based Motion Planning Under Uncertainty

Wen Sun[1], Jur van den Berg[2], and Ron Alterovitz[1]

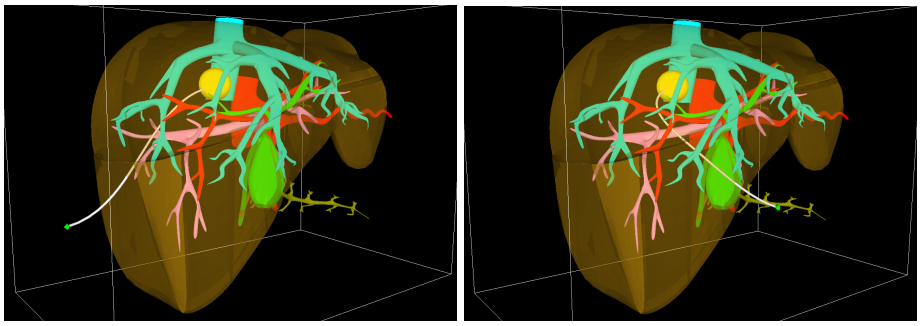
[1] University of North Carolina at Chapel Hill, USA
{wens, ron}@cs.unc.edu
[2] University of Utah, USA
berg@cs.utah.edu

Abstract. We introduce a novel optimization-based motion planner, Stochastic Extended LQR (SELQR), which computes a trajectory and associated linear control policy with the objective of minimizing the expected value of a user-defined cost function. SELQR applies to robotic systems that have stochastic non-linear dynamics with motion uncertainty modeled by Gaussian distributions that can be state- and control-dependent. In each iteration, SELQR uses a combination of forward and backward value iteration to estimate the cost-to-come and the cost-to-go for each state along a trajectory. SELQR then locally optimizes each state along the trajectory at each iteration to minimize the expected total cost, which results in smoothed states that are used for dynamics linearization and cost function quadratization. SELQR progressively improves the approximation of the expected total cost, resulting in higher quality plans. For applications with imperfect sensing, we extend SELQR to plan in the robot's belief space. We show that our iterative approach achieves fast and reliable convergence to high-quality plans in multiple simulated scenarios involving a car-like robot, a quadrotor, and a medical steerable needle performing a liver biopsy procedure.

1 Introduction

When a robot performs a task, the robot's motion may be affected by uncertainty from a variety of sources, including unpredictable external forces or actuation errors. Uncertainty arises in a variety of robotics applications, including aerial robots moving in turbulent conditions, mobile robots maneuvering on unfamiliar terrain, and robotic steerable needles being guided to clinical targets in soft tissue. A deliberative approach that accounts for uncertainty during motion planning before task execution can improve the quality of computed plans, increasing the chances that the robot will complete the desired motion safely and reliably.

We introduce an optimization-based motion planner that explicitly considers the impacts of motion uncertainty. Recent years have seen the introduction of multiple successful optimization-based planners, although most have focused on robots with deterministic dynamics (e.g., [1,2,3]). Compared to commonly used



(a) SELQR trajectory inserted from side (b) SELQR trajectory inserted from front

Fig. 1. We show plans computed by SELQR for needle steering for a liver biopsy with motion uncertainty. The objective is to access the tumor (yellow) while avoiding the hepatic arteries (red), hepatic veins (blue), portal veins (pink), and bile ducts (green). The smooth trajectories explicitly consider uncertainty and minimize the *a priori* expected value of a cost function that considers obstacle avoidance and path length.

sampling-based planners [4], optimization-based planners produce plans that are smoother (without requiring a separate smoothing algorithm) and that are computed faster, albeit sometimes with a loss of completeness and global optimality. Prior optimization-based planners that consider deterministic dynamics can only minimize deterministic cost functions (e.g., minimizing path length while avoiding obstacles). In this paper we focus on robots with stochastic dynamics, and consequently minimize the *a priori* expected value of a cost function when a plan and corresponding controller are executed. The user-defined cost function can be based on path length, control effort, and obstacle collision avoidance.

We first introduce the Stochastic Extended LQR (SELQR) motion planner, a novel optimization-based motion planner with fast and reliable convergence for robotic systems with non-linear dynamics, any cost function with positive (semi)definite Hessians, and motion uncertainty modeled using Gaussian distributions that can be state- and control-dependent. Our approach builds on the linear quadratic regulator (LQR), a commonly used linear controller that does not explicitly consider obstacle avoidance. As an optimization-based approach, SELQR starts motion planning from a start state and returns a high-quality trajectory and an associated linear control policy that consider uncertainty and are optimized with respect to the given cost function.

To achieve fast performance, our approach in each iteration uses both the stochastic forward and inverse dynamics in a manner inspired by an iterated Kalman smoother [5]. In each iteration's backward pass, SELQR uses the stochastic dynamics to compute a control policy and estimate the cost-to-go of each state, which is the minimum expected future cost assuming the robot starts from each state. In each iteration's forward pass, SELQR estimates the cost-to-come to each state, which is the minimum cost to reach each state from the initial state. SELQR then approximates the expected total cost at each state

by summing the cost-to-come and the cost-to-go. SELQR progressively improves the approximation of the cost-to-come and cost-to-go and hence improves its estimate of the expected total cost. A key insight in SELQR is that we locally optimize each state along a trajectory at each iteration to minimize the expected total cost, which results in smoothed states that are cost-informative and used for dynamics linearization and cost function quadratization. These smoothed states enable the fast and reliable convergence of SELQR.

We next extend SELQR to consider uncertainty in both motion and sensing. Although the robot in such cases often cannot directly observe its current state, it can estimate a distribution over the set of possible states (i.e., its belief state) based on noisy and partial sensor measurements. We introduce B-SELQR, a variant of SELQR that plans in belief space rather than state space for robots with both motion and sensing uncertainty, where belief states are modeled with Gaussian distributions. For such robots, the motion planning problem can be modeled as a Partially Observable Markov Decision Process (POMDP). Exact global optimal solutions to POMDPs are prohibitive for most applications since the belief space (over which a control policy is to be computed) is, in the most general formulation, the infinite-dimensional space of all possible probability distributions over the finite dimensional state space. B-SELQR quickly computes a trajectory and locally-valid controller from scratch in belief space.

We demonstrate the speed and effectiveness of SELQR in simulation for a car-like robot, quadrotor, and medical steerable needle (see Fig. 1). We also demonstrate B-SELQR for scenarios with imperfect sensing.

2 Related Work

Optimization-based motion planners have been studied for a variety of robotics applications and typically consider robot dynamics, trajectory smoothness, and obstacle avoidance. Optimization-based approaches have been developed that plan from scratch as well as that locally optimize a feasible plan created by another motion planner (such as a sampling-based motion planner), e.g. [1,3,2,6,7,8]. These methods work well for robots with deterministic dynamics, whereas SELQR is intended for robots with stochastic dynamics.

Our approach builds on Extended LQR [9,10], which extends the standard LQR to handle non-linear dynamics and non-quadratic cost functions. Extended LQR assumes deterministic dynamics, implicitly relying on the fact that the optimal LQR solution is independent of the variance of the motion uncertainty. In contrast to Extended LQR, SELQR explicitly considers stochastic dynamics and incorporates the stochastic dynamics into backward value iteration when computing a control policy, enabling computation of higher quality plans. Approximate Inference Control [11] formulates the optimal control problem using Kullback-Leibler divergence minimization but focuses on cost functions that are quadratic in the control input. Our approach also builds on Iterative Linear Quadratic Gaussian (iLQG) [12], which uses a quadratic approximation to handle state- and control-dependent motion uncertainty but, in its original form, did not implement obstacle avoidance. To ensure that the dynamics linearization and

cost function quadratization are locally valid, iLQG requires special measures such as a line search. Our method does not require a line search, enabling faster performance.

For problems with partial or noisy sensing, the planning and control problem can be modeled as a POMDP [13]. Solving a POMDP to global optimality has been shown to be PSPACE complete. Point-based algorithms (e.g., [14,15,16]) have been developed for problems with discrete state, action, or observation spaces. Another class of methods [17,18,19] utilize sampling-based planners to compute candidate trajectories in the state space, which can be evaluated based on metrics that consider stochastic dynamics. Optimization-based approaches have been developed for planning in belief space [20,21] by approximating beliefs as Gaussian distributions and computing a value function valid only in local regions of the belief space. Platt et al. [21] achieve fast performance by defining deterministic belief system dynamics based on the maximum likelihood observation assumption. Van den Berg et al. [20] require a feasible plan for initialization and then use iLQG to optimize the plan in belief space. We will show that BSELQR, which considers stochastic dynamics, converges faster and more reliably than using iLQG in belief space and can plan from scratch.

3 Problem Definition

Let $\mathbb{X} \subset \mathbb{R}^n$ be the n-dimensional state space of the robot and let $\mathbb{U} \subset \mathbb{R}^m$ be the m-dimensional control input space of the robot. We consider robotic systems with differentiable stochastic dynamics and state- and control-dependent uncertainty modeled using Gaussian distributions. Let $\tau \in \mathbb{R}^+$ denote time, and let us be given a continuous-time stochastic dynamics:

$$d\mathbf{x}(\tau) = \mathbf{f}(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau)d\tau + N(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau)d\mathbf{w}(\tau), \tag{1}$$

with $\mathbf{f} : \mathbb{X} \times \mathbb{U} \times \mathbb{R}^+ \to \dot{\mathbb{X}}$ and $N : \mathbb{X} \times \mathbb{U} \times \mathbb{R}^+ \to \mathbb{R}^{n \times n}$, where $\mathbf{x}(\tau) \in \mathbb{X}$, $\mathbf{u}(\tau) \in \mathbb{U}$, and $\mathbf{w}(\tau)$ is a Wiener process with $d\mathbf{w}(\tau) \sim \mathcal{N}(\mathbf{0}, d\tau I)$.

We assume time is discretized into intervals of duration Δ, and the time step $t \in \mathbb{N}$ starts at time $\tau = t\Delta$. As we will see in Sec. 4.5, by integrating the continuous time dynamics both backward and forward in time, we can construct the stochastic discrete dynamics and the deterministic inverse discrete dynamics:

$$\mathbf{x}_{t+1} = \mathbf{g}_t(\mathbf{x}_t, \mathbf{u}_t) + M_t(\mathbf{x}_t, \mathbf{u}_t)\boldsymbol{\xi}_t, \tag{2}$$

$$\mathbf{x}_t = \bar{\mathbf{g}}_t(\mathbf{x}_{t+1}, \mathbf{u}_t), \tag{3}$$

where $\boldsymbol{\xi}_t \sim \mathcal{N}(\mathbf{0}, I)$, with $\mathbf{g}_t, \bar{\mathbf{g}}_t \in \mathbb{X} \times \mathbb{U} \to \mathbb{X}$ and $M_t \in \mathbb{X} \times \mathbb{U} \to \mathbb{R}^{n \times n}$ as derived in Sec. 4.5. Note that $\mathbf{g}_t(\bar{\mathbf{g}}_t(\mathbf{x}_{t+1}, \mathbf{u}_t), \mathbf{u}_t) = \mathbf{x}_{t+1}$ and $\bar{\mathbf{g}}_t(\mathbf{g}_t(\mathbf{x}_t, \mathbf{u}_t), \mathbf{u}_t) = \mathbf{x}_t$.

Let the control objective be defined by a cost function that can incorporate metrics such as path length, control effort, and obstacle avoidance:

$$\mathbb{E}_{\mathbf{x}}\left[c_l(\mathbf{x}_l) + \sum_{t=0}^{l-1} c_t(\mathbf{x}_t, \mathbf{u}_t)\right], \tag{4}$$

where $l \in \mathbb{N}^+$ is the given time horizon and $c_l : \mathbb{X} \to \mathbb{R}$ and $c_t : \mathbb{X} \times \mathbb{U} \to \mathbb{R}$ are user-defined local cost functions. The expectation is taken because the dynamics

are stochastic. We assume the local cost functions are twice differentiable and have positive (semi)definite Hessians: $\frac{\partial^2 c_l}{\partial \mathbf{x} \partial \mathbf{x}} \geq 0$, $\frac{\partial^2 c_t}{\partial \mathbf{u} \partial \mathbf{u}} > 0$, $\frac{\partial^2 c_t}{\partial [\substack{\mathbf{x} \\ \mathbf{u}}] \partial [\substack{\mathbf{x} \\ \mathbf{u}}]} \geq 0$. The objective is to compute a control policy $\boldsymbol{\pi}$ (defined by $\boldsymbol{\pi}_t : \mathbb{X} \rightarrow \mathbb{U}$ for all $t \in [0, l)$) such that selecting the controls $\mathbf{u}_t = \boldsymbol{\pi}_t(\mathbf{x}_t)$ minimizes Eq. (4) subject to the stochastic discrete-time dynamics. This problem is addressed in Sec. 4.

For robotic systems with imperfect (e.g., partial and noisy) sensing, it is often beneficial during planning to explicitly consider the sensing uncertainty. We assume sensors provide data according to a stochastic observation model:

$$\mathbf{z}_t = \mathbf{h}(\mathbf{x}_t) + \mathbf{n}_t, \quad \mathbf{n}_t \sim \mathcal{N}(\mathbf{0}, V(\mathbf{x}_t)), \tag{5}$$

where $\mathbf{z}_t$ is the sensor measurement at step t and the noise is state-dependent and drawn from a given Gaussian distribution. We formulate this motion planning problem as a POMDP by defining the belief state $\mathbf{b}_t \in \mathbb{B}$, which is the distribution of the state $\mathbf{x}_t$ given all past controls and sensor measurements. We approximate belief states using Gaussian distributions. In belief space we define the cost function as

$$\mathbb{E}_{\mathbf{z}} \left[c_l(\mathbf{b}_l) + \sum_{t=0}^{l-1} c_t(\mathbf{b}_t, \mathbf{u}_t) \right], \tag{6}$$

where the local cost functions are defined analogously to Eq. (4). The objective for this problem is to compute a control policy $\boldsymbol{\pi}$ (defined by $\boldsymbol{\pi}_t : \mathbb{B} \rightarrow \mathbb{U}$ for all $t \in [0, l)$) in order to minimize Eq. (6) subject to the stochastic discrete-time dynamics. This problem is addressed in Sec. 5.

4 Stochastic Extended LQR

SELQR explicitly considers a system's stochastic nature in the planning phase and computes a nominal trajectory and an associated linear control policy that consider the impact of uncertainty. With the control policy from SELQR, the robot then executes the plan in a closed-loop fashion with sensor feedback. As in related methods such as iLQG [12], SELQR approximates the value functions quadratically by linearizing the dynamics and quadratizing the cost functions. But, as we will show, SELQR uses a novel approach to compute promising candidate trajectories around which to linearize the dynamics and quadratize the cost functions, enabling faster performance.

4.1 Method Overview

To consider non-linear dynamics and any cost function with positive (semi)definite Hessians, SELQR uses an iterative approach that linearizes the (stochastic) dynamics and locally quadratizes the cost functions in each iteration. As shown in Algorithm 1 and described below, each iteration includes both a forward pass and a backward pass, where each pass performs value iteration.

As in LQR, SELQR uses backward value iteration to compute a control policy $\boldsymbol{\pi}$ and, for all t, the cost-to-go $v_t(\mathbf{x})$, which is the minimum expected future cost that will be accrued between time step t (including the cost at time step t) and time step l if the robot starts at $\mathbf{x}$ at time step t. The backward value iteration, as described in Sec. 4.2, considers stochastic dynamics. SELQR also uses forward

Algorithm 1: SELQR

Input: stochastic continuous-time dynamics (Eq. (1)); c_t: local cost functions for $0 \leq t \leq l$; Δ: time step duration; l: number of time steps

Variables: $\hat{\mathbf{x}}$: smoothed states; $\boldsymbol{\pi}$: control policy; $\bar{\boldsymbol{\pi}}$: inverse control policy; v_t: cost-to-go function; $\bar{v}_t$: cost-to-come function

$\boldsymbol{\pi}_t = 0$, $S_t = 0$, $\mathbf{s}_t = \mathbf{0}$, $s_t = 0$
repeat
  $\bar{S}_0 := 0$, $\bar{\mathbf{s}}_0 := 0$, $\bar{s}_0 := 0$
  for $t := 0$; $t < l$; $t := t + 1$ **do**
    $\hat{\mathbf{x}}_t = -(S_t + \bar{S}_t)^{-1}(\mathbf{s}_t + \bar{\mathbf{s}}_t)$ (*smoothed states*)
    $\hat{\mathbf{u}}_t = \boldsymbol{\pi}_t(\hat{\mathbf{x}}_t)$, $\hat{\mathbf{x}}_{t+1} = \mathbf{g}(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t)$
    Linearize inverse discrete dynamics around $(\hat{\mathbf{x}}_{t+1}, \hat{\mathbf{u}}_t)$ (Eq. (16))
    Quadratize c_t around $(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t)$ (Eq. (12))
    Compute $\bar{S}_{t+1}$, $\bar{\mathbf{s}}_{t+1}$, $\bar{s}_{t+1}$, $\bar{v}_{t+1}$, $\bar{\boldsymbol{\pi}}_t$ (*forward value iteration in Sec. 4.3*)
  end
  Quadratize c_l around $\hat{\mathbf{x}}_l$ in the form of Eq. (12) to compute Q_l, $\mathbf{q}_l$, and q_l
  $S_l := Q_l$, $\mathbf{s}_l := \mathbf{q}_l$, and $s_l := q_l$.
  for $t := l - 1$; $t \geq 0$; $t := t - 1$ **do**
    $\hat{\mathbf{x}}_{t+1} = -(S_{t+1} + \bar{S}_{t+1})^{-1}(\mathbf{s}_{t+1} + \bar{\mathbf{s}}_{t+1})$ (*smoothed states*)
    $\hat{\mathbf{u}}_t = \bar{\boldsymbol{\pi}}_t(\hat{\mathbf{x}}_{t+1})$, $\hat{\mathbf{x}}_t = \bar{\mathbf{g}}(\hat{\mathbf{x}}_{t+1}, \hat{\mathbf{u}}_t)$
    Linearize stochastic discrete dynamics around $(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t)$ (Eq. (11))
    Quadratize c_t around $(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t)$ (Eq. (12))
    Compute S_t, $\mathbf{s}_t$, s_t, v_t, $\boldsymbol{\pi}_t$ (*backward value iteration in Sec. 4.2*)
  end
until *Converged (e.g., v_0 stops changing significantly)*;
return $\boldsymbol{\pi}_t$ *for* $0 \leq t \leq l$

value iteration to compute the cost-to-come $\bar{v}_t(\mathbf{x})$, which computes the minimum past cost that was accrued from time step 0 to step t (excluding the cost at time step t) assuming the robot's dynamics is deterministic, as described in Sec. 4.3. The sum of $v_t(\mathbf{x})$ and $\bar{v}_t(\mathbf{x})$ provides an estimate of $\hat{v}_t(\mathbf{x})$, the minimum expected total cost for the entire task execution given that the robot passes through state $\mathbf{x}$ at step t. Selecting $\mathbf{x}$ to minimize $\hat{v}_t$ yields a sequence of *smoothed states*

$$\hat{\mathbf{x}}_t = \text{argmin}_{\mathbf{x}} \hat{v}_t(\mathbf{x}) = \text{argmin}_{\mathbf{x}}(\bar{v}_t(\mathbf{x}) + v_t(\mathbf{x})), \quad 0 \leq t \leq l. \tag{7}$$

At each iteration, SELQR linearizes the (stochastic) dynamics and quadratizes the cost function around the smoothed states. With each iteration, SELQR progressively improves the estimate of the cost-to-come and cost-to-go at each state along a plan, and hence improves its estimate of the minimum expected total cost. With this improved estimate comes a better control policy. The algorithm terminates when the estimated total cost converges. The output of the motion planner is the control policy $\boldsymbol{\pi}_t$ for all t, where each $\boldsymbol{\pi}_t$ is computed during the backward value iteration, which considers the stochastic dynamics. During execution, a robot at state $\mathbf{x}$ executes control $\mathbf{u}_t = \boldsymbol{\pi}_t(\mathbf{x})$ at time step t.

SELQR accounts for non-linear dynamics and non-quadratic cost functions in a manner inspired in part by the iterated Kalman Smoother [5], which iteratively

performs a forward pass (filtering) and a backward pass (smoothing) and at each iteration linearizes the non-linear system around the states from the smoothing pass. Likewise, SELQR consists of a backward pass (a backward value iteration) and a forward pass (a forward value iteration). The combination of these two passes at each iteration enables us to compute smoothed states around which we linearize the (stochastic) dynamics and quadratize the cost functions.

4.2 Backward Pass

We assume the cost-to-come functions $\bar{v}_t(\mathbf{x})$, the inverse control policy $\bar{\boldsymbol{\pi}}_t$, and the smoothed state $\hat{\mathbf{x}}_l$ are available from the previous forward pass. The backward pass computes cost-to-go functions $v_t(\mathbf{x})$ and control policy $\boldsymbol{\pi}_t$, using the approach of backward value iteration [22] in a backward recursive manner:

$$v_\ell(\mathbf{x}) = c_\ell(\mathbf{x}), \quad v_t(\mathbf{x}) = \min_{\mathbf{u}} \left(c_t(\mathbf{x},\mathbf{u}) + \mathbb{E}_{\boldsymbol{\xi}_t}[v_{t+1}(\mathbf{g}_t(\mathbf{x},\mathbf{u}) + M_t(\mathbf{x},\mathbf{u})\boldsymbol{\xi}_t)] \right), \quad (8)$$

$$\boldsymbol{\pi}_t(\mathbf{x}) = \arg\min_{\mathbf{u}} \left(c_t(\mathbf{x},\mathbf{u}) + \mathbb{E}_{\boldsymbol{\xi}_t}[v_{t+1}(\mathbf{g}_t(\mathbf{x},\mathbf{u}) + M_t(\mathbf{x},\mathbf{u})\boldsymbol{\xi}_t)] \right).$$

To make the backward value iteration tractable, SELQR linearizes the stochastic dynamics and quadratizes the local cost functions to maintain a quadratic form of the cost-to-go function $v_t(\mathbf{x})$: $v_t(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T S_t \mathbf{x} + \mathbf{x}^T \mathbf{s}_t + s_t$. The backward pass starts from step l by quadratizing $c_l(\mathbf{x})$ around $\hat{\mathbf{x}}_l$ (line 11) as

$$c_l(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^T Q_l \mathbf{x} + \mathbf{x}^T \mathbf{q}_l + q_l, \quad (9)$$

and constructing quadratic $v_l(\mathbf{x})$ by setting $S_l = Q_l$, $\mathbf{s}_l = \mathbf{q}_l$, and $s_l = q_l$. Starting from $t = l-1$, $v_{t+1}(\mathbf{x})$ is available. To proceed to step t, SELQR first computes

$$\hat{v}_{t+1}(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^T(S_{t+1} + \bar{S}_{t+1})\mathbf{x} + \mathbf{x}^T(\mathbf{s}_{t+1} + \bar{\mathbf{s}}_{t+1}) + (s_{t+1} + \bar{s}_{t+1}). \quad (10)$$

Minimizing the quadratic $\hat{v}_{t+1}(\mathbf{x})$ with respect to $\mathbf{x}$ gives the smoothed states $\hat{\mathbf{x}}_{t+1}$ (line 14). With the inverse control policy $\bar{\boldsymbol{\pi}}_t$ from the last forward pass, SELQR computes $\hat{\mathbf{u}}_t$ and $\hat{\mathbf{x}}_t$ (line 15), around which the stochastic discrete dynamics can be linearized as

$$\mathbf{g}_t(\mathbf{x},\mathbf{u}) = A_t\mathbf{x} + B_t\mathbf{u} + \mathbf{a}_t, \quad M_t^{(i)}(\mathbf{x},\mathbf{u}) = F_t^i\mathbf{x} + G_t^i\mathbf{u} + \mathbf{e}_t^i, \quad 1 < i \leq n, \quad (11)$$

where $M_t^{(i)}$ denotes the i'th column of matrix M_t, and A_t, B_t, F_t^i, G_t^i, $\mathbf{a}_t$, and $\mathbf{e}_t^i$ are given matrices and vectors of the appropriate dimension, and the cost function c_t can be quadratized as

$$c_t(\mathbf{x},\mathbf{u}) = \frac{1}{2}\begin{bmatrix}\mathbf{x}\\ \mathbf{u}\end{bmatrix}^T \begin{bmatrix} Q_t & P_t^T \\ P_t & R_t \end{bmatrix} \begin{bmatrix}\mathbf{x}\\ \mathbf{u}\end{bmatrix} + \begin{bmatrix}\mathbf{x}\\ \mathbf{u}\end{bmatrix}^T \begin{bmatrix}\mathbf{q}_t\\ \mathbf{r}_t\end{bmatrix} + q_t. \quad (12)$$

By substituting the linear stochastic dynamics and quadratic local cost function into Eq. 8, expanding the expectation, and then collecting terms, we get a quadratic expression of the value function $v_t(\mathbf{x})$,

$$v_t(\mathbf{x}) = \min_{\mathbf{u}} \left(\frac{1}{2}\begin{bmatrix}\mathbf{x}\\ \mathbf{u}\end{bmatrix}^T \begin{bmatrix} C_t & E_t^T \\ E_t & D_t \end{bmatrix} \begin{bmatrix}\mathbf{x}\\ \mathbf{u}\end{bmatrix} + \begin{bmatrix}\mathbf{x}\\ \mathbf{u}\end{bmatrix}^T \begin{bmatrix}\mathbf{c}_t\\ \mathbf{d}_t\end{bmatrix} + e_t \right), \quad (13)$$

where C_t, D_t, E_t, $\mathbf{c}_t$, $\mathbf{d}_t$, e_t are parameterized by S_{t+1}, $\mathbf{s}_{t+1}$, s_{t+1}, Q_t, $\mathbf{q}_t$, q_t, P_t, R_t, $\mathbf{r}_t$, A_t, B_t, $\mathbf{a}_t$, F_t^i, G_t^i, and e_t^i following the similar derivation in [12]. Minimizing Eq. (13) with respect to $\mathbf{u}$ gives the linear control policy:

$$\mathbf{u} = \boldsymbol{\pi}_t(\mathbf{x}) = -D_t^{-1}E_t\mathbf{x} - D_t^{-1}\mathbf{d}_t. \quad (14)$$

Filling $\mathbf{u}$ back into Eq. (13) gives $v_t(\mathbf{x})$ as a quadratic function of $\mathbf{x}$ with $S_t = C_t - E_t^T D_t^{-1} E_t$, $\mathbf{s}_t = \mathbf{c}_t - E_t^T D_t^{-1}\mathbf{d}_t$, and $s_t = e_t - \frac{1}{2}\mathbf{d}_t^T D_t^{-1}\mathbf{d}_t$ (line 18).

4.3 Forward Pass

The forward pass recursively computes the cost-to-come functions $\bar{v}_t(\mathbf{x})$ and the inverse control policy $\bar{\boldsymbol{\pi}}_t$ using *forward value iteration* [9]:

$$\bar{v}_0(\mathbf{x}) = 0, \quad \bar{v}_{t+1}(\mathbf{x}) = \min_{\mathbf{u}}(c_t(\bar{\mathbf{g}}_t(\mathbf{x},\mathbf{u}),\mathbf{u}) + \bar{v}_t(\bar{\mathbf{g}}_t(\mathbf{x},\mathbf{u}))), \tag{15}$$
$$\bar{\boldsymbol{\pi}}_t(\mathbf{x}) = \arg\min_{\mathbf{u}}(c_t(\bar{\mathbf{g}}_t(\mathbf{x},\mathbf{u}),\mathbf{u}) + \bar{v}_t(\bar{\mathbf{g}}_t(\mathbf{x},\mathbf{u}))).$$

To make the forward value iteration tractable, we linearize the inverse dynamics and quadratize the local cost functions so that we can maintain a quadratic form of the cost-to-come function $\bar{v}_t(\mathbf{x})$: $\bar{v}_t(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\bar{S}_t\mathbf{x} + \mathbf{x}^T\bar{\mathbf{s}}_t + \bar{s}_t$.

The forward pass starts from time step 0 (line 3) to construct the quadratic $\bar{v}_0(\mathbf{x})$ by setting $\bar{S}_0 = 0$, $\bar{\mathbf{s}}_0 = \mathbf{0}$, and $\bar{s}_0 = 0$. At time step t, we assume $\bar{v}_t(\mathbf{x})$ and $v_t(\mathbf{x})$ are available. To proceed to step $t+1$, SELQR first computes the smoothed state $\hat{\mathbf{x}}_t$ by minimizing the sum of $v_t(\mathbf{x})$ and $\bar{v}_t(\mathbf{x})$ (line 5) which are quadratic. Since $\boldsymbol{\pi}_t$ is available, SELQR then computes the $\hat{\mathbf{u}}_t$ and $\hat{\mathbf{x}}_{t+1}$ as shown in line 7. Then, the deterministic inverse discrete dynamics is linearized around $(\hat{\mathbf{x}}_{t+1}, \hat{\mathbf{u}}_t)$:

$$\bar{\mathbf{g}}_t(\mathbf{x},\mathbf{u}) = \bar{A}_t\mathbf{x} + \bar{B}_t\mathbf{u} + \bar{\mathbf{a}}_t, \tag{16}$$

where $\bar{A}_t$, $\bar{B}_t$, and $\bar{\mathbf{a}}_t$ are given matrices and vectors of the appropriate dimension, and the local cost function c_t is quadratized around $(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t)$ to get the quadratic form as in Eq. (12).

Substituting the linearized inverse dynamics and quadratic local cost function into Eq. (15), expanding the expectation, and then collecting terms, we get a quadratic expression for $\bar{v}_{t+1}(\mathbf{x})$,

$$\bar{v}_{t+1}(\mathbf{x}) = \min_{\mathbf{u}}\left(\frac{1}{2}\begin{bmatrix}\mathbf{x}\\ \mathbf{u}\end{bmatrix}^T\begin{bmatrix}\bar{C}_t & \bar{E}_t^T\\ \bar{E}_t & \bar{D}_t\end{bmatrix}\begin{bmatrix}\mathbf{x}\\ \mathbf{u}\end{bmatrix} + \begin{bmatrix}\mathbf{x}\\ \mathbf{u}\end{bmatrix}^T\begin{bmatrix}\bar{\mathbf{c}}_t\\ \bar{\mathbf{d}}_t\end{bmatrix} + \bar{e}_t\right), \tag{17}$$

where $\bar{C}_t, \bar{D}_t, \bar{E}_t, \bar{\mathbf{c}}_t, \bar{\mathbf{d}}_t, \bar{e}_t$ are computed from $\bar{S}_t$, $\bar{\mathbf{s}}_t$, $\bar{s}_t$, $\bar{A}_t$, $\bar{B}_t$, $\bar{\mathbf{a}}_t$, Q_t, $\mathbf{q}_t$, q_t, P_t, R_t, and $\mathbf{r}_t$ following the derivation in [9]. The corresponding linear inverse control policy that minimizes Eq. (17) has the form

$$\mathbf{u}_t = \bar{\boldsymbol{\pi}}_t(\mathbf{x}_{t+1}) = -\bar{D}_t^{-1}\bar{E}_t\mathbf{x}_{t+1} - \bar{D}_t^{-1}\bar{\mathbf{d}}_t. \tag{18}$$

Plugging $\mathbf{u}_t$ into Eq. (17) gives $\bar{v}_{t+1}(\mathbf{x})$ as a quadratic function of $\mathbf{x}$ with $\bar{S}_{t+1} = \bar{C}_t - \bar{E}_t^T\bar{D}_t^{-1}\bar{E}_t$, $\bar{\mathbf{s}}_{t+1} = \bar{\mathbf{c}}_t - \bar{E}_t^T\bar{D}_t^{-1}\bar{\mathbf{d}}_t$, and $\bar{s}_{t+1} = \bar{e}_t - \frac{1}{2}\bar{\mathbf{d}}_t^T\bar{D}_t^{-1}\bar{\mathbf{d}}_t$ (line 9).

4.4 Iterative Forward and Backward Value Iteration

Without any *a priori* knowledge, SELQR initializes the cost-to-go functions and the control policy to 0's (line 1). As shown in Algorithm 1, SELQR starts with a forward pass and then iteratively performs backward passes and forward passes until convergence (e.g., v_0 stops changing significantly). Similar to the iterated Kalman Smoother and to Extended LQR [9], SELQR performs Gauss-Newton like updates toward a local optimum.

Informed search methods often achieve speedups in practice by exploring from states that minimize a heuristic cost function. Analogously, in SELQR, the cost-to-go provides the minimum expected future cost, and the cost-to-come estimates the minimum expected cost that has been already accrued. The forward

value iteration uses a deterministic inverse dynamics due to the intractability of computing a stochastic discrete inverse dynamics. Hence, the function $\hat{v}_t(\mathbf{x})$ estimates the minimum total cost assuming the robot passes through a given state $\mathbf{x}$ at time step t. Previous methods such as iLQG choose states for linearization and quadratization by blindly shooting the control policy from the last iteration without any information about the cost functions. These methods usually need measures such as line search to maintain stability. By computing smoothed states that are informed by cost for linearization and quadratization, we show, experimentally, that our method provides faster convergence.

4.5 Discrete-Time Dynamics Implementation

If $\mathbf{f}(\mathbf{x}, \mathbf{u}, \tau)$ in Eq. (1) is linear in $\mathbf{x}$ and N is not dependent on $\mathbf{x}$, then the distribution of the state at any time τ is given by $\mathbf{x}(\tau) \sim \mathcal{N}(\hat{\mathbf{x}}(\tau), \Sigma(\tau))$, where $\hat{\mathbf{x}}(\tau)$ and $\Sigma(\tau)$ are defined by the following system of differential equations:

$$\dot{\hat{\mathbf{x}}} = \mathbf{f}(\hat{\mathbf{x}}, \mathbf{u}, \tau), \quad \dot{\Sigma} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\hat{\mathbf{x}}, \mathbf{u}, \tau)\Sigma + \Sigma \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\hat{\mathbf{x}}, \mathbf{u}, \tau)^T + N(\hat{\mathbf{x}}, \mathbf{u}, \tau)N(\hat{\mathbf{x}}, \mathbf{u}, \tau)^T.$$

For non-linear $\mathbf{f}$ and state- and control-dependent N, the equations provide first-order approximations. Instead of using an Euler integration [12], we use the Runge-Kutta method (RK4) to integrate the differential equations for $\hat{\mathbf{x}}$ and Σ forward in time simultaneously to compute $\mathbf{g}_t$ and M_t in Eq. (2), and integrate the differential equation for $\hat{\mathbf{x}}$ to compute the $\bar{\mathbf{g}}_t$ in Eq. (3).

5 Stochastic Extended LQR in Belief Space

We introduce B-SELQR, a belief-state variant of SELQR for robotic systems with both motion and sensing uncertainty, where beliefs are modeled with Gaussian distributions. With an imperfect sensing model defined in the form of Eq. (5) and an objective function in the form of Eq. (6), the motion planning problem is a POMDP. B-SELQR needs a stochastic discrete forward belief dynamics and a deterministic discrete inverse belief dynamics. While the stochastic belief dynamics (Sec. 5.1) can be modeled by an Extended Kalman Filter (EKF) [23] as shown in [20], the key challenge here is to develop the deterministic discrete inverse belief dynamics. We will show in Sec. 5.2 that the inverse belief dynamics can be derived by inverting the EKF.

5.1 Stochastic Discrete Belief Dynamics

Let us be given the belief of the robot's state at time step t as $\mathbf{x}_t \sim \mathcal{N}(\hat{\mathbf{x}}_t, \Sigma_t)$ and a control input $\mathbf{u}_t$ that the robot will execute at time step t. The EKF is used to model the stochastic forward belief dynamics [20] by

$$\begin{aligned} \hat{\mathbf{x}}_{t+1} &= \mathbf{g}(\hat{\mathbf{x}}_t, \mathbf{u}_t) + \mathbf{w}_t, \quad \mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, K_t H_t \Gamma_{t+1}) \\ \Sigma_{t+1} &= \Gamma_{t+1} - K_t H_t \Gamma_{t+1}, \end{aligned} \tag{19}$$

where

$$\Gamma_{t+1} = A_t \Sigma_t A_t^T + M_t(\hat{\mathbf{x}}_t, \mathbf{u}_t) M_t(\hat{\mathbf{x}}_t, \mathbf{u}_t)^T, \quad A_t = \frac{\partial \mathbf{g}}{\partial \mathbf{x}}(\hat{\mathbf{x}}_t, \mathbf{u}_t),$$

$$K_t = \Gamma_{t+1} H_t^T (H_t \Gamma_{t+1} H_t^T + V(\hat{\mathbf{x}}'_{t+1}))^{-1}, \quad H_t = \frac{\partial \mathbf{h}}{\partial \mathbf{x}}(\mathbf{g}(\hat{\mathbf{x}}_t, \mathbf{u}_t)).$$

We refer readers to [20] for details of the derivation. Defining the belief $\mathbf{b}_t = \left[\hat{\mathbf{x}}_t^T, \text{vec}[\sqrt{\Sigma_t}]^T\right]^T$, the stochastic belief dynamics is given by

$$\mathbf{b}_{t+1} = \Phi(\mathbf{b}_t, \mathbf{u}_t) + W(\mathbf{b}_t, \mathbf{u}_t)\boldsymbol{\xi}_t, \quad \boldsymbol{\xi}_t \sim \mathcal{N}(0, I), \tag{20}$$

where $W(\mathbf{b}_t, \mathbf{u}_t) = \left[\sqrt{K_t H_t \Gamma_{t+1}}^T, 0\right]^T$ and vec$[Z]$ returns a vector consisting of all the columns of matrix Z. The dynamics is stochastic since the observation is treated as a random variable.

5.2 Deterministic Inverse Discrete Belief Dynamics

To derive a deterministic inverse belief dynamics, we use the maximum likelihood observation assumption as introduced in [21].

Proposition 1. *(Deterministic Inverse Discrete Belief Dynamics) We assume an EKF with the maximum likelihood observation assumption is used to propagate the beliefs forward in time. Given* $\mathbf{b}_{t+1} = \left[\hat{\mathbf{x}}_{t+1}^T, \text{vec}[\sqrt{\Sigma_{t+1}}]^T\right]^T$ *and the control input* $\mathbf{u}_t$ *applied at time step* t*, there exists a belief* $\mathbf{b}_t = \left[\hat{\mathbf{x}}_t^T, \text{vec}[\sqrt{\Sigma_t}]^T\right]^T$ *such that* $\mathbf{b}_{t+1} = \Phi(\mathbf{b}_t, \mathbf{u}_t)$ *and* $\mathbf{b}_t$ *is represented by*

$$\hat{\mathbf{x}}_t = \bar{\mathbf{g}}(\hat{\mathbf{x}}_{t+1}, \mathbf{u}_t), \tag{21}$$

$$\Sigma_t = \bar{A}_t^{-1}(\bar{\Gamma}_t - \bar{M}_t \bar{M}_t^T)\bar{A}_t^{-T}, \tag{22}$$

where

$$\bar{M}_t = M_t(\bar{\mathbf{g}}(\hat{\mathbf{x}}_{t+1}, \mathbf{u}_t), \mathbf{u}_t), \quad \bar{A}_t = \frac{\partial \mathbf{g}}{\partial \mathbf{x}}(\bar{\mathbf{g}}(\hat{\mathbf{x}}_{t+1}, \mathbf{u}_t), \mathbf{u}_t),$$

$$\bar{\Gamma}_t = (I - \Sigma_{t+1}\bar{H}_t^T \bar{V}_t^{-1}\bar{H}_t)^{-1}\Sigma_{t+1}, \quad \bar{H}_t = \frac{\partial \mathbf{h}}{\partial \mathbf{x}}(\hat{\mathbf{x}}_{t+1}), \quad \bar{V}_t = V(\hat{\mathbf{x}}_{t+1}). \tag{23}$$

Proof. Let us assume $\mathbf{x}_{t+1} \sim \mathcal{N}(\hat{\mathbf{x}}'_{t+1}, \Sigma'_{t+1})$ is the prior belief obtained from the process update of the EKF by evolving the system dynamics from time step t to $t+1$ before any observation is received. With the prior belief, let us assume an observation $\mathbf{z}_{t+1}$ is received, and then the EKF updates the belief as follows:

$$\hat{\mathbf{x}}_{t+1} = \hat{\mathbf{x}}'_{t+1} + \bar{K}_t(\mathbf{z}_{t+1} - \mathbf{h}(\hat{\mathbf{x}}'_{t+1})), \quad \Sigma_{t+1} = \Sigma'_{t+1} - \bar{K}_t \tilde{H}_t \Sigma'_{t+1}, \tag{24}$$

where $\tilde{H}_t = \frac{\partial \mathbf{h}}{\partial \mathbf{x}}(\hat{\mathbf{x}}'_{t+1})$ and

$$\bar{K}_t = \Sigma'_{t+1}\tilde{H}_t^T(\tilde{H}_t \Sigma'_{t+1}\tilde{H}_t^T + V(\hat{\mathbf{x}}'_{t+1}))^{-1}. \tag{25}$$

The maximum likelihood observation assumption means $\mathbf{z}_{t+1} = \mathbf{h}(\hat{\mathbf{x}}'_{t+1})$. Hence we see $\hat{\mathbf{x}}_{t+1} = \hat{\mathbf{x}}'_{t+1}$ from Eq. (24). Due to this equivalence we can see that $\bar{H}_t = \tilde{H}_t$ and $\bar{V}_t = V(\bar{\mathbf{x}}'_{t+1})$ ($\bar{H}_t$ and $\bar{V}_t$ are defined in Eqs. (23)). Hence, Eq. (25) can be re-written using $\bar{H}_t$ and $\bar{V}_t$ as

$$\bar{K}_t = \Sigma'_{t+1}\bar{H}_t^T(\bar{H}_t \Sigma'_{t+1}\bar{H}_t^T + \bar{V}_t)^{-1}. \tag{26}$$

By right multiplying $(\bar{H}_t \Sigma'_{t+1}\bar{H}_t^T + \bar{V}_t)$ on both sides of the above equation and then subtracting the term $\bar{K}_t \bar{H}_t \Sigma'_{t+1}\bar{H}_t^T$ on both sides, we get

$$\bar{K}_t \bar{V}_t = (\Sigma'_{t+1} - \bar{K}_t \bar{H}_t \Sigma'_{t+1})\bar{H}_t^T. \tag{27}$$

By substituting Σ_{t+1} from Eq. (24) into the above equation and then right multiplying $\bar{V}_t^{-1}$ on both sides, we get the expression for $\bar{K}_t$,

$$\bar{K}_t = \Sigma_{t+1}\bar{H}_t^T\bar{V}_t^{-1}. \tag{28}$$

Then, we substitute Eq. (28) back into Eq. (24) and then solve for Σ'_{t+1},

$$\Sigma'_{t+1} = (I - \Sigma_{t+1}\bar{H}_t^T\bar{V}_t^{-1})^{-1}\Sigma_{t+1}. \tag{29}$$

The process update of EKF can be modeled as

$$\hat{\mathbf{x}}'_{t+1} = \mathbf{g}(\hat{\mathbf{x}}_t, \mathbf{u}_t), \;\; \Sigma'_{t+1} = \frac{\partial \mathbf{g}}{\partial \mathbf{x}}(\hat{\mathbf{x}}_t, \mathbf{u}_t)\Sigma_t\frac{\partial \mathbf{g}}{\partial \mathbf{x}}(\hat{\mathbf{x}}_t, \mathbf{u}_t)^T + M_t(\hat{\mathbf{x}}_t, \mathbf{u}_t)M_t(\hat{\mathbf{x}}_t, \mathbf{u}_t)^T. \tag{30}$$

Since $\hat{\mathbf{x}}'_{t+1} = \mathbf{g}(\hat{\mathbf{x}}_t, \mathbf{u}_t)$ and $\hat{\mathbf{x}}'_{t+1} = \hat{\mathbf{x}}_{t+1}$, we see $\hat{\mathbf{x}}_t = \bar{\mathbf{g}}(\hat{\mathbf{x}}'_{t+1}, \mathbf{u}_t) = \bar{\mathbf{g}}(\hat{\mathbf{x}}_{t+1}, \mathbf{u}_t)$. Hence, we prove Eq. (21).

Substituting $\hat{\mathbf{x}}_t = \bar{\mathbf{g}}(\hat{\mathbf{x}}_{t+1}, \mathbf{u}_t)$ into Eq. (30), we get $\Sigma'_{t+1} = \bar{A}_t\Sigma_t\bar{A}_t^T + \bar{M}_t\bar{M}_t^T$, where $\bar{A}_t$ and $\bar{M}_t$ are defined in Eqs. (23). We then solve for Σ_t and get

$$\Sigma_t = \bar{A}_t^{-1}(\Sigma'_{t+1} - \bar{M}_t\bar{M}_t^T)\bar{A}_t^{-T}. \tag{31}$$

By substituting Eq. (29) into Eq. (31), we prove Eq. (22). □

Eqs. (21) and (22) model the deterministic discrete inverse belief dynamics, which we write as $\mathbf{b}_t = \bar{\Phi}(\mathbf{b}_{t+1}, \mathbf{u}_t)$. One can show that $\mathbf{b}_{t+1} = \Phi(\bar{\Phi}(\mathbf{b}_{t+1}, \mathbf{u}_t), \mathbf{u}_t)$. With the stochastic discrete forward belief dynamics and deterministic inverse belief dynamics, together with cost objective Eq. (6) defined over belief space, we can directly apply SELQR to planning in belief space.

6 Experiments

We demonstrate SELQR in simulation for a car-like robot, a quadrotor, and a medical steerable needle. Each robot must navigate in an environment with obstacles. We also apply B-SELQR to a car-like robot. We implemented the methods in C++ and ran scenarios on a PC with an Intel i3 2.4 GHz processor.

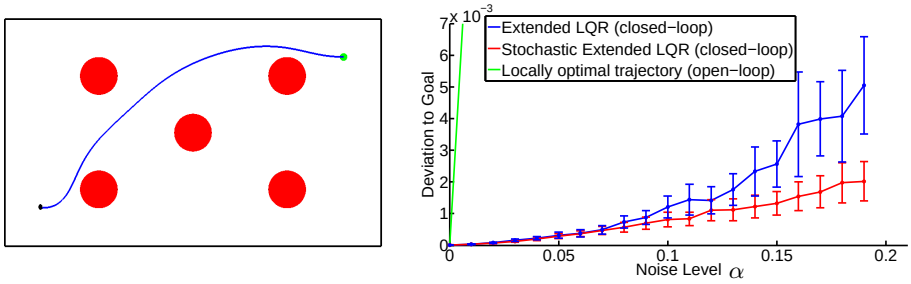
In our experiments, we used the local cost functions

$$c_0(\mathbf{x}, \mathbf{u}) = \tfrac{1}{2}(\mathbf{x} - \mathbf{x}_0^*)^T Q_0(\mathbf{x} - \mathbf{x}_0^*) + \tfrac{1}{2}(\mathbf{u} - \mathbf{u}^*)^T R(\mathbf{u} - \mathbf{u}^*),$$
$$c_t(\mathbf{x}, \mathbf{u}) = \tfrac{1}{2}(\mathbf{u} - \mathbf{u}^*)^T R(\mathbf{u} - \mathbf{u}^*) + f(\mathbf{x}), \;\; c_l(\mathbf{x}, \mathbf{u}) = \tfrac{1}{2}(\mathbf{x} - \mathbf{x}_l^*)^T Q_l(\mathbf{x} - \mathbf{x}_l^*).$$

where Q_0, Q_l, and R are positive definite. We set $\mathbf{x}_0^*$ to be a given initial state and $\mathbf{x}_l^*$ to be a given goal state. Setting Q_0 and Q_l infinitely large equates to fixing the initial state and goal state for planning. We set function $f(\mathbf{x})$ to enforce obstacle avoidance. For SELQR we used the same cost term as in [9]:

$$f(\mathbf{x}) = q\sum_i \exp(-d_i(\mathbf{x})), \tag{32}$$

where $q \in \mathbb{R}^+$ and $d_i(\mathbf{x})$ is the signed distance between the robot at state $\mathbf{x}$ and the i'th obstacle. Since the Hessian of $f(\mathbf{x})$ is not always positive semidefinite, we regularize the Hessian by computing its eigendecomposition and setting the negative eigenvalues to zeros [9]. We assume each obstacle is convex. For non-convex obstacles, we apply convex decomposition. For B-SELQR, to approximately consider the probability of collision we set $f(\mathbf{b}) = q\sum_i \exp(-d_i(\mathbf{b}))$, where $d_i(\mathbf{b})$ is the minimum number of standard deviations of the mean of the robot's belief distribution needed to move to the obstacle's surface [20].



(a) SELQR trajectory (b) Impact of Noise Level

Fig. 2. (a) The SELQR trajectory for a car-like robot moving to a green goal while avoiding red obstacles. (b) Mean and standard deviations for the deviation from the goal over 1,000 simulations for SELQR and related methods with different noise levels.

6.1 Car-like Robot in a 2-D Environment

We first apply SELQR to a non-holonomic car-like robot that navigates in a 2-D environment and can perfectly sense its state. The robot's state $\mathbf{x} = [x, y, \theta, v]$ consists of its position (x, y), orientation θ, and speed v. The control inputs $\mathbf{u} = [a, \phi]$ consist of acceleration a and steering wheel angle ϕ. The deterministic continuous dynamics is given by

$$\dot{x} = v\cos(\theta), \quad \dot{y} = v\sin(\theta), \quad \dot{\theta} = v\tan(\phi)/d, \quad \dot{v} = a, \tag{33}$$

where d is the length of the car-like robot. We assume the dynamics is corrupted by noise from a Weiner process (Eq. 1) and define $N(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau) = \alpha\|\mathbf{u}(\tau)\|$, $\alpha \in \mathbb{R}^+$. For the cost function we set $Q_0 = Q_l = 200I$, $R = 1.0I$, and $q = 0.2$.

Fig. 2(a) shows the environment and the SELQR trajectory (illustrated by the path that results from following the control policy computed by SELQR assuming zero noise). Consideration of stochastic dynamics is important for good performance. Fig. 2(b) shows the deviation from the goal for varying levels of noise α. We compare with Extended LQR, which uses deterministic dynamics to compute the control policy, and with open-loop execution of SELQR's nominal trajectory, which performs poorly due to the motion uncertainty and need for feedback. The control policies from SELQR result in a smaller deviation from the goal since SELQR explicitly considers the control-dependent noise.

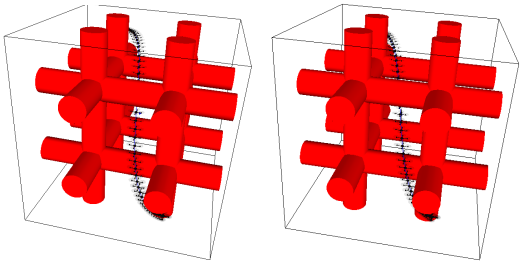
In Table 1, we show SELQR's fast convergence for different values of Δ. The results are averages of 100 independent runs for random instances. In each instance, the initial state $\mathbf{x}_0^*$ was chosen by uniformly sampling in the workspace, and the corresponding goal state was $\mathbf{x}_l^* = -\mathbf{x}_0^*$ (where the origin is the center of the workspace). Compared to iLQG, our method achieved approximately equal costs but required substantially fewer iterations and less computation time.

6.2 Quadrotor in a 3-D Environment

To show that SELQR scales to higher dimensions, we apply it to a simulated quadrotor with a 12-D state space. Its state $\mathbf{x} = [\mathbf{p}, \mathbf{v}, \mathbf{r}, \mathbf{w}] \in \mathbb{R}^{12}$ consists of position $\mathbf{p}$, velocity $\mathbf{v}$, orientation $\mathbf{r}$ (angle-axis representation), and angular

Table 1. Quantitative Comparison of SELQR and iLQG.

Scenario	Δ (s)	SELQR			iLQG		
		Avg Cost	Avg Time (s)	Avg #Iters	Avg Cost	Avg Time (s)	Avg #Iters
Car-like robot	0.05	79.4	0.4	5.7	80.5	1.1	13.4
	0.1	55.5	1.0	16.0	53.4	2.5	43.2
	0.2	50.8	1.2	18.4	51.7	2.0	35.4
Quadrotor	0.025	552.1	30.3	7.7	798.0	52.7	23.4
	0.05	272.7	50.1	14.4	292.1	113.7	51.6
	0.1	191.1	66.3	20.0	197.1	163.9	76.4
Steerable needle	0.075	53.6	0.79	5.3	58.3	1.2	12.5
	0.1	42.6	0.95	6.36	44.5	1.4	14.6
	0.125	39.1	1.3	10.1	40.0	1.5	15.6



(a) $q = 1.0$ (b) $q = 0.3$

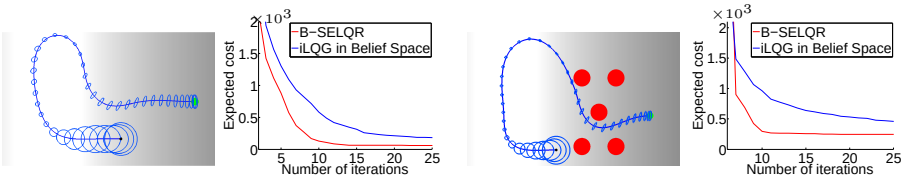
Fig. 3. SELQR trajectories for a quadrotor in an 8 cylindirical obstacle environment.

velocity $\mathbf{w}$. Its control input $\mathbf{u} = [u_1, u_2, u_3, u_4]$ consists of the forces exerted by each of the four rotors. We directly adopt the continuous dynamics $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ with physical parameters of the quadrotor and the environment from [9]. We add noise defined by $N(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau) = \alpha \|\mathbf{u}(\tau)\|$, where $\alpha \in \mathbb{R}^+$.

Fig. 3 shows the SELQR trajectory for two different values of q, where we set $\alpha = 2\%$, $Q_0 = Q_l = 500I$, and $R = 20I$. As expected, the trajectory with larger q has larger clearance from obstacles. In Table 1, we show SELQR's fast convergence for the quadrotor scenario for different values of Δ. We conducted randomized runs in a manner analagous to Sec. 6.1. For the quadrotor, compared to iLQG, our method achieved slightly better costs while requiring substantially fewer iterations and less computation time.

6.3 Medical Needle Steering for Liver Biopsy

We also demonstrate SELQR for steering a flexible bevel-tip needle through liver tissue while avoiding critical vasculature modeled by a trianglular mesh (Fig. 1). We use the stochastic needle model introduced in [24], where the kinematics are defined in $SE(3)$. We represent the state $\mathbf{x}$ by the tip's position $\mathbf{p}$ and orientation $\mathbf{r}$ (angle-axis). The control input is $\mathbf{u} = [v, w, \kappa]^T$, where v is the insertion speed, w is the axial rotation speed, and κ is the curvature, which can vary from 0 to a maximum curvature of κ_0 using duty-cycling. For the cost function, we set $\mathbf{u}^* = [0, 0, 0.5\kappa_0]^T$. Hence, we penalize large insertion speed, which given l and Δ corresponds to penalizing path length. It also penalizes curvatures that are



(a) B-SELQR for scenario w/o obstacles (b) B-SELQR for scenario w/ obstacles

Fig. 4. (a) B-SELQR trajectory for a car-like robot navigating to a goal (green) in a 2-D light-dark domain (adapted from [21]). (b) B-SELQR trajectory for the environment with obstacles (red circles). The blue ellipsoids show 3 standard deviations of the belief distributions. B-SELQR converges faster than iLQG in belief space in both scenarios.

too large (close to the kinematic limits of the device) or too small (requiring high-rate duty cycling, which may cause tissue damage).

Fig. 1 shows the SELQR trajectories for two insertion locations with $\Delta = 0.1$s, $l = 30$, $Q_0 = Q_l = 100I$, $R = I$, and $q = 0.5$. Table 1 shows SELQR's fast convergence for the steerable needle for varying Δ. The results are averages of 100 independent runs for random instances. In each instance, the goal state was held constant, and we set the initial state $\mathbf{x}_0^*$ such that the needle was inserted into the tissue from a uniformly-sampled point on the left (corresponding to the skin surface). Compared to iLQG, our method achieved approximately equal costs but required substantially fewer iterations and less computation time.

6.4 Belief Space Planning for a Car-like Robot

We apply B-SELQR to the car-like robot in Sec. 6.1 but now with added uncertainty in sensing. We consider the light-dark domain scenario suggested in [21]. The robot localizes itself using noisy measurements from sensors in the environment. The reliability of the measurement varies as a function of the robot's position. The robot receives reliable measurements in the bright region and noisier measurements in the darker regions. Formally, the observation model is

$$\mathbf{z}_t = \mathbf{x}_t + \mathbf{n}_t, \quad \mathbf{n}_t \sim \mathcal{N}(\mathbf{0}, ((x - x^*)^2 + 1)\beta I), \tag{34}$$

where $\beta \in \mathbb{R}^+$ is a given constant.

For belief space planning we use the cost functions

$$\begin{aligned} c_0(\mathbf{b}, \mathbf{u}) &= \tfrac{1}{2}(\mathbf{b} - \mathbf{b}_0^*)^T Q_0 (\mathbf{b} - \mathbf{b}_0^*) + \tfrac{1}{2}(\mathbf{u} - \mathbf{u}^*)^T R(\mathbf{u} - \mathbf{u}^*), \\ c_t(\mathbf{b}, \mathbf{u}) &= \tfrac{1}{2}\mathrm{tr}[\sqrt{\Sigma} Q_t \sqrt{\Sigma}] + \tfrac{1}{2}(\mathbf{u} - \mathbf{u}^*)^T R(\mathbf{u} - \mathbf{u}^*) + f(\mathbf{b}), \\ c_l(\mathbf{b}, \mathbf{u}) &= \tfrac{1}{2}(\hat{\mathbf{x}} - \mathbf{x}_l^*)^T Q_l (\hat{\mathbf{x}} - \mathbf{x}_l^*) + \mathrm{tr}[\sqrt{\Sigma} Q_l \sqrt{\Sigma}]. \end{aligned}$$

We set $Q_0 = 1000I$, $R = 2I$, $Q_t = 10I$, $Q_l = 500I$, $q = 0.1$, and $\beta = 0.1$.

Fig. 4 shows the B-SELQR trajectory and associated beliefs along the trajectory for a scenario with and without obstacles. The computed control policies steer the robot to the light region where the measurement noise is smallest in order to better localize the robot before proceeding to the goal. We also show the convergence of B-SELQR. We compare with iLQG executed for the same cost

functions in belief space using the method in [20]. The statistics were computed by averaging the results of 100 random instances. (For each random instance, we randomly sampled the initial state $\mathbf{x}_0^*$.) On average, B-SELQR requires fewer iterations to reach a desired solution quality.

7 Conclusion

We presented Stochastic Extended LQR (SELQR), a novel optimization-based motion planner that computes a trajectory and associated linear control policy with the objective of minimizing the expected value of a user-defined cost function. SELQR applies to robotic systems that have stochastic non-linear dynamics and state- and control-dependent motion uncertainty. We also extended SELQR to applications with imperfect sensing, requiring motion planning in belief space. Our approach converges faster and more reliably than related methods in both the robot's state space and belief space for multiple simulated scenarios, ranging from a mobile robot to a steerable needle.

In future work, we hope to broaden the applicability of the approach. The approach currently assumes motion and sensing uncertainty are modeled using Gaussian distributions. While this assumption is often appropriate, it is not valid for some problems. Our approach also relies on first and second order information, so to improve stability we plan to investigate the use of automatic differentiation. We also plan to apply the methods to physical robots like steerable needles in order to efficiently account for motion and sensing uncertainty.

Acknowledgments. This research was supported in part by the National Science Foundation (NSF) under awards IIS-1117127 and IIS-1149965 and by the National Institutes of Health (NIH) under award R21EB017952.

References

1. Zucker, M., Ratliff, N., Dragan, A.D., Pivtoraiko, M., Matthew, K., Dellin, C.M., Bagnell, J.A., Srinivasa, S.S.: CHOMP: Covariant Hamiltonian optimization for motion planning. Int. J. Robotics Research **32**(9) (August 2012) 1164–1193
2. Schulman, J., Ho, J., Lee, A., Awwal, I., Bradlow, H., Abbeel, P.: Finding locally optimal, collision-free trajectories with sequential convex optimization. In: Robotics: Science and Systems (RSS). (June 2013)
3. Kalakrishnan, M., Chitta, S., Theodorou, E., Pastor, P., Schaal, S.: STOMP: Stochastic trajectory optimization for motion planning. In: Proc. IEEE Int. Conf. Robotics and Automation (ICRA). (May 2011) 4569–4574
4. LaValle, S.M.: Planning Algorithms. Cambridge University Press, Cambridge, U.K. (2006)
5. Bell, B.M.: The iterated Kalman smoother as a Gauss-Newton method. SIAM J. Optimization **4**(3) (1994) 626–636
6. Brock, O., Khatib, O.: Elastic strips: A framework for motion generation in human environments. Int. J. Robotics Research **21**(2) (December 2002) 1031–1052
7. Hauser, K., Ng-Thow-Hing, V.: Fast smoothing of manipulator trajectories using optimal bounded-acceleration shortcuts. In: Proc. IEEE Int. Conf. Robotics and Automation (ICRA). (May 2010) 2493–2498

8. Pan, J., Zhang, L., Manocha, D.: Collision-free and smooth trajectory computation in cluttered environments. Int. J. Robotics Research **31**(10) (September 2012) 1155–1175
9. van den Berg, J.: Extended LQR: Locally-optimal feedback control for systems with non-linear dynamics and non-quadratic cost. In: Int. Symp. Robotics Research (ISRR). (December 2013)
10. van den Berg, J.: Iterated LQR smoothing for locally-optimal feedback control of systems with non-linear dynamics and non-quadratic cost. In: Proc. American Control Conference. (June 2014)
11. Toussaint, M.: Robot trajectory optimization using approximate inference. In: Proc. Int. Conf. Machine Learning (ICML). (2009)
12. Todorov, E.: A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems. Proc. American Control Conference (2005) 300–306
13. Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. Artificial Intelligence **101**(1-2) (1998) 99–134
14. Pineau, J., Gordon, G., Thrun, S.: Point-based value iteration: An anytime algorithm for POMDPs. Int. Joint Conf. Artificial Intelligence (IJCAI) (2003) 1025–1032
15. Kurniawati, H., Hsu, D., Lee, W.: SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In: Robotics: Science and Systems (RSS). (2008)
16. Bai, H., Hsu, D., Lee, W.S.: Integrated perception and planning in the continuous space: A POMDP approach. In: Robotics: Science and Systems (RSS). (2013)
17. Bry, A., Roy, N.: Rapidly-exploring random belief trees for motion planning under uncertainty. In: Proc. IEEE Int. Conf. Robotics and Automation (ICRA). (May 2011) 723–730
18. Prentice, S., Roy, N.: The belief roadmap: Efficient planning in belief space by factoring the covariance. Int. J. Robotics Research **28**(11) (November 2009) 1448–1465
19. van den Berg, J., Abbeel, P., Goldberg, K.: LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information. Int. J. Robotics Research **30**(7) (June 2011) 895–913
20. van den Berg, J., Patil, S., Alterovitz, R.: Motion planning under uncertainty using iterative local optimization in belief space. Int. J. Robotics Research **31**(11) (September 2012) 1263–1278
21. Platt Jr., R., Tedrake, R., Kaelbling, L., Lozano-Perez, T.: Belief space planning assuming maximum likelihood observations. In: Robotics: Science and Systems (RSS). (2010)
22. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT Press (2005)
23. Welch, G., Bishop, G.: An introduction to the Kalman filter. Technical Report TR 95-041, University of North Carolina at Chapel Hill (July 2006)
24. van den Berg, J., Patil, S., Alterovitz, R., Abbeel, P., Goldberg, K.: LQG-based planning, sensing, and control of steerable needles. In Hsu, D., Others, eds.: Algorithmic Foundations of Robotics IX (Proc. WAFR 2010). Volume 68 of Springer Tracts in Advanced Robotics (STAR)., Springer (December 2010) 373–389