icLQG: Combining Local and Global Optimization for Control in Information Space

Vu Anh Huynh and Nicholas Roy

***Abstract*— When a mobile robot does not have perfect knowledge of its position, conventional controllers can experience failures such as collisions because the uncertainty of the position is not considered in choosing control actions. In this paper, we show how global planning and local feedback control can be combined to generate control laws in the space of distributions over position, that is, in information space. We give a novel algorithm for computing "information-constrained" linear quadratic Gaussian (icLQG) policies for controlling a robot with imperfect state information. The icLQG algorithm uses the belief roadmap algorithm to efficiently search for a trajectory that approximates the globally-optimal motion plan in information space, and then iteratively computes a feedback control law to locally optimize the global approximation. The icLQG algorithm is not only robust to imperfect state information but also scalable to high-dimensional systems and environments. In addition, icLQG is capable of answering multiple queries efficiently. We demonstrate performance results for controlling a vehicle on the plane and a helicopter in three dimensions.**

I. INTRODUCTION

We consider the problem of controlling a robot from a starting location to a specific destination in a world where absolute position information such as GPS is unavailable, also known as the partially observable stochastic shortest path (POSSP) problem [1], [2], [3]. When the state of the agent is not fully observable at all times, the most likely state can often be inferred from the history of actions the agent has taken and observations received from the environment. However, in many reasonable situations, the trajectory of the vehicle has an impact on what sensor measurements are received, and a corresponding impact on how accurate the localization process is. When the sensor measurements are uninformative, the position distribution is uncertain and the most likely position estimate is likely to be wrong, leading to potential failures such as collisions. We can improve robot control by incorporating the position uncertainty into choosing control actions.

Existing work in controlling dynamic systems has focused on generating controllers that are robust to the state uncertainty that can result from imperfect or limited sensors. However, when the sensor models are non-linear or discontinuous, existing control algorithms cannot actively control the state uncertainty. In contrast, planning algorithms that do incorporate state uncertainty cannot easily be extended to include continuous system dynamics due to limits of computational scalability. As a result, the planner cannot use the inherent dynamics of the system in generating efficient yet informative motion.

Vu Anh Huynh and Nicholas Roy are members of the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 77 Massachusetts Ave., Cambridge, MA 02139. The authors would like to thank the Singapore-MIT Alliance (SMA) for supporting Vu Anh Huynh to pursue this work, and NSF Division of Information and Intelligent Systems, grant # 0546467 for supporting Nicholas Roy. vuhuynh@mit.edu, nickroy@mit.edu

In this paper we show how global planning and local feedback control can be combined to generate control laws in information space, that is, the space of full probability distributions over the state space. By computing a control in information space, we can explicitly control the uncertainty of the state distributions. We give a novel algorithm for computing "information-constrained" linear-quadratic-Gaussian (icLQG) policies in two phases. The icLQG algorithm operates by first constructing an approximate global plan using the belief roadmap algorithm (BRM) [4] [5], which generates a set of waypoints based on the utility of sensing at each waypoint. The icLQG algorithm then uses iterative LQG (iLQG) [6] to compute a feedback control law that satisfies the sensor, or "information" constraints specified by the BRM trajectory.

This paper provides two contributions. Firstly, we show how the BRM approximation to the optimal trajectory can be constructed using iLQG [6] [7] to produce a non-linear trajectory that is globally "information-constrained" to provide required sensor information. Our second contribution is to show how the BRM algorithm can make use of an iLQG controller to efficiently estimate the cost function associated with each edge in a belief graph. We first formulate the mathematical model in Section II. Section III provides the iLQG algorithm that allows us to solve for the control policy of a non-linear system. The development of the icLQG algorithm, including identifying information constraints in an offline phase and an online optimization of the control policy are described in Sections IV and V. The main experiments and results follow in Section VI. Finally, related work and conclusions are provided in Sections VII and VIII.

II. PROBLEM STATEMENT

We consider a stationary continuous-time non-linear stochastic dynamic system governed by the following ordinary differential equation: $dx(t) = f(x, u)dt + dw(t)$, where

- $x(t) \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$ is the state of the system,
- $u(t) \in \mathcal{U} \subseteq \mathbb{R}^{n_u}$ is the control input,
- $w(t) \in \mathbb{R}^{n_w}$ is Brownian noise, where $n_x = n_w$,
- $f : \mathbb{R}^{n_x+n_u} \mapsto \mathbb{R}^{n_x}$ is a non-linear function of the state and control input.

In the above system, the state $x(t)$ is an element of a space $\mathcal{X}$ that represents state constraints. Similarly, the control $u(t)$ is an element of a space $\mathcal{U}$ that is a set of admissible controls.

When the system state is partially observable, we can infer a probability distribution over the state space from observations received according to $y(t) = g(x)+\vartheta(t)$, where

- $y(t) \in \mathbb{R}^{n_y}$ is the measurement output of the system,

- $\vartheta(t) \in \mathbb{R}^{n_\vartheta}$ is Brownian noise, where $n_y = n_\vartheta$,
- $g : \mathbb{R}^{n_x} \mapsto \mathbb{R}^{n_y}$ is a non-linear function of the state.

Given the distribution over states, we would like to find a feedback control law $\pi^* = \underset{\Pi}{\operatorname{argmin}} J_\pi\big(I(0)\big)$, that minimizes the expected cost from the start time 0 to an unspecified final time $\mathcal{T}$ given an initial observation $I(0) = y(0)$ and a starting state $x(0)$:

$$J_\pi\big(I(0)\big) = E\left[h\big(x(\mathcal{T})\big) + \int_0^{\mathcal{T}} \ell\Big(t, x(t), \pi\big(t, I(t)\big)\Big) dt \Big| I(0)\right],$$

where

- the expectation is taken over the conditional distribution of $x(0), \ldots, x(\tau)$ given $I(0)$,
- $h\big(x(\mathcal{T})\big)$ is the final stage cost,
- $\ell\big(t, x, u\big)$ is the instantaneous cost of control $u(t)$ at state $x(t)$,
- $u(t) = \pi\big(t, I(t)\big)$, with $I(t) = \{y(0..t), u(0..t^-)\}$,
- and Π is the set of admissible control laws: $\pi \in \Pi$.

The information $I(t)$ stores all measurements up to and including the current measurement as well as all past control inputs. Using $I(t)$, a feedback control law π chooses a control at time t using $u(t) = \pi(t, I(t))$.

Discrete-Time Model

The continuous time formulation is extremely difficult to optimize, especially for finite-horizon problems. As a result, most algorithms approximate the solution using a discrete time formulation. Let us discretize the time into intervals, with each time step lasting Δ seconds; thus the control horizon is $N = \frac{\mathcal{T}}{\Delta}$. The problem under consideration can be approximated as the following constrained optimization problem:

$$\min_{\Pi} E\left[h(x_N) + \sum_{k=0}^{N-1} \ell_k(x_k, u_k) \Big| I_0\right] \tag{1}$$

subject to:

$$x_{k+1} = x_k + f(x_k, u_k)\Delta + w_k\sqrt{\Delta}, \tag{2}$$
$$y_k = g(x_k) + \vartheta_k \tag{3}$$
$$\pi = \{u_0, u_1, ..., u_{N-1}\}, \tag{4}$$
$$\ell_k\big(x_k, u_k\big) = \ell\big((k-1)\Delta, x_k, u_k\big)\Delta \tag{5}$$
$$x_0 \sim N(x_0^-, \Lambda_0^-),\ I_0 = y_0, \tag{6}$$

where $k = \overline{1, N-1}$, and the discrete variables are samples of continuous counterparts $x_k \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$ and $u_k \in \mathcal{U} \subseteq \mathbb{R}^{n_u}$, and (2) is the approximation of the Ito integral. This formulation is open-loop as the specific u_k do not depend on the current history $I(k)$.

To create a feedback policy, we define each control $u_k = \mu(I_k)$, such that

$$\pi = \{\mu_0, \mu_1, ..., \mu_{N-1}\},$$
$$\ell_k\big(x_k, \mu_k(I_k)\big) = \ell\big((k-1)\Delta, x_k, \mu_k(I_k)\big)\Delta.$$

Rather than explicitly maintaining the complete information vector I_k, we can maintain a mean $\hat{x}_k = E[x_k|I_k]$ and covariance $\Lambda_k = Var[x_k|I_k]$ using the extended Kalman filter (EKF) [8] [9]. For a linear-Gaussian system, the Gaussian distribution is a sufficient statistic over the information history; for a non-linear system, the mean and covariance produced by the local linearization of the EKF is (usually) a good approximation to the information vector. We assume that the w_k are i.i.d according to $w_k \sim N(0, \Omega^w)$; ϑ_ks are i.i.d according to $\vartheta_k \sim N(0, \Omega^\vartheta)$; x_0 has a Gaussian distribution $N(x_0^-, \Lambda_0^-)$, where x_0^- is a mean value. Given the system dynamics in (2) and (3), the EKF updates are:

Predictive step:

$$\hat{x}_{k+1}^- = \hat{x}_k + f(\hat{x}_k, u_k)\Delta,\ A_k = I + \frac{\partial f}{\partial x}\Big|_{\hat{x}_k, u_k}\Delta,$$
$$\Lambda_{k+1}^- = A_k \Lambda_k A_k' + \Delta\Omega^w.$$

Update step:

$$\hat{x}_{k+1} = \hat{x}_{k+1}^- + K_k\big(y_{k+1} - g(\hat{x}_{k+1}^-)\big),$$
$$\Lambda_{k+1} = \Lambda_{k+1}^- - K_k F_{k+1} \Lambda_{k+1}^-,$$
$$K_k = \Lambda_{k+1}^- F_{k+1}'(F_{k+1}\Lambda_{k+1}^- F_{k+1}' + \Omega^v)^{-1},$$
$$F_{k+1} = \frac{dg}{dx}\Big|_{\hat{x}_{k+1}^-}.$$

III. iLQG ALGORITHM

In order to solve for a sequence of controls that minimizes our expected cost for a non-linear system, the iLQG algorithm [6] [7] linearizes the system dynamics and approximates the cost function up to second order around a series of nominal trajectories. The first nominal trajectory is chosen arbitrarily, and iteratively improved until a convergence criteria is met, leading to a second-order approximation of the optimal trajectory.

Let us assume that a sequence of open-loop control inputs $\bar{u}_k$ is given; we can then generate a nominal trajectory $\bar{x}_k$ by simulating the dynamics without noise using control inputs $\bar{u}_k$, such that $\bar{x}_{k+1} = f(\bar{x}_k, \bar{u}_k)$. Given a nominal trajectory $\bar{x}_k$, if function $\ell_k(x_k, u_k)$ is separable for variables x_k and u_k, such that $\frac{\partial^2 \ell_k}{\partial x \partial u} = 0$, we can then approximate the system dynamics for *any* sequence of states x_k as:

$$\delta x_k = x_k - \bar{x}_k,\ \delta u_k = u_k - \bar{u}_k, \tag{7}$$
$$\delta x_{k+1} = A_k \delta x_k \Delta + B_k \delta u_k \Delta + C_k w_k, \tag{8}$$
$$A_k = I + \frac{\partial f}{\partial x}\Big|_{\bar{x}_k, \bar{u}_k},\ B_k = \frac{\partial f}{\partial u}\Big|_{\bar{x}_k, \bar{u}_k},\ C_k = \sqrt{\Delta}. \tag{9}$$

Additionally, we can approximate the terminal cost as

$$h(x_N) = \delta x_N' Q_N \delta x_N + q_N' \delta x_N + p_N = h(\delta x_N), \tag{10}$$
$$Q_N = \frac{d^2 h}{2dx^2}\Big|_{\bar{x}_N},\ q_N = \frac{dh}{dx}\Big|_{\bar{x}_N},\ p_N = h(\bar{x}_N), \tag{11}$$

and the instantaneous costs as

$$\ell_k(x_k, u_k) = \delta x_k' Q_k \delta x_k + q_k' \delta x_k + \delta u_k' T_k \delta u_k + t_k' \delta u_k + p_k = \ell_k(\delta x_k, \delta u_k), \tag{12}$$
$$Q_k = \frac{\partial^2 \ell_k}{2\partial x^2}\Big|_{\bar{x}_N, \bar{u}_k}, q_k = \frac{\partial \ell_k}{\partial x}\Big|_{\bar{x}_N, \bar{u}_k}, \tag{13}$$
$$T_k = \frac{\partial^2 \ell_k}{2\partial u^2}\Big|_{\bar{x}_N, \bar{u}_k}, t_k = \frac{\partial \ell_k}{\partial u}\Big|_{\bar{x}_N, \bar{u}_k}, p_k = \ell_k(\bar{x}_k, \bar{u}_k). \tag{14}$$

The approximations of (7-14) result from expanding the Taylor's series of (2-6) up to the second order around the nominal trajectory $\bar{x}_k$ and nominal control inputs $\bar{u}_k$. Variables δx_k and δu_k represent the deviation from nominal states $\bar{x}_k$ and nominal control inputs $\bar{u}_k$ of actual random

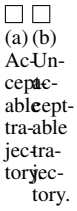


Fig. 1. Example trajectories. In (a), the robot motion is relatively accurate and so the robot position remains well-localized. In (b), the robot motion is extremely noisy, and so the robot position becomes uncertain quickly. The controller cannot model the non-linear range limit of its sensor and therefore cannot adjust the trajectory to improve its sensing.

variables x_k and u_k. Notice that each cost term can be written solely as a function of the deviation variables; therefore, within a *tube* along a nominal trajectory, we solve a *local* optimization problem in terms of the expected deviations from the nominal.

Given the optimization problem described in (1) and the second-order approximation described in (7-14), in one iteration, we can generate a new control law around a nominal trajectory using the following backward recursive equations (a brief derivation is given in Section IV-C):

$$u_k = \mu_k(I_k) = \bar{u}_k + l_k + L_k(\hat{x}_k - \bar{x}_k), \tag{15}$$

where

$$l_k = -\frac{1}{2}H_k g_k,\ L_k = -\frac{1}{2}H_k G_k, \tag{16}$$

$$H_k = T_k + B_k' S_{k+1} B_k,\ g_k = (t_k' + {s_{k+1}}' B_k)', \tag{17}$$

$$G_k = 2(A_k' S_{k+1} B_k)',\ P_k = L_k' H_k L_k + {G_k}' L_k, \tag{18}$$

$$S_k = Q_k + A_k' S_{k+1} A_k + P_k, \tag{19}$$

$$s_k = (q_k' + {s_{k+1}}' A_k + l_k' G_k + 2 l_k' H_k L_k + {g_k}' L_k)', \tag{20}$$

$$\hat{x}_k = E[x_k | I_k], \Lambda_k = Var[x_k | I_k]. \tag{21}$$

Since S_k, s_k, G_k and g_k depend on S_{k+1} and s_{k+1}, we set

$$S_N = Q_N, s_N = q_N. \tag{22}$$

Then, the new candidate for the next nominal open-loop control sequence can be computed by $\bar{u}_k^{new} = \rho u_k + (1 - \rho)\bar{u}_k$, where $\rho \in [0, 1]$. The iterative process stops when a nominal open-loop control sequence converges.

Note that l_k is the refined open-loop component, and L_k is the feedback gain matrix for the time index k. Furthermore, the weight matrix S_k represents the effort which the controller should spend to reach the destination, due to the deviation from $\bar{x}_k$ from the time index k to the final time index N. Note also that the solution to $\mu_k(I_k)$ may violate the constraints on u_k, such that $u_k \in \mathcal{U}_k$, for example actuator limits. The solution to the fully constrained optimization is outside the scope of this paper, but roughly speaking, the constraints can be incorporated using heuristics to project the control into the admissible set.

Figure 1(a) shows an example trajectory. The iLQG algorithm produces a trajectory for a planar car from the green star to the red star. The sensor infers its position from the known position of the features shown by the red dots, and the dashed blue circles around the features are the maximum range at which the features can be sensed. There are three trajectories plotted in this figure: the nominal trajectory in blue, the true trajectory in red, and the estimated trajectory in green. The covariances of the position estimates are shown by the ellipses around the estimated trajectory. The covariances become larger over time if the car is not within range of landmarks.

Unfortunately, while the iLQG algorithm can generate non-linear trajectories, the solution to the control policy does not depend on the covariance of the state estimate. As a result, the policy can fail when the covariances become large. Figure 1(b) depicts a trajectory with a motion model that is substantially noisy. When the covariances are large, the location of the car becomes highly uncertain. Because the sensor is limited in range and extremely non-linear, iLQG is unable to improve the trajectory by ensuring better sensing and therefore better position estimation. As a result, iLQG risks collisions with the obstacles represented by the solid red circles. To mitigate this issue, we incorporate the belief roadmap (BRM) [4] [5] to provide global guidance.

IV. Information Constraints

The BRM algorithm is a stochastic sampling planning algorithm, related to the probabilistic roadmap-based (PRM) algorithms [10] but in information space. The BRM optimizes a similar objective function as (1) under the assumption of Gaussian state distributions, by sampling distributions in the form $b = N(x, \Lambda)$ in information space and searching for a planned trajectory that leads to the minimum expected cost at the goal.

There are three open questions in implementing the BRM algorithm. Firstly, we must determine the sequence of controls required to move from an initial distribution $b_0 = N(x_0, \Lambda_0)$. We will use the iLQG algorithm described in Section III to determine the control policy from b_i to b_j. Secondly, we would like an efficient way of determining the transfer function for computing the posterior covariance given a control policy, which will be described in Section IV-B following [4] [5]. Finally, we would like an efficient way of determining the cost of the control policy along an edge: this is the second major contribution of our paper and is described in Section IV-C. By using the iLQG algorithm to determine a control policy for an edge, we can also determine a one-step cost update for the control policy, improving the approximation quality of the solution.

The icLQG algorithm addresses these three questions in the offline phase by using the BRM algorithm to determine an approximation to the globally optimal control policy. Since the BRM algorithm is a random sampling algorithm, it only converges to the optimal policy in the limit of infinite number of samples. Thus, the final online icLQG algorithm will use the BRM waypoints as constraints in the iLQG optimization (hence "information constrained" LQG), leading to a final trajectory that will be a locally optimal trajectory in information space.

A. Computing a local feedback control law for an edge

Let us begin by constructing the nodes of a graph $\mathcal{G}$ by sampling mean positions from the free part of the world configuration space. We add an edge between any two nodes if we can obtain a collision-free local feedback control law with a nominal velocity v_0, to traverse the edge using the iLQG algorithm. We do this by solving the following subproblem: let $b_i = N(x_i^s, \Lambda_i^s)$ and $b_j = N(x_j^s, \Lambda_j^s)$ be two

vertices in a belief graph. The superscript s in this section indicates sampled nodes in the graph. A local feedback control law π^{ij} to traverse the edge connecting x_i^s and x_j^s is obtained by running the iLQG algorithm with the following parameters:

$$N = N^{ij} = \frac{distance(x_i^s, x_j^s)}{v_0},$$
$$h(x_N) = (x_N - x_j^s)' Q_N (x_N - x_j^s),\ Q_N \succ 0,$$
$$\ell_k(x_k, u_k) = u_k' R_k u_k,\ R_k \succ 0,\ x_0 \sim N(x_i^s, \Lambda_0^s).$$

B. Constructing covariance transfer functions

Using a matrix inversion lemma [4] [5], we can factor the covariance matrix. If a conditional covariance Λ_k is factored as $\Lambda_k = \mathcal{B}_k \mathcal{C}_k^{-1}$, the predictive covariance, and the update covariance can be factored as $\Lambda_{k+1}^- = \mathcal{B}_{k+1}^- (\mathcal{C}_{k+1}^-)^{-1},\ \Lambda_{k+1} = \mathcal{B}_{k+1} \mathcal{C}_{k+1}^{-1}$, where

$$\begin{bmatrix} \mathcal{B}_{k+1}^- \\ \mathcal{C}_{k+1}^- \end{bmatrix} = \begin{bmatrix} A_k & \Delta\Omega^w (A_k')^{-1} \\ 0 & (A_k')^{-1} \end{bmatrix} \begin{bmatrix} \mathcal{B}_k \\ \mathcal{C}_k \end{bmatrix} = \begin{bmatrix} \xi_{k+1}^- \end{bmatrix} \begin{bmatrix} \mathcal{B}_k \\ \mathcal{C}_k \end{bmatrix},$$

$$\begin{bmatrix} \mathcal{B}_{k+1} \\ \mathcal{C}_{k+1} \end{bmatrix} = \begin{bmatrix} I & 0 \\ F_{k+1}' (\Omega^v)^{-1} F_{k+1} & I \end{bmatrix} \begin{bmatrix} \mathcal{B}_{k+1}^- \\ \mathcal{C}_{k+1}^- \end{bmatrix} = [\xi_{k+1}] \begin{bmatrix} \mathcal{B}_{k+1}^- \\ \mathcal{C}_{k+1}^- \end{bmatrix}.$$

Theorem 4.1: (Covariance transfer function) Given a control law π^{ij} for N time steps in a reasonable environment from x_i^s to x_j^s, there is a linear operator ξ^{ij} such that the final conditional covariance can be predicted as $[\Psi] = \begin{bmatrix} \xi^{ij} \end{bmatrix} \begin{bmatrix} \Lambda_0 \\ I \end{bmatrix}$, $\Lambda_N = [\Psi_{1,1}] [\Psi_{2,1}]^{-1}$.

Proof: Using the control law π^{ij} to *simulate a trajectory once*, the controller provides a control consequence $\{u_0, u_1, ..., u_{N-1}\}$, and the robot receives a measurement sequence $\{y_1, y_2, ..., y_{N-1}, y_N\}$. Let $\Lambda_0 = \Lambda_0 I^{-1}$, we can construct matrices $\xi_1^-, \xi_1, \xi_2^-, ..., \xi_N^-, \xi_N$ such that:

$$[\Psi] = \begin{bmatrix} \mathcal{B}_N \\ \mathcal{C}_N \end{bmatrix} = [\xi_N] \begin{bmatrix} \xi_N^- \end{bmatrix} ... [\xi_1] \begin{bmatrix} \xi_1^- \end{bmatrix} \begin{bmatrix} \Lambda_0 \\ I \end{bmatrix}$$

So: $\xi^{ij} = [\xi_N] \begin{bmatrix} \xi_N^- \end{bmatrix} ... [\xi_1] \begin{bmatrix} \xi_1^- \end{bmatrix}$, and $\Lambda_j^s = [\Psi_{1,1}] [\Psi_{2,1}]^{-1}$. ■

Each ξ_k contains Jacobians that are computed by linearizing around the specific mean $\hat{x}_k$. However, under mild assumptions, we expect that generated control inputs and received measurements would not be significantly different in several queries, and the Jacobians can therefore be approximated as constant. Thus, the matrix ξ^{ij}, which is defined as a covariance transfer function, is *computed once* using a simulation of the control law π^{ij}. Given an initial series of computation to construct ξ^{ij} and a starting covariance $\Lambda_0 = \Lambda_i^s$ as an input parameter, repeated queries of the effect of a series of controls and observations can be calculated efficiently. This significantly improves the computational speed of the EKF prediction.

C. Constructing cost transfer functions

Before providing a suitable format for cost transfer functions, let us prove (15-22) in one iteration of the iLQG algorithm.

Theorem 4.2: Given the approximation in (7-14), in a typical iteration of the iLQG algorithm, a new control law around a nominal trajectory can be computed recursively backward using (15-22).

Proof: First, we will prove by backward induction that sub-problem $J_k(I_k)$ from the time index k to the time index N has the approximated reduced form:

$$J_k(I_k) = E_{x_k} [r_k + s_k' \delta x_k + \delta x_k' S_k \delta x_k | I_k]. \quad (23)$$

The term r_k represents the expected effort the controller should spend if there is no deviation at the time index k. This effort also takes into account of future disturbances encoded in Λ_k to Λ_N. Indeed, we have for $k = N$:

$$J_N(I_N) = E_{x_N} \left[p_N + q_N' \delta x_N + \delta x_N' Q_N \delta x_N \middle| I_N \right],$$

and we set S_N, s_N according to (22), and and $r_N = p_N$. Using dynamic programming (DP), we have the unconstrained minimization:

$$J_k(I_k) = \min_{u_k} E \left[\ell_k(x_k, u_k) + J_{k+1}(I_{k+1}) \middle| I_k, u_k \right].$$

Assuming $J_{k+1}(I_{k+1})$ has the corresponding proved format, expanding all sub terms with $\delta x_{k+1} = A_k \delta x_k + B_k \delta u_k + C_k w_k$ in the terms $\ell_k(x_k, u_k)$, $E\left[\delta x_{k+1}' S_{k+1} \delta x_{k+1} | x_k, u_k\right]$, and $E\left[{s_{k+1}}' \delta x_{k+1} | x_k, u_k\right]$, we have:

$$J_k(I_k) = E \left[\delta x_k' S_k^- \delta x_k + {s_k^-}' \delta x_k + r_k^- | I_k \right] + \min_{u_k} E \left[\delta u_k' H_k \delta u_k + (g_k + G_k \delta x_k)' \delta u_k \middle| I_k, u_k \right].$$

where

$$S_k^- = Q_k + A_k' S_{k+1} A_k,\ s_k^- = (q_k' + {s_{k+1}}' A_k)', \quad (24)$$
$$r_k^- = r_{k+1} + p_k + tr(C_k' S_{k+1} C_k \Omega_k^w), \quad (25)$$
$$H_k = T_k + B_k' S_{k+1} B_k,\ g_k = (t_k' + {s_{k+1}}' B_k)', \quad (26)$$
$$G_k = (A_k' (S_{k+1} + {S_{k+1}}') B_k)'. \quad (27)$$

Let us denote $\delta\hat{x}_k = E[x_k | I_k] - \bar{x}_k = \hat{x}_k - \bar{x}_k$, and minimize over δu_k without constraints, we have:

$$(H_k + H_k')\delta u_k + g_k + G_k \delta\hat{x}_k = 0 \Leftrightarrow \delta u_k = l_k + L_k \delta\hat{x}_k, \quad (28)$$

where $l_k = -(H_k + H_k')^{-1} g_k,\ L_k = -(H_k + H_k')^{-1} G_k$. Expanding δu_k in terms of l_k, and L_k in $\delta u_k' H_k \delta u_k$, ${g_k}' \delta u_k$, and $\delta x_k' {G_k}' \delta u_k$, we have $J_k(I_k) = E_{x_k} \left[r_k + s_k' \delta x_k + \delta x_k' S_k \delta x_k \middle| I_k \right]$ where:

$$\begin{aligned}
P_k &= L_k' H_k L_k + {G_k}' L_k, \\
S_k &= S_k^- + P_k = Q_k + A_k' S_{k+1} A_k + P_k, \\
s_k &= s_k^- + (l_k' G_k)' + (l_k' (H_k + H_k') L_k + {g_k}' L_k)' \\
&= (q_k' + {s_{k+1}}' A_k + l_k' G_k + l_k' (H_k + H_k') L_k + {g_k}' L_k)', \\
r_k &= r_k^- + l_k' H_k l_k + {g_k}' l_k - tr(P_k \Lambda_k), \\
&= r_{k+1} + p_k + tr(C_k' S_{k+1} C_k \Omega_k^w) + l_k' H_k l_k \\
&\quad + {g_k}' l_k - tr(P_k \Lambda_k).
\end{aligned}$$

In the above mathematical manipulation, we use the following properties:

- $tr\big(A\Lambda_k\big) = tr\big(AVar[x_k | I_k]\big) = tr\big(AVar[\delta x_k | I_k]\big) = E[\delta x_k' A \delta x_k | I_k] - \delta\hat{x}_k' A \delta\hat{x}_k$,
- We assume that H_k is invertible;
- We ignore the term r_k, which involves future conditional covariances in the term r_{k+1}. Under the *CE assumption*, we assume that future disturbances do not affect the choice of a control input at the current time.

Thus, we have proved (23). Furthermore, during this proof, from (28), $\delta u_k = l_k + L_k \delta \hat{x}_k$, we can infer that $u_k = \bar{u}_k + l_k + L_k(\hat{x}_k - \bar{x}_k)$. Note that, we can show that H_k, S_x matrices are symmetric in the next lemma, and thus, what we need to prove follows. ■

Lemma 4.1: Assume that ℓ_ks produce Q_k, T_k that are symmetric positive definite matrices, H_k and S_k are symmetric positive definite matrices.

Proof: For $k = N$, $S_N = Q_N$ is a symmetric positive definite matrix. Let us assume that S_{k+1} is a symmetric positive definite matrix. Indeed, $H_k = T_k + B_k' S_{k+1} B_k$ is a positive definite matrix as it is a sum of two positive definite matrices. We can check that $S_k = A_k' S_{k+1} A_k + L_k' H_k L_k - \frac{1}{2} G_k' H_k G_k$, which is again a symmetric positive definite. ■

From the proof, (23) suggests that the objective function value can be approximated as

$$J_0(I_0) = E_{x_0}\left[r_0 + s_0' \delta x_0 + \delta x_0' S_0 \delta x_0 | I_0\right] \tag{29}$$

$$\approx \widetilde{r}_0 + (\widetilde{s}_0)' \delta \hat{x}_0 + \delta \hat{x}_0' \widetilde{S}_0 \delta \hat{x}_0 + tr(\widetilde{P}\Lambda_0) \tag{30}$$

$$= \widetilde{r}_0 + tr(\widetilde{P}\Lambda_0). \tag{31}$$

The first approximation is due to $E_{x_0}\left[\delta x_0' S_0 \delta x_0 | I_0\right] = \delta \hat{x}_0' S_0 \delta \hat{x}_0 + tr(S_0 \Lambda_0)$, and r_0 contains terms relating to future covariances. Therefore, we approximate $\widetilde{P}$ as a weight matrix of uncertainty in the cost function. The last equality is due to $\delta \hat{x}_0 = \hat{x}_0 - \bar{x}_0 = \bar{x}_0 - \bar{x}_0 = 0$.

Thus, after obtaining a control law π^{ij} from iLQG for the edge $b_i b_j$, we learn $\widetilde{r}_0$ and $\widetilde{P}$ for that edge using stochastic iterative algorithms [2] as follows.

Theorem 4.3: (Cost transfer function) Starting with arbitrary values of $\widetilde{r}_0$ and $\widetilde{P}$, the following iterative procedure provides a numerical solution to the approximated values of $\widetilde{r}_0$ and $\widetilde{P}$ in $J_0(I_0) \approx \widetilde{J}(\Lambda_0) = \widetilde{r}_0 + tr(\widetilde{P}\Lambda_0)$:

- Initialize a random initial covariance Λ_0,
- Compute the current approximated value of $J_0^c = \widetilde{r}_0 + tr(\widetilde{P}\Lambda_0)$,
- Simulate value J_0^s of $J_{\pi^{ij}}$ when following a fixed iLQG control law π^{ij},
- Update $\widetilde{r}_0$ and $\widetilde{P}$ as $\widetilde{r}_0^{new} = \widetilde{r}_0 - \gamma(J_0^c - J_0^s)$, $\widetilde{P}^{new} = \widetilde{P} - \gamma(J_0^c - J_0^s)\Lambda_0$.

We reduce γ to 0 as the number of iterative steps increases.

Proof: Given the current values of $\widetilde{r}_0$ and $\widetilde{P}$, finding the next approximated values of these variables is equivalent to solving the following optimization problem:

$$\min_{\widetilde{r}_0, \widetilde{P}} \frac{1}{2}\left(\widetilde{r}_0 + tr(\widetilde{P}\Lambda_0) - J_0^s\right)^2.$$

Denote:

$$L(\widetilde{r}_0, \widetilde{P}) = \frac{1}{2}\left(\widetilde{r}_0 + tr(\widetilde{P}\Lambda_i^s) - J_0^s\right)^2$$

$$= \frac{1}{2}\left(\widetilde{r}_0 + E\left[(x_0 - \hat{x}_0)' \widetilde{P}(x_0 - \hat{x}_0) | I_0\right] - J_0^s\right)^2.$$

Using the gradient method, the next numerical solution is:

$$\widetilde{r}_0^{new} = \widetilde{r}_0 - \gamma \nabla_{\widetilde{r}_0} L(\widetilde{r}_0, \widetilde{P}) \tag{32}$$

$$= \widetilde{r}_0 - \gamma(\widetilde{r}_0 + tr(\widetilde{P}\Lambda_0) - J_0^s) \tag{33}$$

$$= \widetilde{r}_0 - \gamma(J_0^c - J_0^s), \tag{34}$$

$$\widetilde{P}^{new} = \widetilde{P} - \gamma \nabla_{\widetilde{P}} L(\widetilde{r}_0, \widetilde{P}) \tag{35}$$

$$= \widetilde{P} - \gamma(J_0^c - J_0^s)(E\left[(x_0 - \hat{x}_0)(x_0 - \hat{x}_0)' | I_0\right]) \tag{36}$$

$$= \widetilde{P} - \gamma(J_0^c - J_0^s)\Lambda_0. \tag{37}$$

Algorithm 1 Offline phase of the icLQG

```
procedure OFFLINEPHASE(map)
    G.V ← ∅, G.E ← ∅
    for all feature m do
        for i ← 1, 2, ..., numberofsamples do
            Sample x^s ∈ C_free
            Add b = (x^s, Λ^s = ∅) to set G.V
        end for
    end for
    for all vertex b_i do
        for all vertex b_j (i ≠ j) do
            if x_i^s x_j^s ∈ C_free then
                Add edge b_i b_j to set G.E
            end if
        end for
    end for
    for all edge b_i b_j ∈ G.E do
        N ← distance(x_i^s, x_j^s)/v_0
        h(x_N) = (x_N − x_j^s)' Q_N (x_N − x_j^s), ℓ_k(x_k, u_k) = u_k' R_k u_k
        π^ij = iLQG(f, g, h, ℓ, N, x_i^s, Λ_i^s)
        Simulate π^ij: b_i b_j.ξ ← [ξ_N] [ξ_N^−] ... [ξ_1] [ξ_1^−]
        Learn type-1 b_i b_j.J̃
    end for
    return G
end procedure
```

To ensure convergence, the step size γ in the gradient method should approach 0 as the number of iterations increases [2]. The convergence criteria for this procedure is when norms of the changes to $\widetilde{r}_0$ and $\widetilde{P}$ are below some thresholds. ■

Definition 4.1: The type-1 and type-2 cost transfer functions of an edge approximate the cost-to-go of a control law **without**, and **with** the final terminating cost respectively:

$$J_\pi^1(I_0) = E\left[\sum_{k=0}^{N-1} \ell_k(x_k, u_k) \Big| I_0\right],$$

$$J_\pi^2(I_0) = E\left[h(x_N) + \sum_{k=0}^{N-1} \ell_k(x_k, u_k) \Big| I_0\right].$$

As waypoints are often transition points in a trajectory, we are not concerned with the terminating cost at waypoints. Thus, the type-1 cost to traverse an edge $b_i b_j$ using the control law π^{ij} can be approximated from a transfer function $\widetilde{J}^{ij}$. The offline phase of the icLQG algorithm is summarized in Algorithm 1.

V. Online Search

In this section, the constructed information in the offline phase is used for multiple queries efficiently in the online phase. The mission is to plan a trajectory from an initial belief $N(x_0^-, \Lambda_0^-)$ to a destination with the final state x_G.

First, the initial starting location and the destination are added into the belief graph. We construct two additional nearest edges: a type-1 edge *from* the initial location and a type-2 edge *to* the destination, as well as their corresponding transfer functions. Moreover, we also construct a direct type-2 edge together with transfer functions from the initial location to the destination.

Second, from the initial belief, covariances at other vertices can be computed efficiently by propagating Λ_0^- using covariance transfer functions. Then, the approximated cost to traverse an edge is computed by plugging a covariance at a departing vertex into the associated cost transfer function of that edge. We consider these cost values as edge weights,

and therefore the Dijkstra's search can be applied to find a trajectory with the smallest cost-to-go. Thus, this Dijkstra's search provides us with a trade-off between keeping the covariance of a robot state estimate small and keeping the energy to find a destination small. This is well-known as the trade-off between exploration and exploitation [11].

Suppose that the Dijkstra's search provides a trajectory α with ordered waypoints $\{b_{\alpha_1}, b_{\alpha_2}, ..., b_{\alpha_m}\}$, and the corresponding horizons to reach these vertices from its preceding vertices are $\{N_{\alpha_1}, ..., N_{\alpha_m}, N_{\alpha_{m+1}}\}$. In particular, there are N_{α_1} time steps to traverse from the initial location to a vertex b_{α_1}, and there are $N_{\alpha_{m+1}}$ time steps to traverse from a vertex b_{α_m} to the final destination. Let N be a total number of time steps to follow the trajectory, and k_j is the time index to visit a vertex b_{α_j}. We have: $N = N_{\alpha_1} + N_{\alpha_2} + ... + N_{\alpha_m} + N_{\alpha_{m+1}}$, $k_j = N_{\alpha_1} + ... + N_{\alpha_j}$. The corresponding objective function to go from the initial location to a destination location via these waypoints in an induce optimization problem are defined as

$$\begin{gathered} h(x_N) = (x_N - x_G)' Q_N (x_N - x_G), \quad Q_N \succ 0, \\ \ell_k(x_k, u_k) = u_k' R_k u_k, \quad R_k \succ 0, \quad k \notin \{k_1, k_2, ..., k_m\}, \\ \ell_{k_j}(x_{k_j}, u_{k_j}) = u_{k_j}' R_{k_j} u_{k_j} + (x_{k_j} - x^s_{\alpha_j})' Q_{k_j} (x_{k_j} - x^s_{\alpha_j}), \\ R_{k_j} \succ 0, Q_{k_j} \succ 0. \end{gathered}$$

Again, the above optimization can be solved sup-optimally by the iLQG algorithm to obtain a local-feedback control law π. The overall summary of the icLQG online phase is presented in Algorithm 2.

VI. EXPERIMENTS AND RESULTS

We consider a planar car such that when applying a control $u(t) = [v\ \omega]_t'$ to a state $x(t) = [x\ y\ \theta]_t'$, the system follows the following dynamics:

$$d(x(t)) = f(x,u)dt + dw(t) = \begin{bmatrix} v\cos(\theta) \\ v\sin(\theta) \\ \omega \end{bmatrix} dt + dw(t).$$

The car is equipped with sensors to work in the area with landmarks locating at coordinates $[m_x\ m_y]'$. The car can sense the distance and relative bearing to the landmark if it is within the visibility region of the car. We assume that when the car senses the landmark, it knows the landmark coordinates. The measurement model is described by

$$\begin{aligned} if \quad & \sqrt{(m_x - x)^2 + (m_y - y)^2} \le R_m : \\ y(t,m) &= g(x,m) + \vartheta(t) \\ &= \begin{bmatrix} \sqrt{(m_x - x)^2 + (m_y - y)^2} \\ atan2(m_y - y, m_x - x) - \theta \end{bmatrix} + \vartheta(t), \end{aligned}$$

where R_m is the range of visibility of the car for that landmark.

A. Performance of the iLQG algorithm

We calculated ratios $\frac{\|\bar{u}^{k+1} - \bar{u}^*\|}{\|\bar{u}^k - \bar{u}^*\|^{1.3}}$ where $\bar{u}^*$ is the converged nominal control sequence. The ratios were in the range $(0, 1)$ with a rate of convergence measured experimentally at approximately 1.3 in this case. Note that when updating a new nominal control sequence, the first step is a Newton-like method, which has a quadratic rate of convergence. Then the new control sequence is interpolated with the old control sequence via the parameter ρ to avoid arbitrary diverged simulated control sequences due to noise. Thus, the rate of convergence is less than 2 as expected.

Algorithm 2 Online phase of the icLQG

procedure ONLINEPHASE($\mathcal{G}$, $N(x_0^-, \Lambda_0^-)$,x_G)
  Receive an initial measurement y_0
  Perform the EKF update to get $N(x_0, \Lambda_0)$
  $gz \leftarrow size(\mathcal{G}.V)$
  Add vertex $b_{gz+1} = (x_0, \Lambda_0)$ to $\mathcal{G}.V$
  Add vertex $b_{gz+2} = (x_G, \Lambda_G = \emptyset)$ to $\mathcal{G}.V$
  Add nearest type-1 edge $b_{gz+1}b_{j_1}$, type-2 edge $b_{j_2}b_{gz+2}$ to $\mathcal{G}.E$
  Add direct type-2 edge $b_{gz+1}b_{gz+2}$ to $\mathcal{G}.E$
  for all $b \in \mathcal{G}.V$ **do**
    $cost[b] \leftarrow \infty$, $prev[b] \leftarrow -1$
  end for
  $cost[b_{gz+1}] \leftarrow 0$, $Queue \leftarrow \mathcal{G}.V$
  while $!empty(Queue)$ **do**
    $b \leftarrow popMin(Queue)$
    if $b! = b_{gz+2}$ **then** break
    end if
    for all neighbor v of b **do**
      $[\Psi] \leftarrow [bv.\xi] \begin{bmatrix} b.\Lambda \\ I \end{bmatrix}$, $\Lambda \leftarrow [\Psi_{1,1}][\Psi_{2,1}]^{-1}$
      $alt \leftarrow cost[b] + bv.\widetilde{J}(\Lambda)$
      if $alt < cost[v]$ **then**
        $cost[v] \leftarrow alt$, $v.\Lambda \leftarrow \Lambda$, $prev[v] \leftarrow b$
      end if
    end for
  end while
  $wp = \{b_{\alpha_1}, ..., b_{\alpha_m}\} \leftarrow traceBack(prev)$
  $N \leftarrow N_{\alpha_1} + ... + N_{\alpha_{m+1}}$
  for $j \leftarrow 1, ..., m$ **do**
    $k_j \leftarrow N_{\alpha_1} + ... + N_{\alpha_j}$
  end for
  $h(x_N) = (x_N - x_G)' Q_N (x_N - x_G)$
  $\ell_k(x_k, u_k) = u_k' R_k u_k, \quad k \notin \{k_1, k_2, ..., k_m\}$
  $\ell_{k_j}(x_{k_j}, u_{k_j}) = u_{k_j}' R_{k_j} u_{k_j} + (x_{k_j} - x^s_{\alpha_j})' Q_{k_j} (x_{k_j} - x^s_{\alpha_j})$
  $\pi = iLQG(f, g, h, \ell, N, x_0, \Lambda_0)$
  Simulate the control law π to get a trajectory x
  return x, π
end procedure

TABLE I
AN EXAMPLE OF SETTINGS IN AN ENVIRONMENT

(Landmarks,Radius (m))
$(-15, -3, 7)$
$(3, -22, 8)$
$(2, 13, 7)$
$(20, -12, 4)$
$(14, 12, 5)$
$\Omega^w = \begin{bmatrix} 0.01 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 0.01 \end{bmatrix}$
$\Omega^\vartheta = \begin{bmatrix} 0.2 & 0 \\ 0 & 0.002 \end{bmatrix}$
$\Lambda_0^- = \begin{bmatrix} 0.3 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.001 \end{bmatrix}$

TABLE II
UNPLANNED V.S. PLANNED TRAJECTORY COSTS.

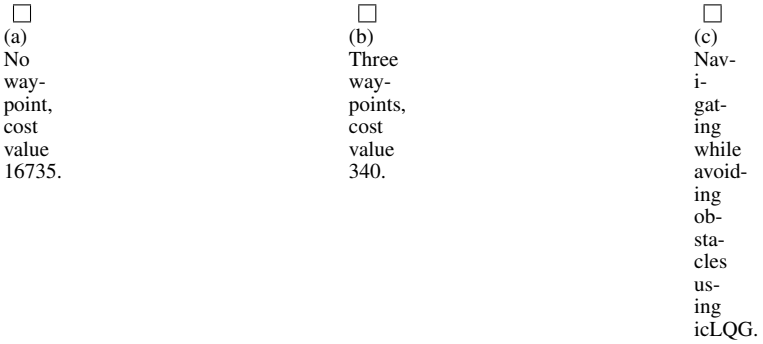
Std. dev. (m)	Unplanned cost	Planned cost
$\sqrt{0.1}$	1.74×10^4	370
$\sqrt{0.2}$	2.03×10^4	394
$\sqrt{0.3}$	2.11×10^4	408
$\sqrt{0.4}$	2.22×10^4	434
$\sqrt{0.5}$	2.42×10^4	445

TABLE III
RUNNING TIMES.

Offline time (s)	Online time (s)
600.43	1.32

B. Performance of transfer functions

In the second experiment, we compared the estimated covariances with the fully updated EKF covariances for final state estimates at incoming vertices. Random covariance matrices with standard deviations for the X, Y directions up to 1 meter were given at a departing vertex. Corresponding estimated covariances and fully updated EKF covariances were recorded. Figure 3(a) depicts the traces of estimated covariances versus the traces of fully updated covariances. As we can see, different initial covariances yield different final covariances, but the trace values of the final covariances are closely matched by the two methods. This observation verifies that variance transfer functions are able to preserve



(a) No waypoint, cost value 16735.

(b) Three waypoints, cost value 340.

(c) Navigating while avoiding obstacles using icLQG.

Fig. 2. Comparison of 2D trajectories.

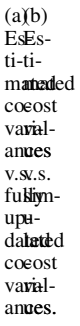
the accuracy of predicted covariances.

Next, we evaluated the performance of cost transfer functions for estimating the cost-to-go values for edges. Similar to the covariance transfer functions, different initial covariances were given and we recorded both the estimated cost values and average simulated cost values over 600 samples. Figure 3 shows a comparison of these cost values. These plots show that the proposed approximation structure $\widetilde{r}_0 + tr(\widetilde{P}\Lambda)$ is able to preserve the shape and trend of objective values with respect to initial covariances. The accuracy of estimated values by the cost transfer functions depends on the specific noise terms and the structure of the environment.

C. Planning and control

In the third experiment, we planned a mission to navigate from a starting location to a destination. There were five landmarks, and a belief graph with ten sampled vertices was created in the offline phase. The online phase returned a trajectory and a set of local feedback control laws to follow this trajectory. Figures 2(a) and 2(b) show the advantages of the icLQG algorithm in planning and controlling the car. On the first hand, Figure 2(a) plots a trajectory between two locations without going through any waypoints to reduce covariances. Thus, although the estimated mean of the final state is at the destination, the actual position of the car is far away from the destination. On the second hand, the icLQG algorithm provides the planned trajectory with three black waypoints in visibility areas in Figure 2(b). Therefore, the car is able to reach the destination with high accuracy despite the long curved route.

In Table II, we compare the average simulated cost values



(a) Estimated cost values v.s. fully updated cost values.

(b) Estimated cost values v.s. updated cost values.

Fig. 3. Evaluation of accuracy of covariance and cost transfer functions.

of the two trajectories. As we can see, the planned trajectory cost from the icLQG algorithm is significantly smaller than the unplanned trajectory cost. Hence, this result infers that more energy to traverse the longer trajectory benefits the car tremendously. In addition, when the standard deviations increase, the simulated cost of the planned trajectory increases slightly, which indicates that the icLQG algorithm is robust against initial state uncertainty.

Table III summarizes the running time in seconds to build the belief graph in the offline phase, and the searching in the online phase. As we can see, compared to the online time, the offline time is almost 455 times slower, which is substantially dominant in the icLQG algorithm. Thus, most of the burden of computation is moved to the offline phase.

Moreover, Figure 2(c) shows how the icLQG algorithm, which is a combination of local and global optimization, navigates and avoids obstacles using three waypoints as compared to Figure 1(b).

D. Scalability to high-dimensional systems

Finally, we verified that the algorithm is scalable to planning and controlling the helicopter in 3D environments. In this case, the problem has control constraints on the thrust components of control inputs. We compared the effect of trajectory planning and control using the iLQG algorithm alone and using icLQG. In Figure 4(a), we present an example of an unplanned trajectory, which is returned from the iLQG algorithm. As we can see, the covariances, which are represented by the dotted cloud around each estimated mean, are enormous. Thus, the helicopter is uncertain about reaching the desired destination. In contrast, in Figure 4(b), a planned trajectory with two waypoints in black, which is returned from the combination of the Dijkstra's search in the belief graph and the iLQG algorithm, is used. This trajectory arrives at the destination with high accuracy by taking the advantage of the necessary nearby features.

VII. RELATED WORK

Approaching from the local solution point of view, Li and Todorov have recently proposed the iLQG algorithm [12] [7] [6] to address the problem of control in partially observable environments. The main idea of their iLQG version is similar to what have been discussed here with more complicated noise. However, with strongly non-linear problems like helicopter control, iLQG does not guarantee

Fig. 4. Comparison of 3D trajectories.

a global optimum. Thus, icLQG provides high level global guidance to the final iLQG run.

The idea of using landmarks to provide additional information in motion planning with uncertainties was addressed earlier in [13]. In this work, Lazanas et al. proposed a sound and complete planner of polynomial complexity using landmark information. Unlike icLQG, their algorithm uses preimage planning approach with worst case analysis under an assumption that uncertainties are bounded. Hence, their algorithm concerns the likelihood of achieving a goal but barely addresses the efficiency of the solution, which is the main interest of icLQG.

Roy et al. [4] [5] [14] proposed the belief roadmap (BRM) algorithm to use an available map to plan a set of waypoints to a destination. The method can efficiently plan a trajectory even in large-scale environments. However, they only consider minimizing the final uncertainty at a destination but not consumed energy. Furthermore, they do not provide a corresponding control law to follow this trajectory.

VIII. CONCLUSION

In this research, we have addressed the problem of planning and controlling mobile robots in partially observable stochastic environments. The icLQG algorithm is inspired by the PRM and BRM algorithms [10] [5] [14]. The key concept is to sample in the mean space and search in the covariance space of robots' belief states. To plan a trajectory efficiently, each edge of the belief graph is associated with a covariance transfer function and a cost transfer function. These functions parameterize the set of different trajectories based on covariances of initial beliefs. The icLQG algorithm also provides an approximate control policy along a planned trajectory using the iLQG algorithm [6]. The principle of the iLQG algorithm is based on solving successive linear quadratic Gaussian problems using dynamic programming. Overall, the icLQG algorithm is a method to solve the partially observable stochastic shortest path problem.

To the best of our knowledge, it is the first time that the problem of planning and controlling a six-degree-of-freedom helicopter with *coastal navigation* trajectories in continuous spaces is reported. Finally, as we have been working so far in simulation, the next step would be to verify the icLQG algorithm on real hardware.

REFERENCES

[1] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Athena Scientific, 2007, vol. I–II.

[2] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming (Optim. and Neural Comp. Series, 3)*. Athena Scientific, May 1996.

[3] J. N. Tsitsiklis, "Efficient algorithms for globally optimal trajectories," *IEEE Trans. Automatic Control*, vol. 40, no. 9, pp. 1528–1538, 1995.

[4] S. Prentice and N. Roy, "The Belief Roadmap: Efficient planning in linear POMDPs by factoring the covariance," in *Proc. 13th International Symposium of Robotics Research (ISRR)*, Hiroshima, Japan, 2007.

[5] R. He, S. Prentice, and N. Roy, "Planning in information space for a quadrotor helicopter in a GPS-denied environment," in *Proc. ICRA*, 2008.

[6] W. Li and E. Todorov, "Iterative linearization methods for approximately optimal control and estimation of non-linear stochastic systems," *International Journal of Control*, vol. 80, pp. 1439–1453, 2007.

[7] ——, "Iterative optimal control and estimation design for nonlinear stochastic systems," in *Proceedings of the 45th IEEE Control and Decision Conference*, San Diego, California, December 2006.

[8] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, September 2005.

[9] M. I. Ribeiro, "Kalman and extended Kalman filters: Concept, derivation and properties," 2004.

[10] L. Kavraki, P. Svestka, J. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[11] A. Simpkins, R. de Callafon, and E. Todorov, "Optimal trade-off between exploration and exploitation," in *Proceedings of the American Control Conference*, Seattle, Washington, June 2008, pp. 300–306.

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

[13] A. Lazanas and J.-C. Latombe, "Motion planning with uncertainty: a landmark approach," *Artif. Intell.*, vol. 76, no. 1-2, pp. 287–317, 1995.

[14] N. Roy and S. Thrun, "Coastal navigation with mobile robots," in *Advances in Neural Processing Systems 12*, vol. 12, 1999, pp. 1043–1049.