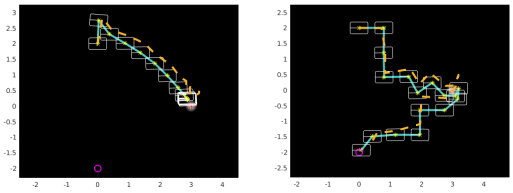
T-LQG: Closed-Loop Belief Space Planning via Trajectory-Optimized LQG*

Mohammadhussein Rafieisakhaei[1], Suman Chakravorty[2] and P. R. Kumar[1]

***Abstract*— Planning under motion and observation uncertainties requires the solution of a stochastic control problem in the space of feedback policies. In this paper, by restricting the policy class to the linear feedback polices, we reduce the general (n^2+n)-dimensional belief space planning problem to an (n)-dimensional problem. As opposed to the previous literature that search in the space of open-loop optimal control policies, we obtain this reduction in the space of closed-loop policies by obtaining a Linear Quadratic Gaussian (LQG) design with the best nominal performance. Then, by taking the entire underlying trajectory of the LQG controller as the decision variable, we pose a coupled design of the trajectory and estimator (while keeping the design of the controller separate) as a NonLinear Program (NLP) that can be solved by a general NLP solver. We prove that under a first-order approximation and a careful usage of the separation principle, our approximations are valid. We provide an analysis on the existing major belief space planning methods and show that our algorithm keeps the lowest computational burden while searching in the policy space. Finally, we extend our solution to contain general state and control constraints. Our simulation results support our design.**

I. Introduction

Planning under process and sensing uncertainties is referred to as the belief space planning problem. In general, it requires the solution of the Hamilton-Jacobi-Bellman (HJB) equations to obtain the optimal feedback policy [1], [2]. The Linear Quadratic Gaussian (LQG) methodology provides the optimal estimator and controller for linear systems with Gaussian noises [3]. However, an LQG planner requires a nominal trajectory to begin with. One approach utilizing this methodology decouples the two procedures of designing the trajectory and the policy and provides a sequential design of the trajectory and the LQG policy (the estimator plus controller), either by providing various different a priori trajectories and comparing the LQG performance over each one [4], or by performing the decoupled procedure iteratively [5], [6]. An approach to a coupled design of trajectory and the policy in nonlinear systems utilizing the Extended Kalman Filter (EKF) is based on the heuristic assumption of Most-Likely Observations (MLO) during planning [7], [8]. Another approach considers general belief distributions, either by reformulating the problem in belief space as a Partially Observed Markov Decision Process (POMDP) [9] or by utilizing a Monte-Carlo representation of beliefs [10], [11]. The state-of-the-art POMDP solvers are posed on decision trees, reducing the search space to a finite set of reachable belief nodes given an initial belief. However, this approach to solve POMDPs for continuous action and observation spaces requires continuous (uncountable) branching in the decision tree of beliefs, which leads to intractable computations.



(a) RHC-based results after $K = 16$ steps

(b) T-LQG results after one plan and execution

Fig. 1. Comparison between T-LQG and an MLO plus RHC-based method [8]. In each figure, dashed line shows the ground truth trajectory and solid line shows the state estimate trajectory. Purple circle shows the target and the white region shows the landmark for a range and bearing observation model. a) In RHC-based method replanning is triggered at every step. However, these methods for *stochastic* systems fail to reach the goal after K number of steps, and require heuristic tweaks to work; b) however, in T-LQG, for this example, planning only happens once and the resulting feedback policy is executed for the entire horizon, reaching the goal state after K steps.

In this paper, we overcome this hurdle by providing a coupled design of trajectory and estimator aiming for the best estimation performance using the underlying trajectory of the LQG controller as the optimization variable, while keeping the design of controller separate from the design of the estimator. This simplifies the belief space planning to an optimization problem that can be solved by a general NonLinear Programming (NLP) solver aimed for the design of the nominal trajectory with the best nominal estimation performance. In addition, the design of the controller is performed utilizing the "separation principle" [1], [12]. We term this method as the Trajectory-optimized LQG (T-LQG). Essentially, we use the separation principle and the structure of the LQG method to pose an optimization problem on sequence of control actions parameterizing LQG polices, rather than optimizing over the general policy space. This method reduces the dimension of the underlying state in the optimization problem from $n + n^2$ (Gaussian belief dimension) to n (state dimension), breaking the computational burden of belief space planning problems. The computational complexity of our method is $O(Kn^3)$, where K is the planning horizon and n is the state dimension, which is lower than any other Gaussian belief space planning method in the space of feedback policies, i.e., the optimization is equivalent to doing an optimization over LQG parameterized policies.

*This material is based upon work partially supported by the U.S. Army Research Office under Contract No. W911NF-15-1-0279, the NSF under Science & Technology Center Grant CCF-0939370, and NPRP 6-784-2-329 from the Qatar National Research Fund, a member of Qatar Foundation.

[1]M. Rafieisakhaei and P. R. Kumar are with the Department of Electrical and Computer Engineering, and [2]S. Chakravorty are with the Department of Aerospace Engineering, Texas A&M University, College Station, Texas, 77840 USA. {mrafieis, schakrav, prk@tamu.edu}

We utilize the separation principle to separate the design of estimator and the controller. Then—assuming the existence of a linear controller to stabilize the system around a given nominal trajectory—we prove that under a first-order approximation, the stochastic control objective is dominated by the nominal part of the cost function. Moreover, over the given nominal trajectory, the nominal performance of the estimator is given by the dynamic Riccati equations independent from the actual observations and the controller form. The original problem is reformulated and reduced to a deterministic problem over the state space by choosing the underlying nominal trajectory as the optimization variable, aiming for the best estimation performance over that trajectory. The key observation is: fixing the feedback policy to an linear policy and designing an LQG policy for a linearized system around a nominal trajectory offers the solution of an optimal estimation and control performance along that specific trajectory.

For a fixed linearization trajectory, LQG gives the best estimator and controller to track that nominal trajectory. Our method uses the nominal trajectory itself as an optimization variable in order to obtain the best trajectory, and, subsequently, the best estimator and controller to follow that trajectory. The central difference between this method and the MLO method of [7], [8] is the utilization of the separation principle, which maintains a separate design of the controller to keep the state around the nominal trajectory. Otherwise, the state deviation from the nominal trajectory keeps growing and the assumptions of the nominal trajectory (of control and subsequent state and observations) collapse, reducing the approach to a heuristic design that requires replanning at every time step as in [7], [8], [13] (see Fig. 1). On the contrary, in our approach, the controller keeps the state around the nominal trajectory, and therefore, the nominal estimation performance remains valid, thereby obviating the need to a constant replanning (see Sec. V). Also, using a high-dimensional controller such as a belief-LQR over an $(n + n^2)$-dimensional space as in [7] or [5] and [6] entails disregarding the separation principle by coupling the controller design with the design of the estimator. Using the separation principle also enables us to pose the problem as a standard NLP in n-dimensional space, rather than using a dynamic programming mechanism (which involves calculations in an $(n + n^2)$-dimensional space to solve the coupled equations of the belief estimation and the controller design, as in [5], [6]).

Finally, when the accumulated linearization error (or other errors) increases above a tolerable threshold during the execution, replanning is triggered. This is also a merit of posing the planning problem as a standard NLP with low dimension: replanning for a long horizon becomes possible in online applications. Moreover, it enables the use of off the shelf state-of-the-art optimization software and tools.

Unlike point-based POMDP solvers [14]–[16], in T-LQG the time-horizon is a linear factor in the computational complexity, rather than a factor in the exponent—the curse of history. This means that T-LQG is capable of solving belief space problems on a considerably larger scale. Indeed, current point-based solvers cannot scale to the continuous state, action and observation spaces that are considered in this paper.

In Section II, we define the general problem, and in sections III and IV we explain our method. In Section V, we provide a theoretical analysis on the validity of our method and provide sufficient conditions under which our solution is a valid approximation of the original problem, and provide an extensive theoretical comparison between the state-of-the-art belief space planning methods in section VI. A more detailed version of this paper exists as a technical report [17].

II. General Problem

The general belief space planning problem is formulated as a stochastic control problem in the space of feedback policies. In this section, we define the basic elements of the problem, including system equations and belief dynamics.

System equations: We denote the state, control and observation vectors by $\mathbf{x} \in \mathbb{X} \subset \mathbb{R}^{n_x}$, $\mathbf{u} \in \mathbb{U} \subset \mathbb{R}^{n_u}$, and $\mathbf{z} \in \mathbb{Z} \subset \mathbb{R}^{n_z}$, respectively. The motion $f : \mathbb{X} \times \mathbb{U} \times \mathbb{R}^{n_x} \to \mathbb{X}$, and observation $h : \mathbb{X} \to \mathbb{Z}$ processes are defined as:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, \boldsymbol{\omega}_t), \quad \boldsymbol{\omega}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{\omega}_t}) \tag{1a}$$

$$\mathbf{z}_t = h(\mathbf{x}_t, \boldsymbol{\nu}_t), \qquad \boldsymbol{\nu}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{\nu}_t}) \tag{1b}$$

where $\{\boldsymbol{\omega}_t\}$ and $\{\boldsymbol{\nu}_t\}$ are zero mean independent, identically distributed (i.i.d.) mutually independent random sequences.

Belief (information state): The conditional distribution of $\mathbf{x}_t$ given the data history up to time t, is called the belief $b_t : \mathbb{X} \times \mathbb{Z}^t \times \mathbb{U}^t \to \mathbb{R}$. It is defined as $b_t(\mathbf{x}, \mathbf{z}_{0:t}, \mathbf{u}_{0:t-1}, b_0) := p_{\mathbf{X}_t|\mathbf{Z}_{0:t};\mathbf{U}_{0:t-1}}(\mathbf{x}|\mathbf{z}_{0:t}; \mathbf{u}_{0:t-1}; b_0)$, and denoted by b_t in this paper. We use a Gaussian representation of belief in belief space, $\mathbb{B}$, [18]. The belief dynamics follow Bayesian update equations, summarized as a function $\tau : \mathbb{B} \times \mathbb{U} \times \mathbb{Z} \to \mathbb{B}$, where $b_{t+1} = \tau(b_t, \mathbf{u}_t, \mathbf{z}_{t+1})$ [1], [12].

Problem 1: **Stochastic Control Problem** Given an initial belief state b_0, solve for the optimal policy as follows:

$$\min_{\pi} \; J^{\pi} := \mathbb{E}[\sum_{t=0}^{K-1} c_t^{\pi}(b_t, \mathbf{u}_t) + c_K^{\pi}(b_K)]$$
$$s.t. \; b_{t+1} = \tau(b_t, \mathbf{u}_t, \mathbf{z}_{t+1}), \tag{2}$$

where the optimization is over feasible policies, and $\pi := \{\pi_0, \cdots, \pi_t\}$ where $\pi_t : \mathbb{B} \to \mathbb{U}$ specifies an action given the belief, $\mathbf{u}_t = \pi_t(\mathbf{b}_t)$. Moreover, $c_t^{\pi}(\cdot, \cdot) : \mathbb{B} \times \mathbb{U} \to \mathbb{R}$ is the one-step cost function, $c_K^{\pi}(\cdot) : \mathbb{B} \to \mathbb{R}$ denotes the terminal cost, and the expectation is taken over all randomness.

III. Belief Space Planning Method: T-LQG

We provide details of our design for the planning problem.

Parameterized Trajectories (p–$traj$): Using the noiseless equation of (1a) we parametrize the possible feasible propagated trajectories of the initial estimate, $\hat{\mathbf{x}}_0$, given a set of unknown control inputs $\{\mathbf{u}_t^p\}_{t=0}^{K-1}$, as:

$$\mathbf{x}_{t+1}^p = f(\mathbf{x}_t^p, \mathbf{u}_t^p, \mathbf{0}), \quad 0 \leq t \leq K-1$$

where $\mathbf{x}_0^p = \hat{\mathbf{x}}_0$. The trajectory $\{\mathbf{x}_t^p\}_{t=0}^{K}$ changes by changing the underlying control inputs $\{\mathbf{u}_t^p\}_{t=0}^{K-1}$. We will refer to the parametrized trajectory of $\{\mathbf{x}_t^p\}_{t=0}^{K}$, $\{\mathbf{u}_t^p\}_{t=0}^{K-1}$ as the p–$traj$.

Linearization of the system equations: We linearize the nonlinear motion and observation models of equation (1) about the parametrized trajectory $p\text{–}traj$:

$$\tilde{\mathbf{x}}_{t+1}=\mathbf{A}_t^p(\mathbf{x}_t^p,\mathbf{u}_t^p)\tilde{\mathbf{x}}_t+\mathbf{B}_t^p(\mathbf{x}_t^p,\mathbf{u}_t^p)\tilde{\mathbf{u}}_t+\mathbf{G}_t^p(\mathbf{x}_t^p,\mathbf{u}_t^p)\boldsymbol{\omega}_t \quad (3a)$$

$$\tilde{\mathbf{z}}_t=\mathbf{H}_t^p(\mathbf{x}_t^p)\tilde{\mathbf{x}}_t+\mathbf{M}_t^p(\mathbf{x}_t^p)\boldsymbol{\nu}_t, \quad (3b)$$

where $\mathbf{A}_t^p(\mathbf{x}_t^p,\mathbf{u}_t^p) = \partial f(\mathbf{x},\mathbf{u},\boldsymbol{\omega})/\partial\mathbf{x}|_{\mathbf{x}_t^p,\mathbf{u}_t^p,\mathbf{0}}$, $\mathbf{B}_t^p(\mathbf{x}_t^p,\mathbf{u}_t^p) = \partial f(\mathbf{x},\mathbf{u},\boldsymbol{\omega})/\partial\mathbf{u}|_{\mathbf{x}_t^p,\mathbf{u}_t^p,\mathbf{0}}$, $\mathbf{G}_t^p(\mathbf{x}_t^p,\mathbf{u}_t^p) = \partial f(\mathbf{x},\mathbf{u}_t,\boldsymbol{\omega})/\partial\boldsymbol{\omega}|_{\mathbf{x}_t^p,\mathbf{u}_t^p,\mathbf{0}}$, $\mathbf{H}_t^p(\mathbf{x}_t^p) = \partial h(\mathbf{x},\boldsymbol{\nu})/\partial\mathbf{x}|_{\mathbf{x}_t^p,\mathbf{0}}$ and $\mathbf{M}_t^p(\mathbf{x}_t^p) = \partial h(\mathbf{x},\boldsymbol{\nu})/\partial\boldsymbol{\nu}|_{\mathbf{x}_t^p,\mathbf{0}}$. Moreover, let $\tilde{\mathbf{x}}_t := \mathbf{x}_t - \mathbf{x}_t^p$, $\tilde{\mathbf{u}}_t := \mathbf{u}_t - \mathbf{u}_t^p$ and $\tilde{\mathbf{z}}_t := \mathbf{z}_t - \mathbf{z}_t^p$ define the state, control and observation errors, respectively, and let $\mathbf{z}_t^p = h(\mathbf{x}_t^p,\mathbf{0})$. As the control inputs change, the underlying trajectory they represent, $\{\mathbf{u}_t^p\}_{t=0}^{K-1}$ and $\{\mathbf{x}_t^p\}_{t=0}^{K}$, also change, and therefore the Jacobian matrices change.

Minimization over policies: In the cost function of problem 1, the minimization is over all policies in the feasible policy space. We assume that the search is over linear feedback policies (LQG class), which is a valid assumption for locally controlling a linearized model around a nominal trajectory.

Separation principle: It can be shown that the minimization of the stochastic cost function for a linear Gaussian system in terms of state error $\mathbf{x}_t - \mathbf{x}_t^p$ is equivalent to two separate minimizations in terms of the estimation error $\mathbf{x}_t - \hat{\mathbf{x}}_t$ and the controller error $\hat{\mathbf{x}}_t - \mathbf{x}_t^p$, where $\mathbf{x}_t - \mathbf{x}_t^p = \mathbf{x}_t - \hat{\mathbf{x}}_t + \hat{\mathbf{x}}_t - \mathbf{x}_t^p$ [12]. As a result, the design of a stochastic controller with partially observed states reformulates to two separate designs of an estimator and a controller assuming full state observation. In the LQG case, the estimator is a KF and the controller is an LQR controller.

Estimation cost: The optimization objective is quadratic:

$$J := \mathbb{E}[\sum_{t=1}^{K} \check{\mathbf{x}}_t^T \mathbf{W}_t^x \check{\mathbf{x}}_t + \tilde{\mathbf{u}}_{t-1}^T \mathbf{W}_t^u \tilde{\mathbf{u}}_{t-1}], \quad (4)$$

where $\check{\mathbf{x}}_t := \mathbf{x}_t - \hat{\mathbf{x}}_t$ is the estimator error (KF error), and $\mathbf{W}_t^x, \mathbf{W}_t^u \succeq 0$ are two positive-definite weight matrices, and $\mathbf{W}_t^x$ is symmetric, thereby, it has a square root. This cost function can be rewritten as a function of the belief trajectory [17]. However, we show in Section V under a first-order approximation, J is dominated by the part of the cost function that depends only on the underlying nominal trajectory ($p\text{–}traj$ here). Thus, we approximate J with J^p:

$$J^p := \sum_{t=1}^{K} (\mathbb{E}[\tilde{\mathbf{x}}_t^T \mathbf{W}_t^x \tilde{\mathbf{x}}_t] + (\mathbf{u}_{t-1}^p)^T \mathbf{W}_t^u \mathbf{u}_{t-1}^p), \quad (5)$$

where the first term is a function of the nominal belief which is discussed next.

Nominal evolution of covariance: Define $\mathbf{P}_{b_t^p}^+(\mathbf{x}_{0:t}^p,\mathbf{u}_{0:t-1}^p) := \mathbb{E}[(\mathbf{x}_t - \mathbf{x}_t^p)(\mathbf{x}_t - \mathbf{x}_t^p)^T]$ to be the covariance of the nominal belief along the $p\text{–}traj$. Therefore, the first term of the cost function (5), can be rewritten as $\sum_{t=1}^{K} \mathrm{tr}(\mathbf{W}_t \mathbf{P}_{b_t^p}^+ \mathbf{W}_t^T)$, where $\mathbf{W}_t := (\mathbf{W}_t^x)^{1/2}$. For a presumptive parametrized trajectory $p\text{–}traj$, the evolution of $\mathbf{P}_{b_t^p}^+$ is given by the recursive Riccati equations (6a)-(6d) bellow, independent from the observations but as a function of the trajectory itself. This also provides the nominal performance of the estimator along that trajectory. The dependencies on the $p\text{–}traj$ inside the parenthesis is dropped for simplicity in the following.

Problem 2: **Planning Problem** Given an initial belief $b_0 \in \mathbb{B}$, a goal region represented as an ℓ_2-norm ball, $B_{r_g}(\mathbf{x}_g)$, of radius r_g around a goal state $\mathbf{x}_g \in \mathbb{X}$, and a planning horizon of $K > 0$, we define the following problem:

$$\min_{\mathbf{u}_{0:K-1}^p} \sum_{t=1}^{K} [\mathrm{tr}(\mathbf{W}_t \mathbf{P}_{b_t^p}^+ \mathbf{W}_t^T) + (\mathbf{u}_{t-1}^p)^T \mathbf{W}_t^u \mathbf{u}_{t-1}^p]$$

$$s.t. \quad \mathbf{P}_{b_t^p}^- = \mathbf{A}_{t-1}^p \mathbf{P}_{b_{t-1}^p}^+ (\mathbf{A}_{t-1}^p)^T + \mathbf{G}_{t-1}^p \boldsymbol{\Sigma}_{\boldsymbol{\omega}_{t-1}} (\mathbf{G}_{t-1})^T \quad (6a)$$

$$\mathbf{S}_{b_t^p} = \mathbf{H}_t^p \mathbf{P}_{b_t^p}^- (\mathbf{H}_t^p)^T + \mathbf{M}_t^p \boldsymbol{\Sigma}_{\boldsymbol{\nu}_t} (\mathbf{M}_t^p)^T \quad (6b)$$

$$\mathbf{P}_{b_t^p}^+ = (\mathbf{I} - \mathbf{P}_{b_t^p}^- (\mathbf{H}_t^p)^T \mathbf{S}_{b_t^p}^{-1} \mathbf{H}_t^p) \mathbf{P}_{b_t^p}^- \quad (6c)$$

$$\mathbf{P}_{b_0^p}^+ = \boldsymbol{\Sigma}_{\mathbf{x}_0} \quad (6d)$$

$$\mathbf{x}_0^p = \mathbb{E}_{\mathbf{x}}[b_0(\mathbf{x})] \quad (6e)$$

$$\mathbf{x}_{t+1}^p = f(\mathbf{x}_t^p, \mathbf{u}_t^p, 0) \;\; 0 \le t \le K-1 \quad (6f)$$

$$\|\mathbf{x}_K^p - \mathbf{x}_g\|_2 < r_g \quad (6g)$$

$$\|\mathbf{u}_t^p\|_2 \le r_u, \;\; 1 \le t \le K, \quad (6h)$$

where equations (6a)-(6c) are regarded as one constraint at each time step, and are used to calculate the first term of the objective at that time step, equations (6d) and (6e) represent the initial conditions, equation (6f) defines the state propagation (and relates the optimization variables to the state trajectory), equation (6g) constrains the terminal state to $B_{r_g}(\mathbf{x}_g)$, equation (6h) accounts for the saturation constraints for $r_u > 0$. Moreover, the first term of the objective aims for minimizing the estimation uncertainty, whereas the second term penalizes the control effort. This problem is an optimization in the space of control actions with every other variable, such as the covariances, as a function of those controls.

Optimized Trajectory, $o\text{–}traj$: We will denote the resulting optimized $p\text{–}traj$ of problem 2 with $\{\mathbf{x}_t^o\}_{t=0}^{K}$, $\{\mathbf{u}_t^o\}_{t=0}^{K-1}$ and refer to it as the $o\text{–}traj$.

Feedback control: The resulting trajectory from the optimization problem is optimized in terms of estimation performance. Now, using the separation principle, the LQR controller is designed to follow the $o\text{–}traj$. Therefore, the LQR cost is designed for the controller error $\hat{\mathbf{x}}_t - \mathbf{x}_t^o$. The resulting control policy is a linear feedback policy, and $\mathbf{u}_t = -\mathbf{L}_t(\hat{\mathbf{x}}_t - \mathbf{x}_t^o) + \mathbf{u}_t^o$, where the feedback gain $\mathbf{L}_t$ is obtained using the backward Riccati recursions, and the evolution of $\hat{\mathbf{x}}_t$ is obtained from the KF equations using the actual observations in the execution. The details of these equations are common to any LQG problem [17]. Note, in a Receding Horizon Control (RHC) implementation as in [8], $\mathbf{u}_t$ would only consist of $\mathbf{u}_t^o$, and to get the corrections from the output, the planning problem is solved again at each time step from the current belief, Multiplying the whole effort of the algorithm (optimization problem plus convergence to an optimized trajectory) to a factor of K.

Replanning during execution: In a stochastic system, even with a closed loop control strategy, after a finite number of execution steps, the estimate may deviate from the planned

trajectory. This happens due to the accumulation of errors resulting from the un-modeled dynamics or forces, noise, and nonlinearities. In such a situation, the planned policy becomes irrelevant and a new policy is needed to drive the agent toward the predefined goals. In order to overcome the problem, we track the nominal belief. Then, based on the Kullback-Leibler divergence between the nominal and current belief, we define a symmetric distance (average of unsymmetrical KL-divergences). Once such a distance is greater than a predefined threshold $d_{th} > 0$, a deviation is detected, the planning module is initialized with the current belief, and all planning procedures are performed again. The expanded details of this derivation is discussed in [17].

IV. Non-Convex State Constraints

Barrier functions are used for non-convex state constraints.

Polygonal obstacles approximated by ellipsoids: Given a set of vertices that constitute a polygonal obstacle, we find the Minimum Volume Enclosing Ellipsoid (MVEE) and obtain its parameters [19]. Particularly, for obstacle i, the barrier function includes a Gaussian-like function, where the argument of the exponential is the MVEE, which can be disambiguated with its center $\mathbf{c}^i$ and a positive definite matrix $\mathbf{E}^i$ that determines the rotation and axes of the ellipsoid. Moreover, we add several number of inverse functions that tend to infinity along the major and minor axes of the ellipsoid. So, the overall function acts as a barrier to prevent the trajectory from entering the region enclosed by the ellipsoid. Note for non-polygonal obstacles, one can find the MVEE, and the algorithm works independently from this fact. Thus, given the ellipsoid parameters $\mathcal{C} := (\mathbf{c}^1, \mathbf{c}^2, \cdots, \mathbf{c}^{n_b}) \in \mathbb{R}^{n_x \times n_b}$ and $\mathcal{E} := (\mathbf{E}^1, \cdots, \mathbf{E}^{n_b}) \in \mathbb{R}^{n_x^2 \times n_b}$, the Obstacle Barrier Function (OBF) is constructed as:

$$\Phi^{(\mathcal{E},\mathcal{C})}(\mathbf{x}) := \sum_{i=1}^{n_b} \Big[M_1 \exp(-[(\mathbf{x}-\mathbf{c}^i)^T \mathbf{E}^i (\mathbf{x}-\mathbf{c}^i)]^q) + M_2 \sum_{\theta=0:\epsilon_m:1} (\|\mathbf{x} - (\theta\zeta^{i,1} + (1-\theta)\zeta^{i,2})\|_2^{-2} + \|\mathbf{x} - (\theta\xi^{i,1} + (1-\theta)\xi^{i,2})\|_2^{-2}) \Big],$$

where $\epsilon_m = 1/m, m \in \mathbb{Z}^+$, $M_1, M_2 \geq 0$, $q \in \mathbb{Z}^+$, and $\zeta^{i,1}$, $\zeta^{i,2}$ and $\xi^{i,1}$, $\xi^{i,2}$ are the endpoints of the major and minor axes of the ellipsoid, respectively. Therefore, the second term in the sum places inverse functions whose values tend to infinity along the axes of the ellipsoid at points formed by convex combination of the two endpoints of each axis. As ϵ_m tends to zero, the entire axes of the ellipsoid become infinite, and, therefore, act as a barrier to any continuous trajectory of states. we define the cost of avoiding obstacles as:

$$\text{cost}_{obst}(\mathbf{x}_{t1}, \mathbf{x}_{t2}) := \int_{\mathbf{x}_{t1}}^{\mathbf{x}_{t2}} \Phi^{(\mathcal{E},\mathcal{C})}(\mathbf{x}') d\mathbf{x}', \tag{7}$$

which is the line integral of the OBF between two given points of the trajectory $\mathbf{x}_{t1}$ and $\mathbf{x}_{t2}$. Therefore, the addition of this cost to the optimization objective, ensures the solver minimizes this cost and keeps the trajectory out of banned regions. Using this equation, we add the running obstacle cost of $\text{cost}_{obst}(\mathbf{x}_{t-1}, \mathbf{x}_t)$ to the optimization objective and use the modified optimization problem to obtain locally optimal solutions in the inter-obstacle feasible space using gradient descent methods [20].

V. Theoretical Considerations

In this section, we derive the conditions upon which our planning strategy is an acceptable approximation of the original problem. We perform a first-order approximation on the propagated errors of the state, control, observation, belief, and cost function around a nominal trajectory that stems from a nominal trajectory of the control actions. We show that under some sufficient conditions, our planning strategy provides an acceptable approximation to the original problem. Due to the lack of space, details of the derivations and proofs are in the technical report [17].

Linearization of process model, observation model, belief dynamics, and cost function: Assuming there exists a nominal trajectory of control actions, $\{\mathbf{u}_t^p\}_{t=0}^{K-1}$, then there exists a corresponding nominal trajectory of states, $\{\mathbf{x}_t^p\}_{t=0}^{K}$ and observations $\{\mathbf{z}_t^p\}_{t=0}^{K-1}$, where $\mathbf{x}_{t+1}^p = f(\mathbf{x}_t^p, \mathbf{u}_t^p, \mathbf{0})$, $\mathbf{z}_t^p = h(\mathbf{x}_t^p)$, and the state, control, and observation error vectors are $\tilde{\mathbf{u}}_t = \mathbf{u}_t - \mathbf{u}_t^p$, $\tilde{\mathbf{x}}_t = \mathbf{x}_t - \mathbf{x}_t^p$ and $\tilde{\mathbf{z}}_t = \mathbf{z}_t - \mathbf{z}_t^p$, respectively. We linearize the state and observation dynamics around the nominal trajectories of state and control. Corresponding to the nominal trajectory of states, there exists a nominal (Gaussian) trajectory of beliefs, $\{b_t^p\}_{t=0}^{K}$, where $\mathbf{x}_t^p = \mathbb{E}[b_t^p]$, and the covariance evolution follows the Riccati equations from a KF. Likewise, we linearize the belief dynamics around the nominal trajectories of belief, control and observation and linearize the cost function around the nominal trajectory of belief and control as $J \approx \sum_{t=0}^{K-1} c_t(b_t^p, \mathbf{u}_t^p) + c_K(b_K^p) + \tilde{J}$, where we assume continuity of the cost function, and $\mathbf{C}_t^b = \partial c_t(b, \mathbf{u})/\partial b|_{b_t^p, \mathbf{u}_t^p}$, $\mathbf{C}_t^u = \partial c_t(b, \mathbf{u})/\partial \mathbf{u}|_{b_t^p, \mathbf{u}_t^p}$, $\mathbf{C}_K^b = \partial c_K(b)/\partial b|_{b_K^p}$, $\tilde{b}_t := b_t - b_t^p$ denotes the belief error, and $\tilde{J} := \mathbb{E}[\sum_{t=0}^{K-1} \mathbf{C}_t^b \tilde{b}_t + \mathbf{C}_t^u \tilde{\mathbf{u}}_t + \mathbf{C}_K^b \tilde{b}_K]$.

Feedback controller: As mentioned before, we assume the search is over linear feedback policies, which is a valid assumption for locally controlling a linearized model around a nominal trajectory. Our design, based on the separation principle, supposes the existence of an LQR controller to track and stabilize the trajectory of states around the nominal designed trajectory. Thus, $\tilde{\mathbf{u}}_t = -\mathbf{L}_t(\hat{\mathbf{x}}_t - \mathbf{x}_t^p) + \mathbf{u}_t^p$.

Lemma 1: **Belief Error Propagation** Given the nominal trajectory of $p\text{–}traj$, and linearization of state, observation, control and belief dynamics around the $p\text{–}traj$, let belief error be $\tilde{\mathbf{b}}_t = \mathbf{b}_t - \mathbf{b}_t^p$ for $t \geq 0$. Then, for a design that follows the separation principle, there exists matrices $\tilde{\mathbf{T}}_t^{\mathbf{x}_0}$, $\tilde{\mathbf{T}}_{s,t}^{\boldsymbol{\omega}}$, and $\tilde{\mathbf{T}}_{s,t}^{\boldsymbol{\nu}}$, such that for $t \geq 0$ the belief error propagation can be written as follows:

$$\tilde{\mathbf{b}}_t = \tilde{\mathbf{T}}_t^{\mathbf{x}_0} \tilde{\mathbf{x}}_0 + \sum_{s=0}^{t-1} \tilde{\mathbf{T}}_{s,t}^{\boldsymbol{\omega}} \boldsymbol{\omega}_s + \sum_{s=1}^{t} \tilde{\mathbf{T}}_{s,t}^{\boldsymbol{\nu}} \boldsymbol{\nu}_s, \tag{8}$$

where $\tilde{\mathbf{x}}_0$, $\boldsymbol{\omega}_s$ and $\boldsymbol{\nu}_s$ denote the initial state error, process and measurement noises, respectively.

Theorem 1: **Cost Function Error** Let cost function error be $\tilde{J} = J - J^p$ for $t \geq 0$. Given that process and observation noises are zero mean i.i.d. and are mutually independent from each other and the initial belief, under a first-order approximation, the expected error in the stochastic cost function is zero, i.e., $\tilde{J} = 0$.

Thus, the error in our cost function is only dependent on the propagated error of the process and observation models' linearizations. In practice, the time horizon is only chosen large enough to find feasible solutions, and the linearization error is neglected. Thus, choosing the underlying linearization trajectory (or equivalently, the control actions corresponding to the nominal trajectory) as the optimization variables, the optimal underlying trajectory can be calculated. Moreover, whenever the accumulated error of the belief approximation under these assumption gets higher than a threshold, the problem restarts and replanning follows, as described previously. It is important to note that the separation principle is the central idea behind the method, since it provides the mechanism to design the controller and estimator separate from each other. The T-LQG method utilizes this theory and the dependence of the LQG on the underlying trajectory to find the LQG controller with the best performance.

VI. Comparison of Methods

In this section, we provide a comparison between T-LQG and other state-of-the-art belief space planning approaches from a methodology and computational complexity perspective. We make occasional references to the following methods: a) LQG-MP [4], b) iLQG-based method [5], c) SELQR [6] d) the method utilizing MLO [7], e) the non-Gaussian Receding Horizon Control (RHC)-based method [10], f) the non-Gaussian observation covariance reduction method [11], g) the covariance-free open-loop optimization problem coupled with RHC implementation [8], and h) the point-based POMDP solvers [9], [14]–[16]. Table I summarizes the key differences between the methods. Due to the lack of space, more discussion points and a detailed explanation of all these are laid out in the technical report [17]. Regarding the Table, we note that:

- We assume the size of vectors $\mathbf{x}, \mathbf{u}$ and $\mathbf{z}$ are all $O(n)$, and K is planning horizon
- n_r is the number of RRT paths generated in [4]
- For the method of [7], n_{tr} is the number of transcription steps in the direct transcription; k is the number of unit vectors pointing in the desired directions to minimize the covariance in; the second computational complexity is valid if the B-LQR is also used, otherwise, the first complexity is more accurate
- N is the number of samples, ϵ is the convergence error
- *Convergence Rate* is the number of calls needed to the oracle to converge using the optimization method
- *DP* is Dynamic Programming
- *Second order* is the general rate of Newton-like methods
- Method of [8] defines an optimization problem with dimension of $O(n_x + n_u)$ whereas T-LQG's problem dimension is $O(n_u)$. Moreover, [8] utilizes an approach similar to [7] with MLO assumption and EKF design; however, [8] utilizes RHC as the final implementation.
- The computational complexity only reflects the calculations of the core problems for belief space planning in each method. For obstacle-avoidance, each method has a different approach, which is out of the scope of this

TABLE I. Comparison of belief space planning methods on important issues.

	Planning as an Optimization	Linearization Trajectory (Exploitable for Optimization)	Planning Observations	Computational Complexity	Convergence Rate
LQG-MP [4]	None	RRT trajectories (No)	—	$O(n_r K n^3)$	—
iLQG-based [5]	DP	Fixed at each iteration (No)	Stochastic observations	$O(Kn^6)$	Second order (line-search tuning)
SELQR [6]	DP	Fixed at each iteration (No)	MLO	$O(Kn^6)$	Second order
MLO [7]	NLP	Predicted mean update (Yes)	MLO	$O(n_{tr}(Kn^3{+}kn^2))$ or $O(n_{tr}(Kn^3{+}kn^2){+}Kn^6)$	SQP rate
Non-Gaussian RHC-Based [10]	Convex	Linear propagation of initial estimate (Yes)	MLO	$O(NK(Kn^3{+}Nn^2))$	$\Omega((N{+}Kn)\log(\frac{1}{\epsilon}))$
Non-Gaussian Obs. Cov. Reduction [11]	Convex	Linear propagation of initial estimate (Yes)	Predicted ensemble of observation particles	$O(Kn^3{+}Nn^2)$	$\Omega(Kn\log(\frac{1}{\epsilon}))$
Cov.-Free RHC [8]	NLP	Predicted mean update (Yes)	MLO	$O(Kn^3)$	Second order
T-LQG	NLP	Nonlinear propagation of initial estimate (Yes)	—	$O(Kn^3)$	Second order

discussion and can be further detailed in a pure motion-planning perspective. The information in table I and the calculations regarding the computational complexity are estimated to the best of our knowledge.

As reflected in the table, a central difference between these methods is the way the system and observation equations are linearized. After linearization of the equations, the corresponding Jacobians become coupled with the trajectory. Therefore, if the underlying linearization trajectory is a sequence of fixed points, the Jacobians become constant matrices for the entire optimization, and the structure of the system models (on which depends many other properties of the system, such as sensitivity of the observations, controllability, reachability, etc.) essentially become fixed, untouchable, and, more importantly, un-exploitable for the optimization purposes. As noted, our method fully exploits this property and finds the best linearization trajectory among the methods. Moreover, no assumptions on observations in our method are made. Most importantly, the computational complexity of T-LQG is the lowest among all, while still providing a feedback policy, a claim none of the other methods can make.

Point-Based POMDP Solvers [9], [14]–[16] The POMDP problem was introduced in 1971 in [9], with an algorithm to obtain the exact optimal solution using the alpha-vectors. The algorithm then evolved into an anytime algorithm in 2003 in [14], introducing the point-based POMDP solvers. This method has been the foundation for the majority of research in the POMDP field [15]. There has been many successes in solving POMDP benchmark problems with low CPU-times. Even the latest advancements in the field, such as [16], suffer from multiple limitations. For instance, the scalability with time-horizon—exponential dependency—seems to be a fundamental limitation that might be difficult to overcome. Ad-hoc solutions to reduce the planning time horizon to local planning (which are much lower than enough for reaching the goal region) and replanning every few steps may not be a feasible solution for practical problems.

Moreover, in T-LQG, by tracking the nominal and true belief during online implementation, whenever the optimality deviation is more than a tolerable threshold, replanning occurs, which may be impractical in long-horizon point-based POMDP solvers. FIRM [21] on the other hand, provides an offline approach to tackle the original POMDP problem by solving the dynamic programming over a graph in the belief space and breaking the curse of history; but, to get closer to optimality, more FIRM nodes need to be sampled offline.

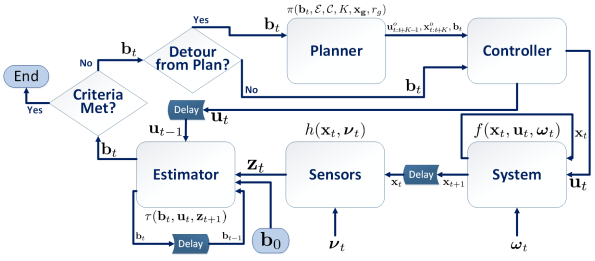


Fig. 2. The overall feedback control loop.

Algorithm 1: T-LQG

Input: Initial belief b_0, Goal region $B_{r_g}(\mathbf{x}_g)$, Planning horizon K, Obstacle parameters $(\mathcal{E}, \mathcal{C})$

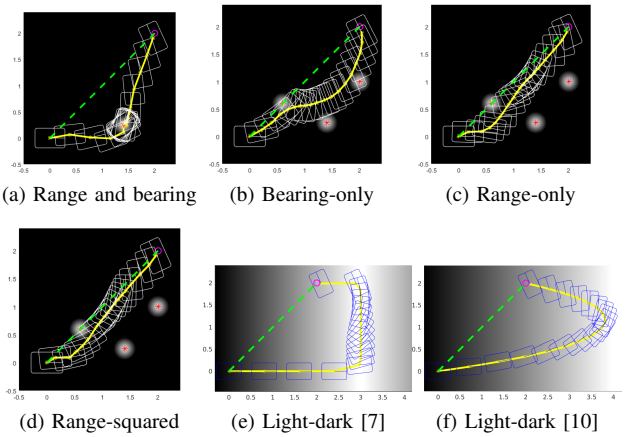
$t \leftarrow 0$;
while $\mathcal{P}(b_t, r_g, \mathbf{x}_g) \leq p_g$ **do**
  if $d(b_t, b_t^o) > d_{th}$ **or** $t == 0$ **or** $t == K$ **then**
    Optimal Trajectory: $\{\mathbf{u}^o_{0:K-1}, \mathbf{x}^o_{0:K}\} \leftarrow \pi(b_0, \mathcal{E}, \mathcal{C}, K, \mathbf{x_g})$;
    $t \leftarrow 0$;
  end
  else
    Policy Function: $\hat{\mathbf{x}}_t \leftarrow \mathbb{E}[b_t]$, $\mathbf{u}_t \leftarrow -\mathbf{L}_t(\hat{\mathbf{x}}_t - \mathbf{x}^o_t) + \mathbf{u}^o_t$;
    Execution: $\mathbf{x}_{t+1} \leftarrow f(\mathbf{x}_t, \mathbf{u}_t, \boldsymbol{\omega}_t)$;
    Perception: $\mathbf{z}_{t+1} \leftarrow h(\mathbf{x}_{t+1}, \boldsymbol{\nu}_{t+1})$;
    Estimation: $b_{t+1} \leftarrow \tau(b_t, \mathbf{u}_t, \mathbf{z}_{t+1})$;
    $t \leftarrow t + 1$;
  end
end

VII. Simulation Results

In this section, we provide our simulation results to show the performance of T-LQG. Our simulations are performed in MATLAB 2016a with a 2.90 GHz CORE i7 machine with dual core technology and 8 GB of RAM. We use the MATLAB's fmincon solver to solve the NLP problem. First, we provide the overall algorithm and the overall control loop. Then, we investigate several situations in which the environment is obstacle-free. We perform 6 simulations for a KUKA youBot base model, with 6 different observation models including models adapted from the literature. In the second scenario, we perform a comparison between the performance of T-LQG and an RHC-based method [8]. Then, we perform a simulation in a complex environment with many obstacles. We conduct this scenario for two different initial trajectories and compare the results. In each scenario, we show the initial trajectory used to initialize the optimization problem along with the optimized output trajectory.

Implementation: The overall control loop is shown in Fig. 2, and the overall T-LQG algorithm is reflected in Algorithm 1. As it is seen in Fig. 2 and Alg. 1, the planning problem starts with the supply of an initial belief and ends whenever the probability of reaching the goal region is greater than a predefined threshold $p_g > 0$. The planner π is fed the initial belief b_0, the obstacle parameters $(\mathcal{E}, \mathcal{C})$, planning horizon K, a goal region $B_{r_g}(\mathbf{x}_g)$ and other parameters, such as system equations. The resultant planned trajectory is provided to the controller, whose output is the policy function. The policy is executed, a new observation is perceived, and a new belief is obtained. If the distance between the updated belief and the nominal belief $d(b_t, b_t^o) > d_{th}$ is greater than a threshold, or the policy execution is finished but the criteria is not met, the planning algorithm restarts.

Obstacle-free environment: Let us use the kinematics equations of KUKA youBot base as described in [22]. Particularly, the state vector can be denoted by a 3D vector,



(a) Range and bearing (b) Bearing-only (c) Range-only

(d) Range-squared (e) Light-dark [7] (f) Light-dark [10]

Fig. 3. Simulation results for obstacle-free situation with different observation models. The information is color-coded. Lighter means less noisy observations. The dashed green line represents the initial trajectory; the solid yellow line shows the optimized trajectory. In all cases, $\hat{\mathbf{x}}_0 = (0, 0, 0)$, $\mathbf{x}_g = (2, 2, 2)$, and $r_g = 0.1$.

TABLE II. Comparison T-LQG with method of [8].

	Final Distance from Goal (after 16 steps)	Total Time (MATLAB)
RHC-based [8]	4.09 (m)	352 (seconds)
T-LQG	0.45 (m)	20 (seconds)

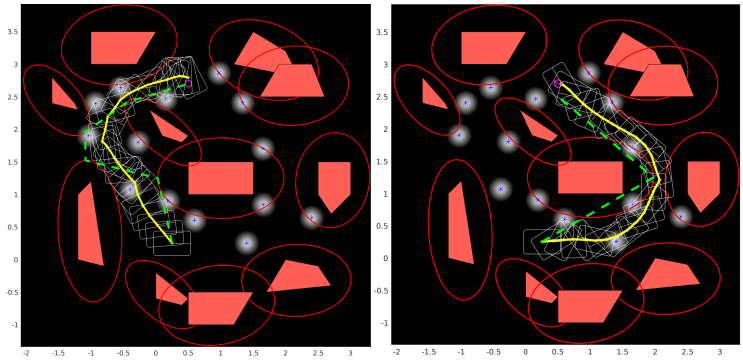
$\mathbf{x} = [\mathbf{x}_\mathbf{x}, \mathbf{x}_\mathbf{y}, \mathbf{x}_\theta]^T$, which describes the position and heading of the robot base, and $\mathbf{x} \in SO(2)$. The control consists of the velocities of the four wheels. It can be shown that the discrete motion model can be written as $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, \boldsymbol{\omega}_t) = \mathbf{x}_t + \mathbf{B}\mathbf{u}_t dt + \mathbf{G}\boldsymbol{\omega}_t\sqrt{dt}$, where $\mathbf{B}$ and $\mathbf{G}$ are appropriate constant matrices, and dt is the time-discretization period. The results depicted in Fig. 3 are for different observation models; including the range and bearing; bearing-only and range-only observation from landmarks in cases (a)-(c). In case (d), the observation function is changed to the square of the range function. Finally, in cases (e) and (f), the light-dark models of the papers [7] and [10] are adopted. In both cases, the observation functions are linear, and the covariance of the observation noise is state dependent. It is a quadratic function with a minimum at 3 in case (e) and a hyperbolic function with a minimum at $+\infty$ in (f). More details of these functions can be found in the corresponding papers and [17]. Finally as noted in all cases, the optimization is initialized with the trivial straight-line, which is reflected in the figures with the dashed green line, whereas the optimal trajectories are depicted with solid lines.

Comparison: Depicted in Fig. 1, is the results of T-LQG and our implementation of an RHC-based method which uses the MLO assumption [8] for a youbot in a landmark-based observation model with range and bearing information. Table II shows the comparison of the costs after $K = 16$ steps of execution. In our method, the optimization problem is solved only once and then the resulting feedback policy is executed without the need to re-plan. Whereas, in the method of [8], at every time step, replanning is triggered in order to close the feedback loop. However, this means the optimization problem is solved 16 times, and yet the agent does not reach to the goal after 16 steps. It is worth mentioning that although both methods have same order of complexity for the optimization problem, the fact that in [8] the optimization is solved for convergence 16 times more (and possibly much more is need to reach to goal), in T-LQG it is only solved once (for this example), and thus, the overall execution time of T-LQG is $O(K)$ times lower, with much reliable plans.

Complex environment: Next, we perform a simulation in an environment full of obstacles for the youBot, with range and bearing observations from several landmarks. Inspired by [23], we model the robot with a configuration of a set of points that represent the balls' centers that cover the body of the robot. As it is seen in Fig. 4. we have initialed the optimization problem with two different initial trajectories obtained using a modified viability graph algorithm (shown with the green dashed lines). It should be noted that there is nothing particular about the initialization algorithm and methods—a planner such as RRT can be used as well, as long as the initialization trajectory is semi-feasible in that it does not pass through the infeasible local minima of the barrier functions. As it seen in this figure, the planning horizon is large (26 steps in case (a) and 25 steps in case (b)), which shows the scalability of T-LQG. The results show that the optimized trajectory (reflected with solid lines) avoids entering the banned regions bordered by the ellipsoids, so that the robot itself avoids colliding with the obstacles. Moreover, the locally optimal trajectory gets closer to the information sources and thereby obtains the best predicted estimation performance. In this scenario, by comparing the cost of the two optimize trajectories, the better of the two (trajectory in Fig. 4b) is chosen as the plan for execution.

VIII. Conclusion

In this paper, we simplified the solution of the belief space planning problem by proposing a scalable method that is backed by theoretical analysis supported by the control literature. Particularly, we proposed a deterministic optimal control problem that can be solved by an NLP solver with $O(Kn^3)$ computational complexity. The goal of Trajectory-optimized LQG is to find an LQG policy with the best nominal performance. T-LQG achieves this by finding the best underlying linearization trajectory for a nonlinear system with a nonlinear observation model, utilizing the trajectory-dependent covariance evolution of the Kalman filter given by the dynamic Riccati equations. We could do this by the proper usage of the separation principle that provided us with an LQR controller for a linearized system along that nominal trajectory. We proved that the accumulated error that is resulted by our calculations is deterministic under a first-order approximation and only depends on the linearization error. This can be overcome by either increasing the linearization points or by replanning whenever the deviation from the planned trajectory is higher than a predefined tolerance. We also extended the method to non-convex environments by adding a cost function to avoid collision with the obstacles. Finally, we performed simulations for a common robotic system with several observation functions in obstacle-free environments, and complex narrow passages with obstacles.



(a) (b)

Fig. 4. Simulation results for two different initializations with obstacles. The obstacles are the red solid polygons; the ellipses show the inflated regions around them avoided by the configuration of points that represent the robot (they are also the argument of the Gaussian function in the obstacle cost). In all cases, $\hat{\mathbf{x}}_0 = (0.25, 0.25, 0)$, $\mathbf{x}_g = (0.5, 2.7, 2)$, and $r_g = 0.1$. The optimized trajectory in case (b) has a lower overall cost.

In conclusion, while T-LQG and the MLO method of [7] address a similar optimization problem, their theoretical approach is vastly different:

- where MLO uses a heuristic approach, T-LQG uses the separation principle;
- MLO does not have a controller in the design, whereas T-LQG does;
- MLO uses assumptions on the observations to reach the optimization problem, while in T-LQG, assumptions on observations are inconsequential;
- MLO designs a belief-LQR, but T-LQG only requires an LQR on the state;
- MLO starts with an EKF design and linearizes the system equations around the mean update, while T-LQG starts with linearizing around a nominal trajectory and uses the KF and separation principle to obtain the nominal performance around that trajectory;
- MLO assumes from the beginning that process noise does not exist, and, ultimately, assumes observation noise does not exist either, but in T-LQG, neither of these assumptions exist; and
- while the computational complexity for MLO is $O(n_{tr}(Kn^3 + kn^2))$ or $O(n_{tr}(\ Kn^3 + kn^2) + Kn^6)$, T-LQG minimizes the complexity to $O(Kn^3)$.

Our future works will extend the theory and test the validity of our results for more complex situations.

REFERENCES

[1] P. R. Kumar and P. P. Varaiya, *Stochastic Systems: Estimation, Identification, and Adaptive Control*. Englewood Cliffs, NJ: Prentice-Hall, 1986.

[2] D. Bertsekas, *Dynamic Programming and Stochastic Control*. Academic Press, 1976.

[3] P. Kumar *et al.*, "Control: a perspective," *Automatica*, vol. 50, no. 1, pp. 3–43, 2014.

[4] J. Van Den Berg, P. Abbeel, and K. Goldberg, "Lqg-mp: Optimized path planning for robots with motion uncertainty and imperfect state information," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 895–913, 2011.

[5] J. Van Den Berg, S. Patil, and R. Alterovitz, "Motion planning under uncertainty using iterative local optimization in belief space," *The International Journal of Robotics Research*, vol. 31, no. 11, pp. 1263–1278, 2012.

[6] W. Sun, J. van den Berg, and R. Alterovitz, "Stochastic extended lqr for optimization-based motion planning under uncertainty," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 437–447, 2016.

[7] R. Platt, R. Tedrake, L. Kaelbling, and T. Lozano-Perez, "Belief space planning assuming maximum likelihood observatoins," in *Proceedings of Robotics: Science and Systems (RSS)*, June 2010.

[8] S. Patil, G. Kahn, M. Laskey, J. Schulman, K. Goldberg, and P. Abbeel, "Scaling up gaussian belief space planning through covariance-free trajectory optimization and automatic differentiation," in *Algorithmic Foundations of Robotics XI*. Springer, 2015, pp. 515–533.

[9] E. J. Sondik, "The optimal control of partially observable markov processes," *PhD thesis, Stanford University*, 1971.

[10] R. Platt, "Convex receding horizon control in non-gaussian belief space," in *Algorithmic Foundations of Robotics X*. Springer, 2013, pp. 443–458.

[11] M. Rafieisakhaei, A. Tamjidi, S. Chakravorty, and P. Kumar, "Feedback motion planning under non-gaussian uncertainty and non-convex state constraints," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 4238–4244.

[12] D. Bertsekas, *Dynamic Programming and Optimal Control: 3rd Ed.* Athena Scientific, 2007.

[13] W. Sun, S. Patil, and R. Alterovitz, "High-frequency replanning under uncertainty using parallel sampling-based motion planning," *IEEE Transactions on Robotics*, vol. 31, no. 1, pp. 104–116, 2015.

[14] J. Pineau, G. Gordon, and S. Thrun, "Point-based value iteration: An anytime algorithm for POMDPs," in *International Joint Conference on Artificial Intelligence*, 2003, pp. 1025–1032.

[15] G. Shani, J. Pineau, and R. Kaplow, "A survey of point-based pomdp solvers," *Autonomous Agents and Multi-Agent Systems*, vol. 27, pp. 1–51, 2013.

[16] K. M. Seiler, H. Kurniawati, and S. P. Singh, "An online and approximate solver for pomdps with continuous action space," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 2290–2297.

[17] M. Rafieisakhaei, S. Chakravorty, and P. Kumar, "Belief space planning simplified: Trajectory-optimized lqg (t-lqg)(extended report)," *arXiv preprint arXiv:1608.03013*, 2016.

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

[19] N. Moshtagh, "Minimum volume enclosing ellipsoid," *Convex Optimization*, vol. 111, p. 112, 2005.

[20] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[21] A. Agha-mohammadi, S. Chakravorty, and N. Amato, "Firm: Sampling-based feedback motion planning under motion uncertainty and imperfect measurements," *International Journal of Robotics Research*, no. 2, 2014.

[22] zakharov, "zakharov youbot model," *GitHub*, 2011. [Online]. Available: "https://github.com/zakharov/youbot_model/wiki/KUKA-youBot-kinematics,-dynamics-and-3D-model"

[23] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "Chomp: Covariant hamiltonian optimization for motion planning," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.