CS-BRM: A Probabilistic RoadMap for Consistent Belief Space Planning with Reachability Guarantees

Dongliang Zheng, Jack Ridderhof, Zhiyuan Zhang, Panagiotis Tsiotras
Georgia Institute of Technology, Atlanta, GA 30332-0150, USA
{dzheng, jridderhof3, nickzhang, tsiotras}@gatech.edu

Ali-akbar Agha-mohammadi
Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109, USA
aliagha@jpl.nasa.gov

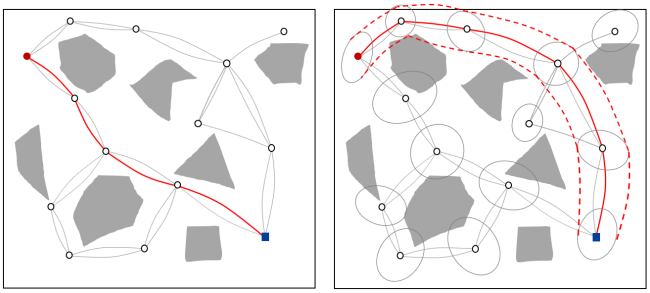
***Abstract*—A new belief space planning algorithm, called covariance steering Belief RoadMap (CS-BRM), is introduced, analyzed, and numerically and experimentally tested. CS-BRM is a multi-query algorithm for motion planning for dynamical systems under simultaneous motion and observation uncertainties. CS-BRM extends the probabilistic roadmap (PRM) approach to belief spaces based on the recently developed theory of covariance steering (CS) that enables guaranteed satisfaction of terminal belief constraints in finite time. The nodes in the CS-BRM are sampled in the belief space and represent distributions of the system states. A covariance steering controller steers the system from one BRM node to another, thus acting as an edge controller of the corresponding belief graph that ensures belief constraint satisfaction. After the edge controller is computed, a specific edge cost is assigned to that edge. The CS-BRM algorithm allows the sampling of non-stationary belief nodes and thus is able to explore the velocity space and find much more efficient trajectories than previous BRM methods. The performance of CS-BRM is evaluated and compared to previous belief space planning approaches using several numerical examples and experimental demonstrations, illustrating the benefits of the proposed approach.**

***Index Terms*—Motion planning, uncertainty, covariance steering, belief space roadmap, probabilistic roadmap**

I. Introduction

Motion uncertainty and measurement noise arise in all real-world robotic applications. When evaluating the safety of a robot under motion and estimation uncertainties, it is no longer sufficient to rely only on deterministic indicators of performance, such as whether the robot is in collision-free or in-collision status. Instead, the state of the robot is best characterized by a probability distribution over all possible states, which is commonly referred to as the *belief* [1]–[3]. Explicitly taking into account the motion and observation uncertainties thus requires planning in the belief space, which allows one to compute the collision probability and thus make more informed decisions.

Planning under motion and observation uncertainties is referred to as belief space planning, which can be formulated as a partially observable Markov decision process (POMDP) problem [4]. Solving POMDPs for continuous state, control, and observation spaces is, however, intractable, especially for long-horizon, global planning problems. Existing methods based on discretization are resolution-limited. Optimization over the entire discretized belief space to find a path is computationally expensive and does not scale well to large-scale problems [5]–[7].



(a) (b)

Fig. 1: Illustration of PRM and BRM algorithms. (a) PRM and the planned path (red line). PRM does not consider motion and observation uncertainties. The planned path has a high probability of collision. (b) BRM and the planned path. BRM nodes are distributions of the state sampled in the belief space. BRM makes informed plans by considering both safety and path efficiency. One challenge of the BRM algorithm is to derive the edge controller that achieves node reachability, i.e., steers the system from one distribution to another. Here only Gaussian distributions are assumed, characterized by their mean and 3σ covariance ellipses.

Sampling-based motion planning algorithms such as probabilistic roadmaps (PRM) [8] and rapidly exploring random trees (RRTs) [9] can be used to solve planning problems in high-dimensional continuous state spaces by building a roadmap or a tree incrementally through sampling. However, traditional PRM-based methods only address deterministic systems. PRM methods can be extended to belief space planning using belief space roadmaps (BRMs) [10]–[12]. An illustration of the difference between PRM and BRM algorithms is shown in Figure 1. One of the main challenges of BRM methods is the reachability of belief nodes. Even if the robot has full control of its mean, it is difficult to meet a specified tolerance (e.g., covariance). Since the nodes in the BRM are sampled in the belief space, the edges in the BRM graph should ideally steer the robot from one distribution to another. If reachability of the BRM nodes is not achieved, an edge in the BRM depends on all preceding edges along the path. This dependence among edges is referred to as the "curse of history" problem for POMDPs [11], [13]. It breaks

the optimal substructure property, which is required for search algorithms such as Dijkstra's algorithm or A*.

Recent developments in explicitly controlling the covariance of a linear system [14], [15] provide an appealing approach to construct a BRM with guarantees of node reachability. In particular, for a discrete-time linear stochastic system, covariance steering theory designs a controller that steers the system from an initial Gaussian distribution to a terminal Gaussian distribution in finite-time [16]–[18]. Applications of this idea have been reported for the stochastic control of a Mars lander during powered descent and during aerocapture in [19] and [20]. In [16], the covariance steering problem is formulated as a convex program. Additional state chance constraints are considered in [18] while nonlinear systems are considered in [21] using iterative linearization. The covariance steering problem with output feedback has also been studied in [22]–[24].

In this paper, we propose the CS-BRM algorithm, which uses covariance steering as the edge controller of a BRM to ensure a *priori* node reachability. Since the goal of covariance steering is to reach a given distribution of the state, it is well-suited for reaching a belief node and thus provides a way of addressing the "curse of history" problem. In addition, covariance steering avoids the limitation of sampling equilibrium states, and thus the proposed CS-BRM method allows sampling of non-stationary belief nodes. While current state-of-the-art methods such as SLQG-FIRM [11] require a full stop (zero velocity) at every node, our method allows searching in the velocity space and thus finds paths with lower cost. In addition, no converging phase is required at each node compared to SLQG-FIRM, thus resulting in more efficient motion.

A. Related Work

Planning in belief space has long been a topic of great interest in the robotics community as a means to handle control and decision-making problems under motion and sensory measurement uncertainties. The problem is typically formulated as a POMDP, whose solution in general domains is, however, very challenging [4], [25], despite some recent progress in terms of more efficient POMDP solvers [26], [27].

In planning problems with partial or incomplete information of the state, one needs to keep track of the whole distribution of all possible system states, referred to as the belief or information state. Planning in such infinite-dimensional spaces can become tractable by the use of roadmaps, that is, discrete graph representations of the environment constructed by sampling. Since their introduction in [10], [28] such belief roadmaps (BRMs) have increased in popularity owing to their simplicity and their ability to avoid local minima.

In general, in the literature, the *belief* may be classified into *(i)* the estimation belief (e-belief), describing the output of the estimator, i.e., the pdf over the error between the estimated value and true value of the state; *(ii)* control belief (c-belief), which refers to the pdf over "separated control" error, i.e., the error between the estimated state and the desired state; and *(iii)* the full belief (f-belief), that is, the belief over the true state. The *e-belief* planning tries to obtain better state estimates and finds a path with minimum estimation uncertainty. The *f-belief* planning aims at minimizing the full uncertainty and takes into account the impact of the controller as well. In this paper, we consider the *f-belief* planning problem and will refer to it as *belief* planning for simplicity.

Planning in *e-belief* space was studied in [10], [29], where the goal was to find the minimum estimation uncertainty path for a robot from a starting position to a goal position. Reference [10] constructed the roadmap using only the mean poses of the robot. Given the initial belief (e.g., mean and covariance), this initial belief is propagated through the roadmap by simulating the control inputs and the received measurements. Then, graph searches are performed to find paths from the initial belief node to other belief nodes in the roadmap. The BRM in [10] depends on the initial belief of the robot and must be re-computed if one wants to start from a different mean or covariance. This means that the BRM cannot be built beforehand as in the case of a PRM. The state estimate error covariance, which is computed using a Kalman filter, gives a measure of the confidence of the state estimate. However, the probability distribution of the true state of the robot is required for motion planning, instead of the estimation error covariance. By taking into account the controller and the sensors used, [12] computes the true *a priori* probability distribution of the state of the robot and studies the *f-belief* planning problem using the RRT algorithm. Reference [30] studies the *f-belief* planning problem by searching over a graph to project a belief tree. However, the independence between edges is not satisfied in [12], [30] and these tree-based methods can not be extended to PRM.

An important advancement in the state-of-the-art of BRM methods was [11], which tackled the "curse of history" problem. The proposed SLQG-FIRM method achieves node reachability using a stationary LQG controller. The edge controller in [11] is a concatenation of a time-varying LQG controller and a stationary linear quadratic Gaussian (SLQG) controller. First, the time-varying LQG controller is used to steer the robot to a neighborhood of the goal node. Then, the controller is switched to the SLQG controller to reach the node. The belief converges to a stationary belief *in distribution*, and the estimation error covariance converges to a stationary covariance. Since the SLQG controller requires an infinite time to converge to a belief node, in practice, the belief will converge to a small neighborhood of the stationary belief.

One limitation of this method is that the nodes have to be stationary. That is, the nodes in the BRM graph need to be sampled in the equilibrium space of the robot, which usually means zero velocity. The robot has to stop at every node, which results in inefficient motion plans. This method, used in [11], cannot explore the velocity space, and hence the resulting paths are suboptimal. Secondly, a converging process is required at every node. The robot will have to "wait" at each node, which will increase the time required for the robot to reach the goal. Some remedies are introduced in [31], [32] for systems that cannot reach zero velocity, such as fixed-wing aircraft, by using periodic trajectories and periodic controllers. The periodic controller is applied

repeatedly until the trajectory of the vehicle converges to the periodic trajectory. Thus, a "waiting" procedure is still required in these approaches. Online replanning in belief space is studied in [33]. The method in [33] improves the online phase by recomputing the local plans, which includes adding a virtual belief node and local belief edges to the FIRM graph, and solving a dynamic program at every time-step. The offline roadmap construction phase is the same as FIRM [11].

Extending the SLQG-FIRM to nonlinear systems requires a nominal trajectory for each edge [11]. These nominal trajectories are either assumed to be given or being approximated by simple straight lines [11], [12]. However, the nominal trajectory has to be dynamically feasible in order to apply a time-varying LQG controller. Also, the nominal trajectory must be optimal if one wishes to generate optimal motion plans. Finding the optimal nominal trajectory requires solving a two-point boundary value problem (TPBVP) [34]. An efficient solution for the TPBVP is only available for some simple systems [31], [35], [36], while solving the TPBVP for general nonlinear systems requires iterative methods [37] and can be computationally expensive. In this paper, in addition to presenting a new edge controller for general classes of BRMs that solves the node reachability problem of current BRMs, we also develop a simple and efficient algorithm to find suitable nominal trajectories between the nodes of the BRM graph to steer the mean states. The algorithm is integrated with our previous method [22] resulting in an output-feedback covariance steering method for planning for general classes of uncertain nonlinear systems.

In this paper, we propose the Covariance Steering Belief RoadMap (CS-BRM) algorithm for multi-query belief space planning. The belief RoadMap can be pre-constructed offline. Online belief space planning can be done more efficiently than solving POMDPs [7], [27] and tree-based algorithms [12], [30] by performing a graph search. Compared with [27] and [7], CS-BRM can deal with long planning horizons, and solve global planning problems in large environments. Compared with [11], CS-BRM avoids sampling stationary beliefs and finds better motion plans.

B. Contributions

The contributions of the paper are summarized as follows.

- A new belief space roadmap method, CS-BRM, is developed, which adopts the covariance steering theory to achieve node reachability in BRM and overcomes the limitation of sampling stationary nodes. Compared to previous BRM methods, CS-BRM allows searching the velocity space of the belief nodes and it does not require a convergence phase at every node.
- An efficient algorithm, called the Compatible Nominal Trajectory (CNT) algorithm, is proposed to compute nominal trajectories for a nonlinear system. CNT utilizes the analytical solution of the mean control of a linear time-varying system along with iterative linearizations.
- An output-feedback covariance steering method for nonlinear systems is developed, which is based on the CNT algorithm and the separation of the mean and covariance controllers.
- A heuristic method for sampling the velocities of BRM nodes is introduced, which provides a strategy to explore the velocity space while, at the same time, tries to avoid adding unnecessary belief nodes and edges.

We compare CS-BRM with the current state-of-the-art SLQG-FIRM method and show that by sampling non-stationary belief nodes and searching the velocity space, our method is able to find more efficient and lower-cost plans.

This paper builds on our previous papers [38] and [22] and extends these works in several directions. First, compared to [38], we provide a more detailed description and analysis of the CS-BRM algorithm, including the proofs of the main results that are missing in [38]. Second, a new heuristic for sampling the velocity space is proposed, and more simulation studies are added, including the experimental validation of the proposed CS-BRM algorithm on a small quadrotor. Third, the output-feedback covariance steering controller developed in [22] is adopted as an edge controller to construct a roadmap in belief space to achieve multi-query motion planning for stochastic systems in cluttered environments.

The paper is organized as follows. The statement of the problem is given in Section II. In Section III, the output-feedback covariance steering theory for linear systems is outlined. The covariance steering controller is used as the edge controller of the BRM edges. In Section IV, the covariance steering theory is extended to deal with nonlinear systems. The main algorithm, CS-BRM, is given in Section V. The numerical implementation of the proposed algorithm in different environments is presented in Section VI. Finally, Section VII concludes the paper.

II. Problem Statement

We consider the problem of planning for a nonholonomic robot in an uncertain environment that contains obstacles. The uncertainty in the problem stems from model uncertainty, as well as from sensor noise that corrupts the measurements. We model such a system by a stochastic difference equation of the form

$$x_{k+1} = f(x_k, u_k, w_k), \tag{1}$$

where $k = 0, 1, \ldots, N-1$ are the discrete time-steps, $x_k \in \mathbb{R}^{n_x}$ is the state, and $u_k \in \mathbb{R}^{n_u}$ is the control input. The steps of the noise process $w_k \in \mathbb{R}^{n_w}$ are i.i.d standard Gaussian random vectors. The measurements are given by the noisy and partial sensing model

$$y_k = h(x_k, v_k), \tag{2}$$

where $y_k \in \mathbb{R}^{n_y}$ is the measurement at time step k, and the steps of the process $v_k \in \mathbb{R}^{n_y}$ are i.i.d standard Gaussian random vectors. We assume that $(w_k)_{k=0}^{N-1}$ and $(v_k)_{k=0}^{N-1}$ are independent.

The objective is to steer the system (1) from some initial state x_0 to some final state x_N within N time steps while avoiding obstacles and, at the same time, minimize a given performance index. The controller u_k at time step k is allowed to depend on the whole history of measurements y_ℓ, $\ell = 0, \ldots, k$ up to time k but not on any future measurements.

This is a difficult problem to solve in its full generality and for high-dimensional spaces. For the case of a fully actuated, holonomic system ($n_x = n_u$) without noise ($w_k = v_k = 0$ for all $k = 0, 1, \ldots, N-1$), the probabilistic roadmap (PRM) algorithm has been initially developed to solve such planning problems in high-dimensional configuration spaces [8]. The PRM algorithm has two phases: a roadmap construction phase and a query phase. The roadmap is constructed by sampling the collision-free configuration space and by making connections between those sampled configurations using feasible paths. The PRM results in an undirected graph where the sampled configurations are the nodes of the graph and the feasible paths between nodes are the edges of the graph. For holonomic or fully actuated systems, the feasible path between two nodes is usually a straight line connecting the two configurations.

The PRM algorithm has been extended to dynamical systems subject to differential constraints (DPRM) in [9], [36], [39]. Compared to the original PRM algorithm, the nodes in DPRM are sampled in the state space of the robot instead of the configuration space. The trajectories are generated by connecting two nodes by directed edges using an edge controller. The edge controller can either be an optimal controller from the solution of the TPBVP or a steering function using, for instance, a trajectory optimizer or reinforcement learning techniques [40]. After this directed graph is constructed, the query phase in DPRM is the same as in the original PRM method.

In the presence of system uncertainty and sensor measurement uncertainty, such as in (1) and (2), the robot cannot be controlled to a state with certainty, and the true state of the robot is also not available. In this case, only a probability distribution over all possible states, referred to as the *belief*, is available for motion planning. Therefore, the PRM needs to be constructed in the belief space. Such belief roadmaps (BRM) address uncertainties in the dynamics along with noisy sensor measurements. Each node sampled in the belief space is a distribution over the state. The edges between nodes in a BRM indicate the ability to steer the state from one distribution to another. Therefore, each edge in the BRM is constructed by applying a controller to the stochastic system (1). In the next section, we describe a methodology to design BRM edge controllers that allow steering from one BRM belief node to another. Similar to previous works [11], [12], we consider Gaussian distributions where the belief is given by the state mean and the state covariance.

III. Edge Controller Design

A. Covariance Steering

It is assumed that the nonlinear system (1)-(2) can be well approximated locally by its linearization about a given, nominal trajectory. Specifically, given a nominal trajectory $(n_k)_{k=0}^{N-1}$, where $n_k = (x_k^r, u_k^r)$, we can construct via linearization the discrete, linear time-varying model

$$x_{k+1} = A_k x_k + B_k u_k + h_k + G_k w_k, \tag{3}$$

$$y_k = C_k x_k + D_k v_k, \tag{4}$$

where $h_k \in \mathbb{R}^{n_x}$ is the drift term, $A_k \in \mathbb{R}^{n_x \times n_x}$, $B_k \in \mathbb{R}^{n_x \times n_u}$, and $G_k \in \mathbb{R}^{n_x \times n_x}$ are system matrices, and $C_k \in \mathbb{R}^{n_y \times n_x}$ and $D_k \in \mathbb{R}^{n_y \times n_y}$ are observation model matrices computed from

$$\begin{aligned} A_k &= \frac{\partial f}{\partial x}|_{(x_k^r, u_k^r, 0)}, \quad B_k = \frac{\partial f}{\partial u}|_{(x_k^r, u_k^r, 0)}, \\ G_k &= \frac{\partial f}{\partial w}|_{(x_k^r, u_k^r, 0)}, \quad h_k = f(x_k^r, u_k^r, 0) - A_k x_k^r - B_k u_k^r, \\ C_k &= \frac{\partial h}{\partial x}|_{(x_k^r, 0)}, \quad D_k = \frac{\partial h}{\partial v}|_{(x_k^r, 0)}, \end{aligned} \tag{5}$$

Note that the linearization of the sensing model may also include a drift term g_k, in which case (4) is replaced by $y_k = C_k x_k + g_k + D_k v_k$, where $g_k = h(x_k^r, 0) - C_k x_k^r$. Since g_k is known and deterministic, henceforth it suffices to consider the sensing model (4) without loss of generality.

Recent developments in steering the covariance of linear systems of the form (3)–(4) provide an appealing approach to construct the BRM with guarantees of node reachability. In particular, for a discrete-time linear stochastic system of the form (3)-(4), covariance steering theory designs a controller that steers the system from an initial Gaussian distribution to a terminal Gaussian distribution in finite time.

We define the covariance steering problem as follows.

Problem 1: Find the control sequence $u = (u_k)_{k=0}^{N-1}$ such that the system given by (3) and (4), starting from the initial state distribution $x_0 \sim \mathcal{N}(\bar{x}_0, P_0)$, reaches the final distribution $x_N \sim \mathcal{N}(\bar{x}_N, P_f)$, while minimizing the cost functional

$$J(u) = \mathbb{E}\left[\sum_{k=0}^{N-1} (x_k - m_k)^\top Q_k (x_k - m_k) + u_k^\top R_k u_k\right], \tag{6}$$

where $(m_k)_{k=0}^{N-1}$ is a given reference trajectory of the states, $\bar{x}_k = \mathbb{E}(x_k)$ is the mean of the state x_k, P_0 and P_f are the covariance matrices of x_0 and x_N, respectively, and the matrices $(Q_k \succeq 0)$ and $(R_k \succ 0)$ are given problem parameters.

Problem 1 is a two-point boundary value problem (TPBVP) with boundary conditions on both the mean and the covariance. In the next section, we construct the BRM edge controller based on the covariance steering approach [18], [22] for a linear system of the form (3)-(4), which is the key ingredient of the CS-BRM algorithm. Later, in Section IV, we extend the covariance steering results to consider nonlinear systems.

B. Separation of Observation and Control

We assume that the control input u_k at time-step k is an affine function of the measurement data. It follows that the state will be Gaussian distributed over the entire horizon of the problem. To solve Problem 1, we use a Kalman filter to estimate the state.

Specifically, let the prior initial state estimate be $\hat{x}_{0^-}$. We assume that the distribution of $\hat{x}_{0^-}$ is known and is given by $\hat{x}_{0^-} \sim \mathcal{N}(\bar{x}_0, \hat{P}_{0^-})$. Let the prior initial estimation error be $\tilde{x}_{0^-} = x_0 - \hat{x}_{0^-}$ and let its distribution be given by $\tilde{x}_{0^-} \sim \mathcal{N}(0, \tilde{P}_{0^-})$. The estimate $\hat{x}_0$ is updated from $\hat{x}_{0^-}$ using

observation y_0 at $k=0$. Define the filtration $(Y_k)_{k=-1}^N$, where $Y_k = \sigma(\hat{x}_{0^-}, y_i : 0 \le i \le k)$, $Y_{-1} = \sigma(\hat{x}_{0^-})$ and $\sigma(\cdot)$ denotes the σ-algebra, and define the estimated state at time k as the conditional expectation $\hat{x}_k = \mathbb{E}[x_k|Y_k]$ and the estimation error as $\tilde{x}_k = x_k - \hat{x}_k$. We then have

$$\mathbb{E}[\hat{x}_k] = \mathbb{E}[\mathbb{E}[x_k|Y_k]] = \mathbb{E}[x_k] \triangleq \bar{x}_k. \tag{7}$$

Notice that $\hat{x}_k$ and $\tilde{x}_k$ are uncorrelated, since

$$\begin{aligned}\mathbb{E}[\hat{x}_k\tilde{x}_k] &= \mathbb{E}[\hat{x}_k(x_k - \hat{x}_k)] \\ &= \mathbb{E}[\mathbb{E}[\hat{x}_k(x_k - \hat{x}_k)|Y_k]] \\ &= \mathbb{E}[\hat{x}_k(\mathbb{E}[x_k|Y_k] - \hat{x}_k)] = 0.\end{aligned} \tag{8}$$

Define the covariance of x_k, $\hat{x}_k$ and $\tilde{x}_k$ as $P_k = \mathbb{E}[(x_k - \bar{x}_k)(x_k - \bar{x}_k)^\top]$, $\hat{P}_k = \mathbb{E}[(\hat{x}_k - \bar{x}_k)(\hat{x}_k - \bar{x}_k)^\top]$ and $\tilde{P}_k = \mathbb{E}[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^\top]$, respectively. Using (8), it can be readily shown that $P_k = \hat{P}_k + \tilde{P}_k$. Define the prior estimated state and prior estimation error as $\hat{x}_{k^-} = \mathbb{E}[x_k|Y_{k-1}]$ and $\tilde{x}_{k^-} = x_k - \hat{x}_{k^-}$. The corresponding covariances are denoted as $\hat{P}_{k^-}$ and $\tilde{P}_{k^-}$. We assume that $\hat{x}_{0^-}$ and $\tilde{x}_{0^-}$ are uncorrelated. Under this assumption, it can be verified that $P_0 = \hat{P}_{0^-} + \tilde{P}_{0^-}$. Similarly to (8), $\hat{x}_{k^-}$ and $\tilde{x}_{k^-}$ are uncorrelated and $P_k = \hat{P}_{k^-} + \tilde{P}_{k^-}$.

The Kalman filter updates are given by

$$\begin{aligned}\hat{x}_k &= \hat{x}_{k^-} + L_k(y_k - C_k\hat{x}_{k^-}), \\ \hat{x}_{k^-} &= A_{k-1}\hat{x}_{k-1} + B_{k-1}u_{k-1} + h_{k-1},\end{aligned} \tag{9}$$

where

$$\begin{aligned}L_k &= \tilde{P}_{k^-}C_k^\top(C_k\tilde{P}_{k^-}C_k^\top + D_kD_k^\top)^{-1}, \\ \tilde{P}_k &= (I - L_kC_k)\tilde{P}_{k^-}, \\ \tilde{P}_{k^-} &= A_{k-1}\tilde{P}_{k-1}A_{k-1}^\top + G_{k-1}G_{k-1}^\top,\end{aligned} \tag{10}$$

where L_k is the Kalman gain. From (10), the evolution of $\tilde{P}_{k^-}$ and $\tilde{P}_k$ do not depend on the control $(u_k)_{k=0}^{N-1}$.

Furthermore, note that,

$$\begin{aligned}&\mathbb{E}[(x_k - m_k)^\top Q_k(x_k - m_k)] \\ &= \mathbb{E}[x_k^\top Q_k x_k] - 2\bar{x}_k^\top Q_k m_k + m_k^\top Q_k m_k \\ &= \mathbb{E}[(\hat{x}_k + \tilde{x}_k)^\top Q_k(\hat{x}_k + \tilde{x}_k)] - 2\bar{x}_k^\top Q_k m_k + m_k^\top Q_k m_k \\ &= \mathbb{E}[\tilde{x}_k^\top Q_k\tilde{x}_k] + \mathbb{E}[\hat{x}_k^\top Q_k\hat{x}_k] - 2\bar{x}_k^\top Q_k m_k + m_k^\top Q_k m_k \\ &= \mathrm{trace}(\tilde{P}_kQ_k) + \mathbb{E}[\hat{x}_k^\top Q_k\hat{x}_k] - 2\bar{x}_k^\top Q_k m_k + m_k^\top Q_k m_k.\end{aligned} \tag{11}$$

Thus, the cost functional (6) can be written as

$$\begin{aligned}J(u) =& \mathbb{E}\left[\sum_{k=0}^{N-1}\hat{x}_k^\top Q_k\hat{x}_k + u_k^\top R_ku_k\right] - 2\sum_{k=0}^{N-1}\bar{x}_k^\top Q_k m_k \\ &+ \sum_{k=0}^{N-1}(\mathrm{trace}(\tilde{P}_kQ_k) + m_k^\top Q_k m_k),\end{aligned} \tag{12}$$

where the last summation is deterministic and does not depend on the control. Thus, we can discard this term without changing the problem.

Next, define the *innovation process* $(\xi_k)_{k=0}^N$ by

$$\xi_k = y_k - \mathbb{E}[y_k|Y_{k-1}], \tag{13}$$

and note that

$$\mathbb{E}[y_k|Y_{k-1}] = \mathbb{E}[C_kx_k + D_kv_k|Y_{k-1}] = C_k\hat{x}_{k^-}. \tag{14}$$

It follows that

$$\xi_k = y_k - C_k\hat{x}_{k^-} = C_k\tilde{x}_{k^-} + D_kv_k, \tag{15}$$

and ξ_k has zero mean, $\mathbb{E}[\xi_k] = 0$. It can be shown that the innovation process at different time steps is uncorrelated [41]. Thus, the covariance of ξ_k at time k is

$$P_{\xi_k} = \mathbb{E}[\xi_k\xi_k^\top] = C_k\tilde{P}_{k^-}C_k^\top + D_kD_k^\top. \tag{16}$$

Substituting the innovation process equations (13) and (14) into the the Kalman filter updates (9), we obtain the estimated state process

$$\hat{x}_{k+1} = A_k\hat{x}_k + B_ku_k + h_k + L_{k+1}\xi_{k+1}, \tag{17}$$

where $\hat{x}_0 = \hat{x}_{0^-} + L_0\xi_0$.

We can then restate Problem 1 as follows.

Problem 2: Find the control sequence $(u_k)_{k=0}^{N-1}$, such that the system (17) starting from the initial distribution $\hat{x}_{0^-} \sim \mathcal{N}(\bar{x}_0, P_0 - \tilde{P}_{0^-})$ reaches the final distribution $\hat{x}_N \sim \mathcal{N}(\bar{x}_N, P_f - \tilde{P}_N)$, while minimizing the cost functional

$$\hat{J}(u) = \mathbb{E}\left[\sum_{k=0}^{N-1}\hat{x}_k^\top Q_k\hat{x}_k + u_k^\top R_ku_k\right] - 2\sum_{k=0}^{N-1}\bar{x}_k^\top Q_k m_k. \tag{18}$$

To summarize, the covariance steering problem of the state x_k with output feedback has been transformed to a covariance steering problem of the estimated state $\hat{x}_k$, where $\hat{x}_k$ is computed using the Kalman filter. While Problem 1 is defined with respect to the unknown state x_k, Problem 2 is defined with respect to the known state estimate $\hat{x}_k$. Also, the noise term G_kw_k in (3) is replaced by the noise term $L_{k+1}\xi_{k+1}$ in (17).

C. Separation of Mean Control and Covariance Control

Problem 2 defines a covariance steering problem for the estimated state $\hat{x}_k$, which is Gaussian distributed. In this section, we will separate Problem 2 into a mean control problem and a covariance control problem.

At each time step k, we can obtain the expression of $\hat{x}_k$ by forward propagating equation (17). Define the augmented vectors

$$U_k = [u_0^\top\ u_1^\top\ \cdots\ u_k^\top]^\top, \quad \Xi_k = [\xi_0^\top\ \xi_1^\top\ \cdots\ \xi_k^\top]^\top. \tag{19}$$

By combining the steps of the iterative equation (17), $\hat{x}_k$ can be written as

$$\hat{x}_k = \bar{A}_k\hat{x}_{0^-} + \bar{B}_kU_{k-1} + H_k + \bar{L}_k\Xi_k, \tag{20}$$

for $k = 1, \ldots, N$, where the system matrices $\bar{A}_k$, $\bar{B}_k$, H_k, and $\bar{L}_k$ are given in Appendix A.

The mean of the estimated state $\hat{x}_k$ is then given by

$$\bar{x}_k = \mathbb{E}[\hat{x}_k] = \bar{A}_k\bar{x}_0 + \bar{B}_k\bar{U}_{k-1} + H_k, \tag{21}$$

where $\bar{U}_k \triangleq \mathbb{E}[U_k]$ is the mean control sequence. Define $\tilde{U}_k \triangleq U_k - \bar{U}_k$, $\check{x}_k \triangleq \hat{x}_k - \bar{x}_k$, and $\check{x}_{0^-} \triangleq \hat{x}_{0^-} - \bar{x}_0$. Using (20) and

(21) we have

$$\check{x}_k = \bar{A}_k \check{x}_{0\text{-}} + \bar{B}_k \tilde{U}_{k-1} + \bar{L}_k \Xi_k. \tag{22}$$

Define $\hat{X}_k = [\hat{x}_0^\top \ \hat{x}_1^\top \ \cdots \ \hat{x}_k^\top]^\top$, $\hat{X} = \hat{X}_N$, $U = U_{N-1}$, and $\Xi = \Xi_N$. Using (20) and stacking the equations of $\hat{x}_k$ from different time-steps, we obtain

$$\hat{X} = A\hat{x}_{0\text{-}} + BU + H + L\Xi, \tag{23}$$

where the system matrices A, B, H, and L are given in Appendix A.

Let $\bar{X} \triangleq \mathbb{E}[\hat{X}]$, $\bar{U} \triangleq \mathbb{E}[U]$, $\check{X} \triangleq \hat{X} - \bar{X}$, and $\tilde{U} \triangleq U - \bar{U}$. It follows that

$$\bar{X} = A\bar{x}_0 + B\bar{U} + H, \tag{24}$$

and

$$\check{X} = A\check{x}_{0\text{-}} + B\tilde{U} + L\Xi. \tag{25}$$

The cost functional in (18) can be rewritten as

$$\begin{aligned}\hat{J}(u) =& \mathbb{E}\left[\hat{X}^\top Q \hat{X} + U^\top R U\right] - 2\bar{X}^\top Q M_r \\ =& \mathbb{E}\left[(\check{X} + \bar{X})^\top Q (\check{X} + \bar{X})\right] \\ &+ \mathbb{E}\left[(\tilde{U} + \bar{U})^\top R (\tilde{U} + \bar{U})\right] - 2\bar{X}^\top Q M_r \\ =& \mathbb{E}[\check{X}^\top Q \check{X} + \tilde{U}^\top R \tilde{U}] + \bar{X}^\top Q \bar{X} + \bar{U}^\top R \bar{U} \\ &- 2\bar{X}^\top Q M_r,\end{aligned} \tag{26}$$

where $Q = \text{blkdiag}(Q_0, Q_1, \ldots, Q_{N-1}, 0)$, $R = \text{blkdiag}(R_0, R_1, \ldots, R_{N-1})$, $M_r = [m_0^\top \ m_1^\top \ \cdots \ m_N^\top]^\top$.

From (24), (25), and (26), the covariance steering problem of the estimated state $\hat{x}$ can be divided into mean control and covariance control problems. The mean control problem is given by

$$\begin{aligned}\min_{\bar{U}} \ & \bar{X}^\top Q \bar{X} + \bar{U}^\top R \bar{U} - 2\bar{X}^\top Q M_r \\ \text{s.t.} \ & \bar{X} = A\bar{x}_0 + B\bar{U} + H, \\ & E_0 \bar{X} = \bar{x}_0, \quad E_N \bar{X} = \bar{x}_N,\end{aligned} \tag{27}$$

where E_k is a matrix defined such that $E_k \bar{X} = \bar{x}_k$. The covariance control problem is given by

$$\begin{aligned}\min_{\tilde{U}} \ & \mathbb{E}[\check{X}^\top Q \check{X} + \tilde{U}^\top R \tilde{U}] \\ \text{s.t.} \ & \check{X} = A\check{x}_{0\text{-}} + B\tilde{U} + L\Xi, \\ & \mathbb{E}[\check{x}_{0\text{-}} \check{x}_{0\text{-}}^\top] = \check{P}_{0\text{-}}, \\ & E_N \mathbb{E}[\check{X}\check{X}^\top] E_N^\top = \check{P}_N\end{aligned} \tag{28}$$

where $\check{P}_{0\text{-}} = P_0 - \tilde{P}_{0\text{-}}$ and $\check{P}_N = P_f - \tilde{P}_N$.

D. Solutions of the Mean Control and Covariance Control Problems

We assume that the discrete-time linear time-varying system (3) is controllable. In this case, an analytic solution for the mean control problem (27) is readily available. We summarize the result as follows.

Proposition 1: The solution to the mean control problem (27) is

$$\begin{aligned}\bar{U}^* =& W(-V + \bar{B}_N^\top (\bar{B}_N M \bar{B}_N^\top)^{-1} \\ & (\bar{x}_N - \bar{A}_N \bar{x}_0 - H_N + \bar{B}_N W V)),\end{aligned} \tag{29}$$

where $W = (B^\top Q B + R)^{-1}$, and $V = B^\top Q (A\bar{x}_0 + H - M_r)$.

Proof. See Appendix B. □

After solving the mean control problem, we obtain the mean trajectory given by $(\bar{x}_k)_{k=0}^N$ and $(\bar{u}_k)_{k=0}^{N-1}$. To solve the covariance control problem, we consider a feedback controller of the form

$$\tilde{u}_k = \sum_{i=0}^{k} K_{k,i} \check{x}_i, \tag{30}$$

for $k = 0, 1, \cdots, N-1$. The problem remains to find the feedback gains $K_{k,i}$. Using (30), we can write $\tilde{U}$ as

$$\tilde{U} = K\check{X}, \tag{31}$$

where

$$K = \begin{bmatrix} K_{0,0} & 0 & \cdots & 0 & 0 \\ K_{1,0} & K_{1,1} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & 0 \\ K_{N-1,0} & K_{N-1,1} & \cdots & K_{N-1,N-1} & 0 \end{bmatrix}. \tag{32}$$

Using (31), (25) becomes

$$\check{X} = A\check{x}_{0\text{-}} + BK\check{X} + L\Xi. \tag{33}$$

Solving for $\check{X}$ using (33), we have

$$\check{X} = (I - BK)^{-1}(A\check{x}_{0\text{-}} + L\Xi). \tag{34}$$

Since K is block lower-triangular and B is strictly block lower-triangular, the matrix $I - BK$ is invertible. Following [42], define the new decision variable F as

$$F = K(I - BK)^{-1}. \tag{35}$$

It can be verified that F satisfies

$$I + BF = (I - BK)^{-1}, \tag{36}$$

and

$$K = F(I + BF)^{-1}. \tag{37}$$

Therefore, we may optimize over F in place of K. Substituting (36) into (34), we get

$$\check{X} = (I + BF)(A\check{x}_{0\text{-}} + L\Xi). \tag{38}$$

By assumption, $\hat{x}_{0\text{-}}$ is uncorrelated with $\tilde{x}_{0\text{-}}$ and v_0. By (15), $\hat{x}_{0\text{-}}$ is uncorrelated with Ξ. Thus, $\check{x}_{0\text{-}}$ is uncorrelated with Ξ. Let $Z \triangleq A\check{x}_{0\text{-}} + L\Xi$. Then Z has covariance

$$\begin{aligned}P_Z &= \mathbb{E}[ZZ^\top] = A\mathbb{E}[\check{x}_{0\text{-}}\check{x}_{0\text{-}}^\top]A^\top + L\mathbb{E}[\Xi\Xi^\top]L^\top \\ &= A\hat{P}_{0\text{-}}A^\top + LP_\Xi L^\top,\end{aligned} \tag{39}$$

where P_Ξ is given by

$$P_\Xi = \mathbb{E}[\Xi\,\Xi^\top] = \text{blkdiag}(P_{\xi_0}, \cdots, P_{\xi_N}). \tag{40}$$

From (38) and (39), the covariance of $\check{X}$ is

$$P_{\check{X}} = \mathbb{E}[\check{X}\check{X}^\top] = (I + BF)P_Z(I + BF)^\top. \tag{41}$$

Using (37), the covariance of the control is given by

$$\begin{aligned} P_U &= \mathbb{E}[\tilde{U}\tilde{U}^\top] = \mathbb{E}[K\check{X}\check{X}^\top K^\top] \\ &= K(I+BF)P_Z(I+BF)^\top K^\top \\ &= FP_ZF^\top. \end{aligned} \tag{42}$$

Using (38) and (42), we can rewrite the cost functional in (28) as

$$\begin{aligned} &\mathbb{E}[\check{X}^\top Q\check{X} + \tilde{U}^\top R\tilde{U}] \\ &= \mathbb{E}[((I+BF)Z)^\top Q(I+BF)Z + \tilde{U}^\top R\tilde{U}] \\ &= \mathbb{E}[\mathrm{trace}(Q(I+BF)ZZ^\top(I+BF)^\top) + \mathrm{trace}(R\tilde{U}\tilde{U}^\top)] \\ &= \mathrm{trace}(Q(I+BF)P_Z(I+BF)^\top) + \mathrm{trace}(RFP_ZF^\top) \\ &= \mathrm{trace}[((I+BF)^\top Q(I+BF) + F^\top RF)P_Z], \end{aligned} \tag{43}$$

which is convex in F. The terminal covariance constraint in (28) may be rewritten as

$$E_N(I+BF)P_Z(I+BF)^\top E_N^\top = P_f - \tilde{P}_N. \tag{44}$$

The above quadratic equality constraint is not necessarily convex. Thus, solving the covariance control problem with equality constraint requires the solution of a nonlinear program. We can relax the equality constraint (44) to an inequality constraint to reduce complexity. The corresponding inequality constraint is

$$E_N(I+BF)P_Z(I+BF)^\top E_N^\top \preceq P_f - \tilde{P}_N, \tag{45}$$

or, equivalently [43],

$$\|P_Z^{1/2}(I+BF)^\top E_N^\top (P_f - \tilde{P}_N)^{-1/2}\| - 1 \leq 0. \tag{46}$$

To summarize, the solution of the covariance control problem is obtained by solving the optimization problem given by (43) and (46). The optimization problem with cost function (43) and constraint (46) is a convex optimization problem and can be solved efficiently with existing solvers [44], [45].

IV. Nonlinear Covariance Steering

The covariance steering problem for stochastic, discrete, linear time-varying systems was studied in Section III. In this section, we extend the results of Section III to deal with nonlinear systems that are locally well-approximated by their linearization. Based on the separation between mean control and covariance control, we develop an efficient algorithm to find nominal trajectories for nonlinear systems using the mean controller developed in Section III-D.

Consider again the nonlinear stochastic system with dynamics (1). The system is linearized along a nominal trajectory $(n_k)_{k=0}^{N-1}$, where $n_k = (x_k^r, u_k^r)$, to obtain the system shown in (3)-(4). Finding the nominal trajectory $(n_k)_{k=0}^{N-1}$ itself is an open-loop controller design problem. A deterministic nonlinear dynamic model, i.e., with the noise w_k set to zero, is considered when designing the nominal trajectory. Next, we define a class of nominal trajectories, referred to as *compatible nominal trajectories*, and we develop an algorithm to find them.

A. Compatible Nominal Trajectory Generation

Consider the deterministic nonlinear system

$$x_{k+1} = f(x_k, u_k, 0). \tag{47}$$

Let $(x_k^r)_{k=0}^{N-1}$ and $(u_k^r)_{k=0}^{N-1}$ be a nominal trajectory for this system. The linearized model (3) along this nominal trajectory will be used by the mean controller (29) to compute a control sequence $(u_k^c)_{k=0}^{N-1}$ and the corresponding state sequence $(x_k^c)_{k=0}^{N-1}$ using (24). Next, we give the precise definition of a compatible nominal trajectory (CNT).

Definition 1: A nominal trajectory is a *compatible nominal trajectory* for system (47) if $x_k^r = x_k^c$ and $u_k^r = u_k^c$ for $k = 0, \ldots, N-1$.

Linearizing along a compatible nominal trajectory ensures that the nonlinear system is linearized at the "correct" points. When applying the designed control to the linearized system, the resulting trajectory is the same as the nominal trajectory, which means that the system reaches the exact points where the linearization is performed. On the other hand, there will be extra errors caused by the linearization if the system is linearized along a non-compatible nominal trajectory.

The iterative algorithm to find a compatible nominal trajectory is given in Algorithm 1. First, we initialize the algorithm with an initial guess for the nominal trajectory. Next, we linearize the nonlinear system along this nominal trajectory. By solving the mean control problem (27) using (29), we obtain the mean trajectory. Then, we update the nominal trajectory using the mean trajectory. We repeat the above three steps until converge or a stop criterion is satisfied. A commonly used convergence criterion is to check if the change of $(x_k^r)_{k=0}^{N-1}$ between successive iterations is smaller than a small threshold.

Algorithm 1: Compatible Nominal Trajectory (CNT)

```
Initialize (x_k^r)_{k=0}^{N-1} and (u_k^r)_{k=0}^{N-1} ;
while NotConverged do
    Linearize (47) along (x_k^r)_{k=0}^{N-1} and (u_k^r)_{k=0}^{N-1} ;
    Compute the mean optimal control using (29) to
      obtain the control sequence (u_k^c)_{k=0}^{N-1} ;
    Compute the the controlled state trajectory
      (x_k^c)_{k=0}^{N-1} using (24) ;
    Set (x_k^r)_{k=0}^{N-1} = (x_k^c)_{k=0}^{N-1} and
      (u_k^r)_{k=0}^{N-1} = (u_k^c)_{k=0}^{N-1} ;
return (x_k^r)_{k=0}^{N-1} and (u_k^r)_{k=0}^{N-1};
```

When the algorithm converges, the nominal trajectory is the same as the mean optimal trajectory, which, by definition, is a compatible nominal trajectory. Note that (29) gives the analytical solution of the mean control, which can be quickly computed. Each iteration of the algorithm has a low computational load and the overall algorithm may be solved efficiently. As seen later in Section VI, the algorithm typically converges within a few iterations.

The next result shows how to choose the reference state trajectory $(m_k)_{k=0}^{N-1}$ in (6).

Proposition 2: Consider the linear time-varying system (3). The cost function given in (12) achieves the minimum only

if the reference trajectory $(m_k)_{k=0}^{N-1}$ is equal to the mean trajectory of the states $(\bar{x}_k)_{k=0}^{N-1}$.

Proof. See Appendix C. □

Choosing the reference state trajectory $(m_k)_{k=0}^{N-1}$ judiciously has the benefit of achieving a lower cost. It should be noted that the cost functional (6) regulates the state trajectories to stay close to $(m_k)_{k=0}^{N-1}$. By setting $m_k = x_k^r$ for all $k = 0, \ldots, N-1$, the resulting state trajectory is regulated to be close to the nominal trajectory, and thus the linearization of the nonlinear system is more likely to be locally valid, which will help the stability of Algorithm 1.

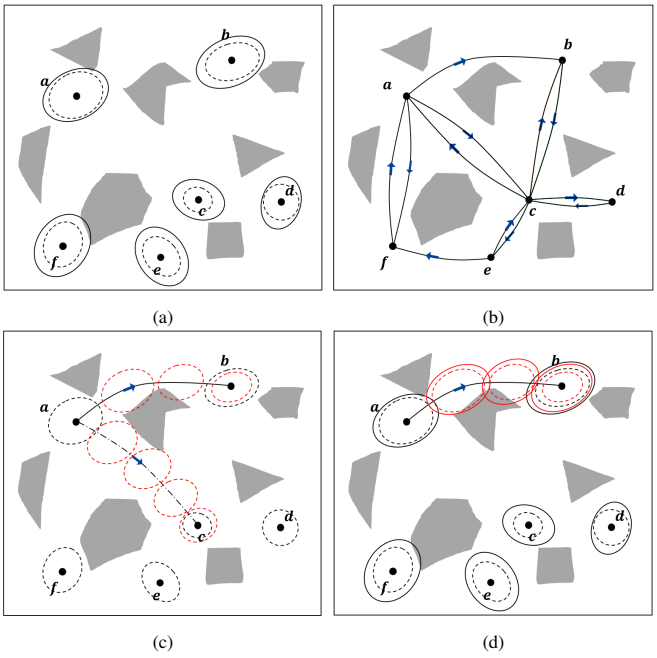


Fig. 2: Construction of the CS-BRM. Gray shapes denote obstacles. (a) Node sampling; (b) Mean trajectories are computed for all neighboring node pairs using the mean controller. Only mean trajectories that are collision-free are preserved. Those mean trajectories indicate possible connections/edges in the CS-BRM; (c) Kalman filter updates are simulated for each possible edge; (d) Finally, covariance control is applied for each edge to execute the transition between the nodes.

V. The CS-BRM Algorithm

The nodes in the CS-BRM are sampled in the belief space. In a partially observable environment, the belief b_k at time-step k is given by the probability distribution of the state x_k conditioned on the history of observations $(y_i)_{i=0}^{k}$ and the history of control inputs $(u_i)_{i=0}^{k-1}$, that is, $b_k = \mathbb{P}(x_k | (y_i)_{i=0}^{k}, (u_i)_{i=0}^{k-1})$. In a Gaussian belief space, b_k can be equivalently represented by the estimated state, $\hat{x}_k$, and the estimation error covariance, $\tilde{P}_k$, that is, $b_k = (\hat{x}_k, \tilde{P}_k)$ [11], [30]. The state estimate $\hat{x}_k$ is Guassian, and is given by $\hat{x}_k \sim \mathcal{N}(\bar{x}_k, \hat{P}_k)$. Hence, the Gaussian belief can be also written as $b_k = (\bar{x}_k, \hat{P}_k, \tilde{P}_k)$. The main idea of CS-BRM is to use covariance steering theory to design the edge controller to achieve node reachability.

An illustration of the steps for building the CS-BRM is shown in Figure 2. The algorithm for constructing the CS-BRM is given in Algorithm 2. The following procedures are used in the algorithm.

Algorithm 2: Constructing CS-BRM

1 $V = \{\mathrm{nod}_1, \ldots, \mathrm{nod}_n\} \leftarrow \mathsf{SampleNodes}(n)$;
2 $V_r \leftarrow V$, $E \leftarrow \emptyset$;
3 **for** $i = 1 : n$ **do**
4 $V_{\mathrm{near}} \leftarrow \mathsf{Neighbor}(V_r, \mathrm{nod}_i)$;
5 **foreach** $\mathrm{nod}_j \in V_{\mathrm{near}}$ **do**
6 $(\bar{U}_{ij}, \tau_{ij}, \mathrm{MCost}_{ij}) \leftarrow \mathsf{MTraj}(\mathrm{nod}_i, \mathrm{nod}_j)$;
7 **if** $\mathsf{ObstacleFree}(\tau_{ij})$ **then**
8 $\tilde{P}_{N^-} \leftarrow \mathsf{KF}(\mathrm{nod}_i, \mathrm{nod}_j)$;
9 **if** $\tilde{P}_{N^-} \preceq \tilde{P}_{\mathrm{nod}_j}$ **then**
10 $(\tilde{U}_{ij}, \mathrm{CovCost}_{ij}) \leftarrow$ $\mathsf{CovControl}(\mathrm{nod}_i, \mathrm{nod}_j)$;
11 $\mathrm{CollisionCost}_{ij} \leftarrow$ $\mathsf{MonteCarlo}(\bar{U}_{ij}, \tilde{U}_{ij})$;
12 $E_{ij} \leftarrow (\bar{U}_{ij}, \tilde{U}_{ij}, \mathrm{EdgeCost}_{ij})$;
13 $E \leftarrow E \cup E_{ij}$
14 $(\bar{U}_{ji}, \tau_{ji}, \mathrm{MCost}_{ji}) \leftarrow \mathsf{MTraj}(\mathrm{nod}_j, \mathrm{nod}_i)$;
15 **if** $\mathsf{ObstacleFree}(\tau_{ji})$ **then**
16 Repeat lines 8-13 with i and j swapped
17 $V_r \leftarrow V_r \setminus \mathrm{nod}_i$;
18 CS-BRM $\leftarrow (V, E)$;
19 **return** CS-BRM;

Sample Nodes: The function $\mathsf{SampleNodes}(n)$ samples n CS-BRM nodes. A node in the CS-BRM is represented by the tuple $(\bar{x}, P, \tilde{P})$. Since $P = \hat{P} + \tilde{P}$, the node can also be equivalently represented by $(\bar{x}, \hat{P}, \tilde{P})$. Note that we can compute the a posteriori covariances $\hat{P}$ and $\tilde{P}$ using $\hat{P}^-$ and $\tilde{P}^-$. Thus, we may also represent the node by $(\bar{x}, \hat{P}, \tilde{P})$. In Figure 2(a), for each node, $\bar{x}$ is shown as a black dot, P is shown as a solid ellipse, and $\tilde{P}^-$ is shown as a dashed ellipse.
Neighbor: The function $\mathsf{Neighbor}(V_r, \mathrm{nod}_i)$ finds all the nodes in V_r that are within a given distance d_1 to node nod_i, where V_r is a node-set containing all nodes in the CS-BRM. Similar to the traditional PRM, every node tries to connect with other nodes in the graph that are within a distance of itself. We use the Wasserstein distance [46] to compute the distance between BRM nodes. Specifically, given two nodes nod_i and nod_j representing the Gaussian distributions $(\bar{x}_i, P_i)$ and $(\bar{x}_j, P_j)$, their distance D_{ij} is computed by

$$D_{ij} = \left[\|\bar{x}_i - \bar{x}_j\|_2^2 + \mathrm{trace}\left(P_i + P_j - 2(P_j^{\frac{1}{2}} P_i P_j^{\frac{1}{2}})^{\frac{1}{2}} \right) \right]^{\frac{1}{2}}.$$

Mean Trajectory: Given nod_i and nod_j, $\mathsf{MTraj}(\mathrm{nod}_i, \mathrm{nod}_j)$ uses the mean controller (29) and Algorithm 1 to find the compatible nominal trajectory, which is also the mean trajectory from nod_i to nod_j. The function returns the mean control $\bar{U}_{ij}$, mean trajectory τ_{ij} from nod_i to nod_j, and the mean control cost MCost_{ij}.
Obstacle Checking: The function $\mathsf{ObstacleFree}(\tau_{ij})$ checks if the trajectory τ_{ij} is collision free.
Kalman Filter : Given two nodes nod_i and nod_j, $\mathsf{KF}(\mathrm{nod}_i, \mathrm{nod}_j)$ returns the state estimation error covariance

at every time step along that edge.

Covariance Control: The function $\mathsf{CovControl}(\mathrm{nod}_i, \mathrm{nod}_j)$ solves the covariance control problem from nod_i to nod_j. It returns the control $\tilde{U}_{ij}$ and the cost $\mathrm{CovCost}_{ij}$.

Monte Carlo: We use Monte Carlo simulations to calculate the probability of collision of the edges. For edge E_{ij}, the initial state x_0 and initial state estimate $\hat{x}_{0\text{-}}$ are sampled from their corresponding distributions. The state trajectory is simulated using the mean control $\bar{U}_{ij}$ and the covariance control $\tilde{U}_{ij}$. Then, collision checking is performed on the simulated state trajectory. By repeating this process, we approximate the probability of collision of this edge. The collision cost $\mathrm{CollisionCost}_{ij}$ is taken to be proportional to the probability of collision along that edge.

Different methods may be used for sampling the belief in the $\mathsf{SampleNodes}$ command. First, sampling the mean $\bar{x}$ is the same as in PRM and RRT, where we uniformly randomly sample the free state space. For sampling the covariances P and $\tilde{P}_{\text{-}}$, we may restrict the covariances to diagonal matrices and sample positive real numbers for the diagonal entries. Then each entry on the diagonal is sampled uniformly from the interval $[\lambda_{\min}, \lambda_{\max}]$ for some $\lambda_{\min}$ and $\lambda_{\max}$. These bounds of the intervals are tuning parameters and they depend on the planning environment. Alternatively, we can sample the covariance from the positive definite matrix space $P = QRQ^\top$ by sampling the diagonal matrix R and the orthogonal matrix Q [47]. Yet another way of sampling the beliefs is to utilize the results from single-query belief space planning methods [12], [30]. In that approach, the beliefs are obtained using the RRT algorithm along with the use of LQG controllers. In our implementation of CS-BRM, we run the single-query belief space planners to obtain the beliefs. Then, we used those beliefs to construct belief roadmaps.

Algorithm 2 starts by sampling n nodes in the belief space using $\mathsf{SampleNodes}$ (Line 1). Lines 3-17 are the steps to add CS-BRM edges. Given two neighboring nodes nod_i and nod_j, Lines 6-13 try to construct the edge E_{ij} and Lines 14-16 try to construct the edge E_{ji}.

Every edge in the CS-BRM is constructed by solving a covariance steering problem. Each node tries to connect to its neighboring nodes, if possible. First, for each edge, we use Algorithm 1 to find the mean trajectory of that edge (Line 6). Next, we check if the mean trajectory is collision-free (Line 7). Then, the Kalman filter updates are simulated, which gives the state estimation error covariance at every time step along that edge. For edge $\overline{ab}$ shown in Figure 2(c), the initial condition of the Kalman filter is $\tilde{P}_{0\text{-}} = \tilde{P}_{a\text{-}}$. The prior estimation error covariance at the final time-step, $\tilde{P}_{N\text{-}}$, is compared with the state estimation error covariance of node b, $\tilde{P}_{b\text{-}}$. If $\tilde{P}_{N\text{-}} \preceq \tilde{P}_{b\text{-}}$ (Line 9), the algorithm proceeds to solve the covariance control problem, and this edge is added to the graph. In Figure 2(c), $\overline{ab}$ will be added as a CS-BRM edge, while $\overline{ac}$ will not. For the covariance control problem of edge $\overline{ab}$, the initial constraint and the terminal constraint are given by $\hat{P}_{0\text{-}} = \hat{P}_{a\text{-}}$ and $\hat{P}_N \preceq \hat{P}_b$. The covariance $\hat{P}_b$ is computed using $\hat{P}_{b\text{-}}$, $\tilde{P}_{b\text{-}}$, first equation in (9), and (15). The final edge controller is the combination of the mean controller, covariance controller, and the Kalman filter.

In addition to the edge controller, the edge cost is computed for each edge. In order to provide more accurate and faster ways to compute collision probabilities, different methods have been developed [11], [48], [49]. Monte Carlo simulations provide more accurate collision probabilities at the cost of expensive computations. Since the roadmap is constructed offline, computation is not an issue for our method. The edge cost $\mathrm{EdgeCost}_{ij}$ is a weighted sum of MCost_{ij}, $\mathrm{CovCost}_{ij}$, and $\mathrm{CollisionCost}_{ij}$. With covariance steering serving as the lower-level controller, the higher-level motion planning problem using the roadmap is a graph search problem similar to a PRM. Thus, the covariance steering approach transforms the belief space roadmap into a traditional PRM with specific edge costs.

A. Node Consistency

In CS-BRM, each edge is an independent covariance steering problem and the planned path using CS-BRM consists of a concatenation of edges. Since the terminal estimated state covariance satisfies an inequality constraint given by (45) and the terminal state estimation error covariance satisfies an inequality relation by construction (Line 9), it is important to verify that the covariance constraints at all nodes are still satisfied by concatenating the edges.

To this end, consider the covariance steering problem from node a to node b. For the edge $\overline{ab}$, we have the initial constraints $\hat{P}_{0\text{-}} = \hat{P}_{a\text{-}}$, $\tilde{P}_{0\text{-}} = \tilde{P}_{a\text{-}}$, and the terminal covariance constraint $\hat{P}_N \preceq \hat{P}_b$. Note that the constraint $\tilde{P}_{N\text{-}} \preceq \tilde{P}_{b\text{-}}$ is satisfied for $\overline{ab}$ (Line 9, Algorithm 2). Suppose that the solution of the covariance control problem of edge $\overline{ab}$ results in the feedback gain K as in (32). By concatenating the edges, the system may not start exactly at node a. Instead, it will start at some node $a' = (\bar{x}_a, \hat{P}'_{a\text{-}}, \tilde{P}'_{a\text{-}})$ that satisfies $\hat{P}'_{a\text{-}} \preceq \hat{P}_{a\text{-}}$ and $\tilde{P}'_{a\text{-}} \preceq \tilde{P}_{a\text{-}}$. Next, we show that by applying the pre-computed feedback gain K, the terminal covariance still satisfies $\hat{P}'_N \preceq \hat{P}_b$ for the new covariances $\hat{P}'_{a\text{-}}$ and $\tilde{P}'_{a\text{-}}$. Before verifying this result, we study the propagation of the estimation error covariance $\tilde{P}_{k\text{-}}$ using the Kalman filter. Given $\tilde{P}_{0\text{-}}$, we obtain a sequence of error covariances $(\tilde{P}_{k\text{-}})_{k=1}^N$ using the Kalman filter updates (10). Similarly, given the new initial error covariance $\tilde{P}'_{0\text{-}}$, we obtain a new sequence of error covariances $(\tilde{P}'_{k\text{-}})_{k=0}^N$. We have the following proposition.

Proposition 3: If $\tilde{P}'_{0\text{-}} \preceq \tilde{P}_{0\text{-}}$, then $\tilde{P}'_{k\text{-}} \preceq \tilde{P}_{k\text{-}}$, for all $k = 0, \cdots, N$.

Proof. See Appendix D. □

From Proposition 3, it is straightforward to show that, if $\tilde{P}'_{0\text{-}} \preceq \tilde{P}_{0\text{-}}$, we also have $\tilde{P}'_k \preceq \tilde{P}_k$ for all $k = 0, \ldots, N$.

Proposition 4: Consider a path on the CS-BRM roadmap, where the initial node of the path is denoted by $(\bar{x}_i, \hat{P}_i, \tilde{P}_i)$ and the final node of the path is denoted by $(\bar{x}_j, \hat{P}_j, \tilde{P}_j)$. Starting from the initial node and following this path by applying the sequence of edge controllers, the robot will arrive at a belief node $(\bar{x}, \hat{P}, \tilde{P})$ such that $\bar{x} = \bar{x}_j$, $\hat{P} \preceq \hat{P}_j$, and $\tilde{P} \preceq \tilde{P}_j$.

Proof. Consider the edge $\overline{ab}$ from node a to node b. From the solution of the covariance steering problem, we have $\hat{P}_N \preceq \hat{P}_b$ and $\tilde{P}_N \preceq \tilde{P}_b$. We only need to show that, with the new initial estimated state covariance $\hat{P}'_{a\text{-}}$ satisfying $\hat{P}'_{a\text{-}} \preceq \hat{P}_{a\text{-}}$

and the new initial estimation error covariance $\tilde{P}'_{a^-}$ satisfying $\tilde{P}'_{a^-} \preceq \tilde{P}_{a^-}$, and applying the pre-computed feedback gain K_{ab}, the terminal covariances satisfies $\hat{P}'_N \preceq \hat{P}_N$ and $\tilde{P}'_N \preceq \tilde{P}_N$. We use $\hat{P}'_N$ to denote the new value of $\hat{P}_N$ that corresponds to the new initial covariances. This notation also applies to other variables in the following derivation.

From Proposition 3 we have $\tilde{P}'_{N^-} \preceq \tilde{P}_{N^-}$ and $\tilde{P}'_N \preceq \tilde{P}_N$. Next, we show that $\hat{P}'_k \preceq \hat{P}_k$ for $k = 0, 1, \cdots, N$. Recall that $\check{x}_k = \hat{x}_k - \bar{x}_k$, thus $\check{P}_k = \hat{P}_k$. From (39) and (41), we have

$$\begin{aligned}\hat{P}_k &= E_k P_{\check{X}} E_k^\top \\ &= E_k(I+BF)(A\hat{P}_{0^-}A^\top + LP_\Xi L^\top)(I+BF)^\top E_k^\top,\end{aligned} \tag{48}$$

where P_Ξ is given by (40). Using (16) and the expression of L_k in (10), we get

$$L_k P_{\xi_k} L_k^\top = \tilde{P}_{k^-} C_k^\top (C_k \tilde{P}_{k^-} C_k^\top + D_k D_k^\top)^{-1} C_k \tilde{P}_{k^-}, \tag{49}$$

which is a non-decreasing function of $\tilde{P}_{k^-}$. Thus, $L'_k P'_{\xi_k} L'^\top_k \preceq L_k P_{\xi_k} L_k^\top$ and $L' P'_\Xi L'^\top \preceq L P_\Xi L^\top$. Therefore, given $\tilde{P}'_{0^-} \preceq \tilde{P}_{0^-}$ and $\hat{P}'_{0^-} \preceq \hat{P}_{0^-}$, we have $\tilde{P}'_{k^-} \preceq \tilde{P}_{k^-}$ and $\hat{P}'_k \preceq \hat{P}_k$ for $k = 0, 1, \cdots, N$. Thus, $\hat{P}'_N \preceq \hat{P}_N$, which completes the proof. □

Proposition 4 guarantees that, when planning on the CS-BRM roadmap, the covariance at each arrived node is always smaller than the assigned fixed covariance at the corresponding node of the CS-BRM roadmap.

B. Velocity Space Sampling

One main advantage of CS-BRM is that the nodes do not need to be stationary. Instead, CS-BRM allows exploring the velocity space and finds paths with lower cost. In this section, we introduce a heuristic method for sampling the velocity space. Sampling the position space and sampling the velocity space are independent events. Exploring the velocity space is important to achieve lower path costs and find smoother plans.

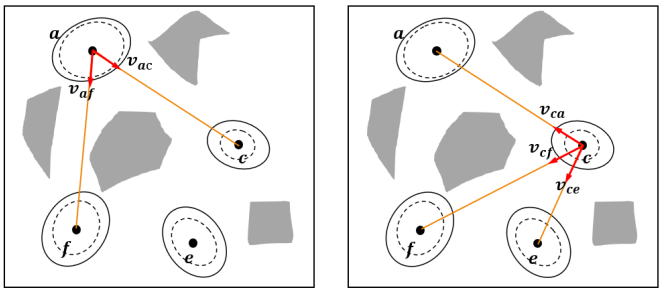


Fig. 3: Heuristic method for sampling the velocity space. Multiple velocities are sampled at each sampled position. For example, v_{af} and v_{ac} are sampled at vertex a, which results in the nodes n_{af} and n_{ac}. v_{ca}, v_{cf}, and v_{ce} are sampled at vertex c, which results in the nodes n_{ca}, n_{cf}, and n_{ce}. When constructing the edges, the edges connecting n_{ac} to n_{ca}, n_{cf}, and n_{ce} will be constructed, but there will be no edges that connect n_{af} to n_{ca}, n_{cf}, or n_{ce}.

The proposed velocity sampling heuristic consists of two procedures: graph-structure-based heading sampling and selective connection. We sample multiple velocities at each position based on the local graph structure. The method is illustrated in Figure 3. For each neighboring vertex j of vertex i, a velocity v_j is sampled for vertex i. In Figure 3, vertex a has two neighbors, vertex f and vertex c. We assign two velocities v_{af} and v_{ac} to vertex a. The directions of v_{af} and v_{ac} are decided by the straight lines af and ac, respectively. The magnitudes of v_{af} and v_{ac} are sampled randomly in an interval. Similarly, three velocities v_{ca}, v_{cf}, and v_{ce} are sampled for vertex c. This process is repeated for all vertices in the map. After finishing with this velocity sampling process, there will be two nodes at vertex a. We denote them as n_{af} and n_{ac}, respectively, where n_{af} corresponds to the node with velocity v_{af} and n_{ac} corresponds to the node with velocity v_{ac}. Similarly, there are three nodes n_{ca}, n_{cf}, and n_{ce} at vertex c.

The second important aspect of this method is selective connection. In this example, only node n_{ac} will be connected to the nodes at vertex c, and node n_{af} will not be connected to any node at vertex c. Similarly, only n_{ca} will be connected to n_{ac} and n_{af}, while nodes n_{cf} and n_{ce} will not be connected to n_{ac} or n_{af}. This strategy avoids many undesired connections. In summary, the proposed heuristic method explores the velocity space without adding too many belief nodes and edges.

VI. Examples

In this section, we first illustrate our theoretical results for the motion planning problem of a 2-D double integrator. We compare our method with the SLQG-FIRM algorithm [11] and show that our method overcomes some of the limitations of SLQG-FIRM, resulting in more efficient plans. Subsequently, the problem of a fixed-wing aerial vehicle in a 3-D environment is studied. The theory is finally experimentally demonstrated on a small quadrotor flying indoors between obstacles.

A. 2-D Double Integrator

A 2-D double integrator is a linear system with a 4-dimensional state space $x_k = [x^{(1)}\ x^{(2)}\ x^{(3)}\ x^{(4)}]^\top$ and a 2-dimensional control input space $u_k = [a_x\ a_y]^\top$. The dynamics of the system is given in (3) with the system matrices given by

$$\begin{aligned}A_k &= \begin{bmatrix}1 & 0 & \Delta t & 0\\ 0 & 1 & 0 & \Delta t\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}, \quad B_k = \begin{bmatrix}\Delta t^2/2 & 0\\ 0 & \Delta t^2/2\\ \Delta t & 0\\ 0 & \Delta t\end{bmatrix},\\ G_k &= \mathrm{diag}(5,8,5,5)\times 10^{-2}, \quad h_k = [0\ 0\ 0\ 0]^\top,\end{aligned} \tag{50}$$

where Δt is the time step size, I_m denotes the identity matrix with dimension m.

1) Comparison of Edge Controllers: One example of computing the nominal trajectory is given in Figure 4. Since the system is linear, the mean controller finds the nominal trajectory in one iteration given a fixed reference trajectory $(m_k)_{k=0}^{N-1}$. The iterations in Figure 4 result owing to updating the reference trajectory at each iteration. The reference state trajectory is initialized as a straight line in all 4 dimensions. Based on the current reference trajectory, the mean control and the mean trajectory are computed. Then, the computed mean trajectory is used to update the reference trajectory and this

process is repeated until convergence. In Figures 4(a) and 4(b), the red lines correspond to the final nominal trajectory, which is also the final reference trajectory. The algorithm converged after 6 iterations. The time step size is $\Delta t = 0.2$ s, the number of steps $N = 18$, and the weights $Q_k = 4I_4$ and $R_k = 2I_2$.

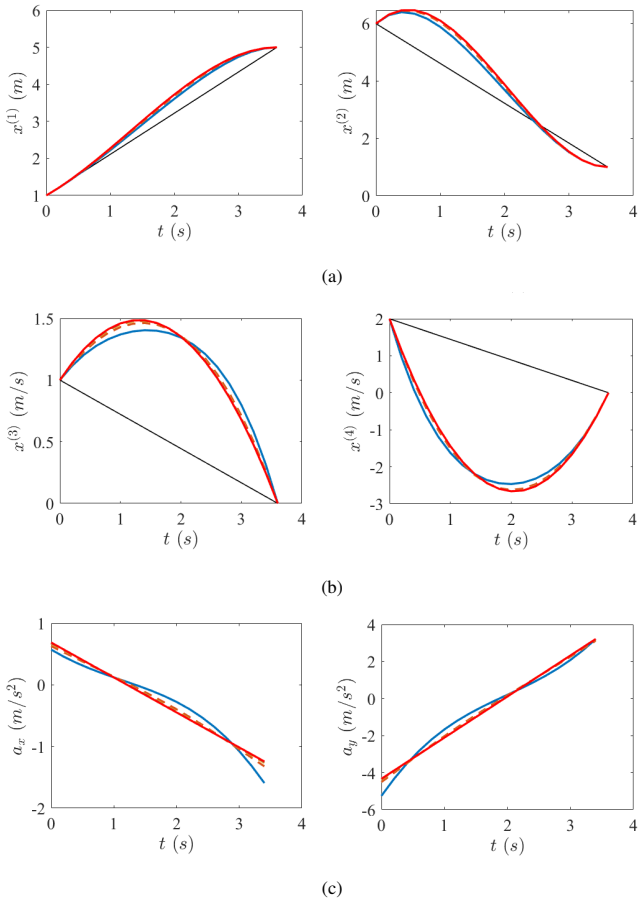


Fig. 4: Results of computing the reference trajectory for the 2-D double integrator. The reference state trajectory is initialized as a straight line in all 4 dimensions. The state trajectories are shown in (a) and (b). The control trajectories are shown in (c).

Next, we compare the proposed edge controller with the edge controller in [11]. For this purpose, we use a simple observation model with the matrices in (4) given by $C_k = I_4$ and $D_k = \text{diag}(10, 5, 10, 5) \times 10^{-2}$. Later, we consider a different observation model where the position of the robot is measured with respect to landmarks in the environment and D_k is a time-varying matrix. The edge controller in [11] is a concatenation of a time-varying LQG controller with an SLQG controller. A dynamically feasible nominal trajectory is required for the time-varying LQG controller. We use the same nominal trajectory from our method for the time-varying LQG controller. The nominal trajectory is also the mean trajectory from the time-varying LQG controller [30]. Thus, the time-varying LQG controller and covariance steering controller have the same mean trajectory.

The belief is defined as $b = (\hat{x}, \tilde{P})$ in [11], where $\hat{x}$ is the estimated state and $\tilde{P}$ is the estimation error covariance. Note that $\hat{x}$ is a random vector. Since the belief is a distribution over the state, we can also define the belief as $b = (\bar{x}, P, \tilde{P})$, where

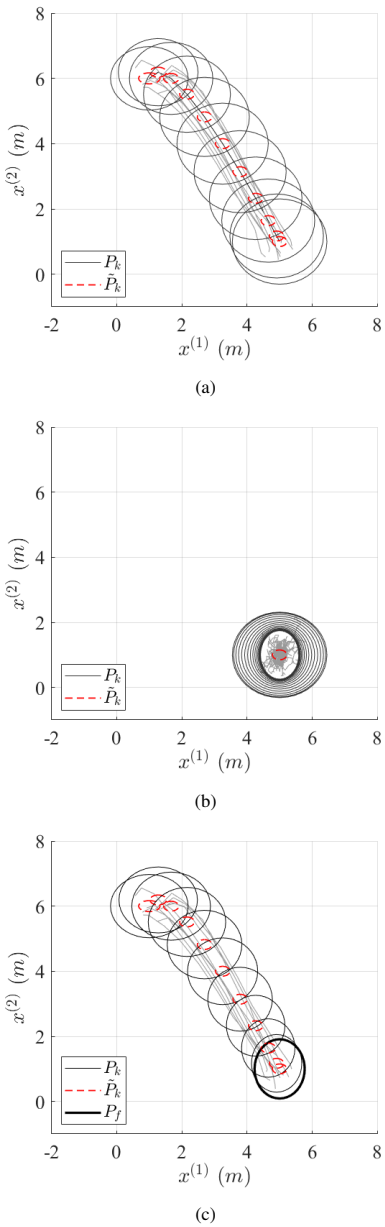


Fig. 5: Comparison of the edge controller in [11] and the proposed edge controller using covariance steering. (a) Time-varying LQG, (b) Time-varying LQG is switched to SLQG to reach a stationary node. (c) Result of our method using covariance steering. The red dash ellipses and the black ellipses show the 3σ confidence intervals of the estimation error covariances and the state covariances respectively.

$\bar{x}$ is the mean, P is the state covariance and $\bar{x}$ is deterministic. The two definitions are equivalent, as the state covariance P includes the estimation error covariance $\tilde{P}$ and the estimated state covariance $\hat{P}$ [30]. For comparison, we will use the same belief definition $b = (\bar{x}, P, \tilde{P})$ for both methods. In [11], the convergence of belief is indicated by the convergence of both $\hat{x}$ and $\tilde{P}$. In this paper, the convergence of belief is indicated by the convergence of both P and $\tilde{P}$. The details of computing the state distributions along the trajectory under the LQG controller are given in [12], [30].

The results of the edge controllers for the two methods are shown in Figure 5. The mean velocity at the endpoint

is set to zero as required by SLQG-FIRM. The edge controller in [11] is obtained using two steps: a time-varying LQG controller (Figure 5(a)) and a stationary LQG controller (Figure 5(b)). From Figures 5(a) and 5(b), we can see that the estimation error covariance (red ellipses) converged to a neighborhood of the stationary covariance in a few steps. The state covariance (black ellipses) converged at a much later time. Figure 5(c) is the result of our method based on covariance steering. In both examples, the initial state mean is $[1\ 6\ 1\ 2]^\top$ and the terminal state is $[5\ 1\ 0\ 0]^\top$. The initial state covariance is $P_0 = \text{diag}(12, 8, 8, 8) \times 10^{-2}$, the initial prior estimation error covariance is $\tilde{P}_{0^-} = 0.8P_0$. The terminal state covariance constraint for the covariance steering method is $P_f = \text{diag}(5, 7, 4, 4) \times 10^{-2}$.

Compared to [11], our method is not restricted to sampling zero velocity and it does not require a converging phase. In this example, the time duration of the time-varying LQG controller (Figure 5(a)) and the time duration of the covariance steering controller (Figure 5(c)) are the same. Since the edge controller in SLQG-FIRM requires a converging phase, the total time required for the edge controller in SLQG-FIRM to reach a node is longer than the CS-BRM method.

2) CS-BRM for the 2-D double integrator: We first provide a detailed description of each procedure of the CS-BRM method. In order to compare with SLQG-FIRM, we first restrict our method to sample nodes with zero mean velocity.

The environment considered is shown in Figure 6. There are ℓ landmarks placed in the environment shown as black stars. The black polygonal shapes represent the obstacles. The agent observes all landmarks and obtains estimates of its own position at all time steps. The agent achieves better position estimates when it is closer to the landmarks. Let the Euclidean distance between the position of the 2-D double integrator and the j^{th} landmarks be given by d_j. Then, the j^{th} position measurement corresponding to landmark j is

$$^j y = [x^{(1)}\ x^{(2)}]^\top + \eta_p d_j v_p, \quad j = 1, 2, \cdots, \ell, \tag{51}$$

where η_p is a parameter related to the intensity of the position measurement noise that is set to 0.1, and v_p is a two-dimensional standard Gaussian random vector. The velocity measurement is given by

$$y_v = [x^{(3)}\ x^{(4)}]^\top + \eta_v v_v, \tag{52}$$

where η_v is a parameter related to the intensity of the velocity measurement noise and is set to 0.2, and v_v is a two-dimensional standard Gaussian random vector. In this case, the velocity is measured using onboard sensors and does not depend on the landmarks. Thus, the total measurement vector y is a $2\ell + 2$ dimensional vector, composed of ℓ position measurements and one velocity measurement.

The sampled CS-BRM nodes are shown in Figure 6. Each node is given by $(\bar{x}, \hat{P}, \tilde{P}^-)$. The mean of the positions $x^{(1)}$, $x^{(2)}$ are sampled from the obstacle-free space. The velocities are all set to be zero for the purpose of comparison with SLQG-FIRM.

Following Algorithm 2, the edge controllers are computed using the covariance steering controller for each edge and the collision cost is computed using Monte Carlo simulations. For each node in the roadmap, the mean trajectories to go from this node to all its neighboring nodes are computed using Algorithm 1. In the covariance steering method introduced in Section III, the number of time-steps N for a particular edge, and the time step size Δt are design variables chosen by the user. In this example, we choose $\Delta t = 0.2$ s. The time duration of the edge is chosen based on the Euclidean distance between the state mean of the two endpoints of that edge. By specifying the desired average speed of the robot, we approximate the time duration of the edge and then obtain the number of time-steps N of this edge. The desired average speed is a design parameter and is set to be 4 m/s.

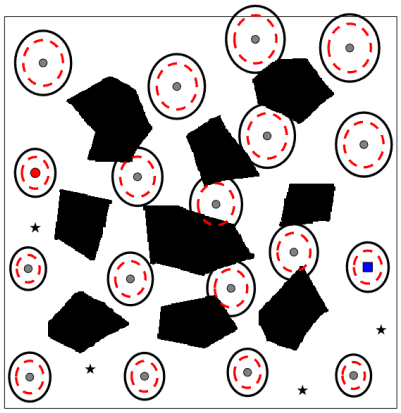


Fig. 6: Planning environment and sampled CS-BRM nodes. The gray dots are the state means $\bar{x}$ of the nodes. The red dot is the state mean of the starting node and the blue square is the state mean of the goal node. The black ellipses around the state represent the 3σ confidence intervals of the position covariances, and the red dash ellipses correspond to the prior estimation error covariances. The goal of the planning problem is to find the optimal path along with the control policy to go from the starting node to the goal node.

The edge cost in CS-BRM is the combination of the mean control cost, the covariance control cost, and the cost from the probability of collision. After the CS-BRM is built, the path planning problem on CS-BRM is the same as the problem of planning using a PRM, which can be easily solved using a graph search algorithm. The difference between the CS-BRM and PRM is that the edge cost in CS-BRM is specifically designed to deal with dynamical system models and uncertainties, and the transition between two nodes of CS-BRM is achieved using covariance steering as the edge controller.

The planned path using our method is shown in Figure 7. The edge cost in CS-BRM is a weighted sum of the mean control cost, covariance control cost, and collision cost. Instead of taking the shortest length path which has a high probability of collision, the algorithm finds a detoured path that is optimal in terms of the defined edge cost.

3) Comparison of Belief Space Roadmap Methods: In this section, we compare the proposed CS-BRM algorithm with the SLQG-FIRM. The sampled belief nodes for SLQG-FIRM are the same as those in Figure 6. Since the sampled state covariance at each node is larger than the corresponding stationary state covariance (which is obtained using the SLQG controller), we determine that the robot has reached a node if, after switching from the time-varying LQG to the SLQG,

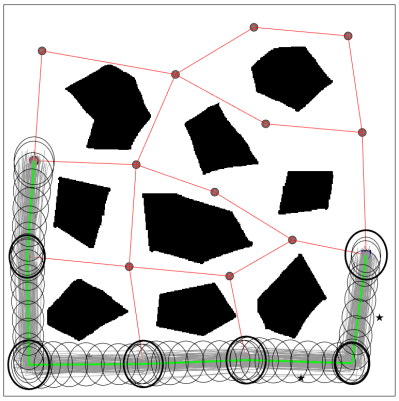


Fig. 7: Path planned by CS-BRM. The path is optimal in terms of the defined edge cost, which is a weighted cost between the control effort and safety.

the state covariance under the control of the SLQG controller converges to a value that is smaller than the sampled state covariance at that node. The planned path is shown in Figure 8.

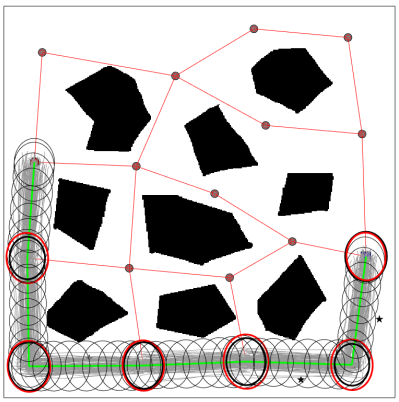


Fig. 8: Planned path using the SLQG-FIRM. The path goes through the same set of belief nodes as those in Figure 7. The red ellipses are the state covariances at the last step of the time-varying LQG controller. The time-varying LQG controller is switched to the SLQG controller until the state covariance converges to a value that is smaller than the state covariance of the belief nodes (bold black ellipses). After that, the path proceeds to the next node.

Both CS-BRM and SLQG-FIRM choose a path that goes through the same set of belief nodes. In both cases, the collision cost is zero because no collision is detected by the Monte Carlo simulations. In Figure 8, the red ellipses are the state covariances at the last step of the time-varying LQG controller. As we see, each one of the ellipses is larger than the corresponding state covariance of the intermediate belief node (bold black ellipses). Thus, a converging step is required at every intermediate node, which means that the path from the SLQG-FIRM method will take more time to reach the goal node compared to the one in Figure 7. The cost of the path planned by CS-BRM in Figure 7 is 244.61. When only considering the cost of the time-varying LQG controller and ignoring the cost of the SLQG controller, the cost of the path planned by SLQG-FIRM in Figure 8 is 240.47. After adding the cost from the SLQG controller, the final cost is 246.13. The mean control cost of the paths in Figures 7 and 8 are the same because CNT is used to compute the mean trajectories in both cases. Also, the collision cost in both cases is zero as no collisions are detected from the Monte Carlo simulations. Thus, the total cost of the paths planned using CS-BRM and SLQG-FIRM are close in this case.

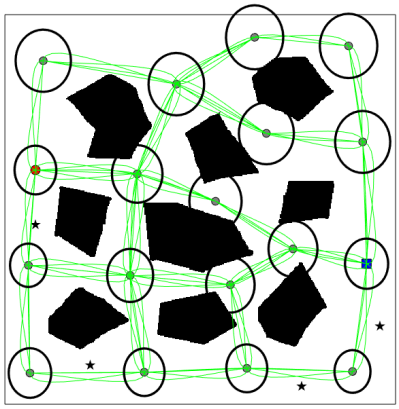


Fig. 9: CS-BRM results based on the heuristic method to sample velocity. The green lines are the mean trajectories of the edges.

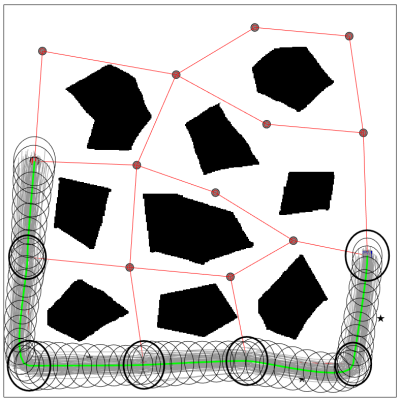


Fig. 10: Planning results using the CS-BRM algorithm and the proposed heuristic method for sampling velocities. The path is optimal in terms of the defined edge cost, which is a weighted sum of the control effort and safety. By sampling the velocity space, the robot does not need to stop at every node. The path cost is much smaller in this case compared to Figures 8.

We also used the method of Section V-B to sample the velocity space. The mean positions, state covariances, and estimation error covariances are the same as those in Figure 6. The only difference is that instead of assigning zero mean velocity to every position, multiple non-zero velocities are sampled for each position. The results are shown in Figure 9. The mean trajectories of the edges are shown as green lines. The ellipses are the state covariances. The planned path is shown in Figure 10. The cost of the planned path is 104.87, which is much lower than the cost of the paths in Figure 7 and Figure 8, which are 244.61 and 246.13 respectively. The decrease in the path cost is due to the decrease in the mean control cost. In Figure 8, the robot has to stop (zero velocity) at every intermediate node. With the proposed method, the robot goes through each intermediate node smoothly, which results in a more efficient and low-cost path.

B. Fixed-Wing Aircraft

In this section, we apply the proposed method to a nonlinear example, namely, planning for a fixed-wing aerial vehicle. The kinematic model of a fixed-wing unmanned aerial vehicle (UAV) is given by [50]

$$\begin{aligned} \dot{x} &= V\cos\psi\cos\gamma, \\ \dot{y} &= V\sin\psi\cos\gamma, \\ \dot{z} &= V\sin\gamma, \\ \dot{\psi} &= \frac{g}{V}\tan\phi, \end{aligned} \tag{53}$$

where the 4-dimensional state space is given by $[x\ y\ z\ \psi]^\top$, where $[x\ y\ z]^\top$ is the 3-D position of the vehicle, and ψ is the heading angle. The 3-D control input space is given by $[V\ \gamma\ \phi]^\top$, where V is the air speed, γ is the flight-path angle, and ϕ is the bank angle.

The discrete-time system model with noise added is given by

$$\begin{aligned} x_{k+1} &= x_k + V_k\cos\psi_k\cos\gamma_k\Delta t + g_{1k}w_{1k}, \\ y_{k+1} &= y_k + V_k\sin\psi_k\cos\gamma_k\Delta t + g_{2k}w_{2k}, \\ z_{k+1} &= z_k + V_k\sin\gamma_k\Delta t + g_{3k}w_{3k}, \\ \psi_{k+1} &= \psi_k + \frac{g}{V_k}\tan\phi_k\Delta t + g_{4k}w_{4k}, \end{aligned} \tag{54}$$

where Δt is the time step size, w_{ik}, $i = 1, 2, 3, 4$, are standard Gaussian random variables, g_{1k}, g_{2k}, g_{3k}, and g_{4k} are multipliers correspond to the magnitude of the noise. Their values are all set to be 0.02. The system is linearized along a nominal trajectory to obtain a discrete, linear time-varying model. The system matrices are computed using equation (5). Similar to the 2-D double integrator example, several landmarks are placed in the environment. We assume that the vehicle can observe all landmarks and obtain estimates of the state at all time steps. The vehicle achieves better state estimates when it is closer to the landmarks. Let the location of the j^{th} landmark be given by L_j and the Euclidean distance between the 3-D position of the vehicle and the j^{th} landmarks be given by d_j. Then, the j^{th} position measurement corresponding to landmark j is

$$^j y = [x\ y\ z\ \phi]^\top + \eta d_j v, \quad j = 1, 2, \ldots, \ell, \tag{55}$$

where η is a parameter related to the intensity of the noise of the measurement and is set to 0.05, and v is a 4-dimensional standard Gaussian random vector. Thus, the total measurement vector y is a 4ℓ-dimensional vector, where ℓ is the number of landmarks.

The 3-D map of the environment is shown in Figure 11. The four spheres in the middle of the environment are the obstacles. The three black stars represent the landmarks. The CS-BRM nodes are also shown in Figure 11. The black dots represent the state mean of the nodes. The red dot is the state mean of the starting node and the blue square is the state mean of the goal node. The state means are deterministically chosen to discretize the environment. Covariances are sampled for each node. The small ellipsoid in the lower-right corner shows the 3σ confidence interval of the covariance of the 3-D position of that node. The covariances for other nodes are omitted.

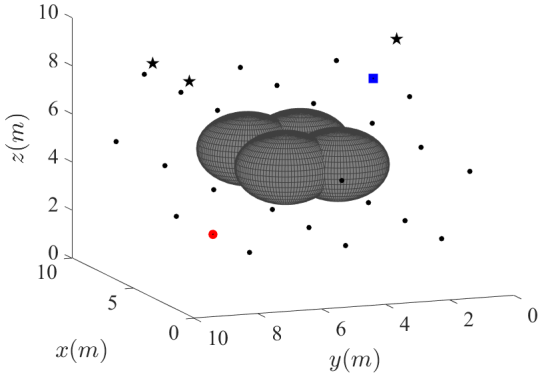


Fig. 11: 3-D map of the planning space. The black stars represent the landmarks' locations. The four grey balls in the middle of the cubic environment are the obstacles. The red dot, blue square, and black circles represent the state mean of the nodes.

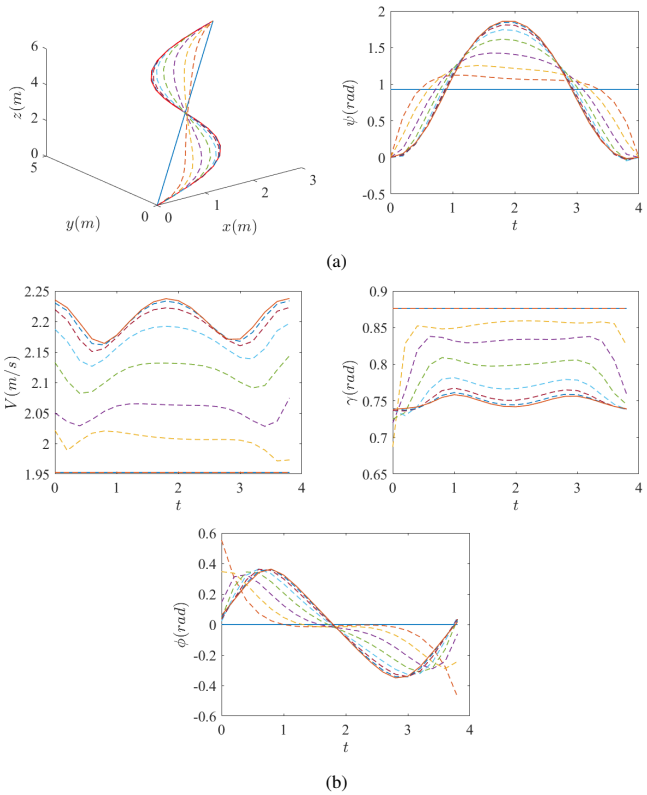


Fig. 12: Generation of a compatible nominal trajectory for the fixed-wing vehicle. The nominal trajectory and the reference trajectory are initialized as straight lines. The state trajectories are shown in (a) and the control trajectories are shown in (b). The blue solid lines correspond to the initial nominal trajectory (which is also the reference trajectory). The red lines correspond to the final mean trajectory, which is also the compatible nominal trajectory and the reference trajectory.

Algorithm 1 was used to find a compatible nominal trajectory for the fixed-wing vehicle, and the results are given in Figure 12. The nominal trajectory and the reference trajectory are initialized as straight lines in all four state dimensions and the three control input dimensions. The initial state is zero, and the terminal state is $[3\ 4\ 6\ 0]^\top$. The number of time-steps is $N = 20$, and $\Delta t = 0.2$ sec, $Q_k = I_4$, $R_k = 4I_3$.

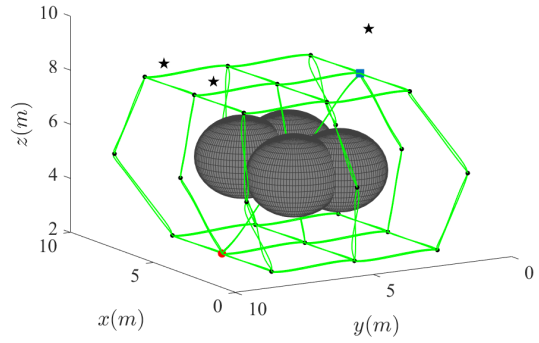


Fig. 13: CS-BRM of the fixed-wing vehicle. The green lines are the mean trajectories between the nodes.

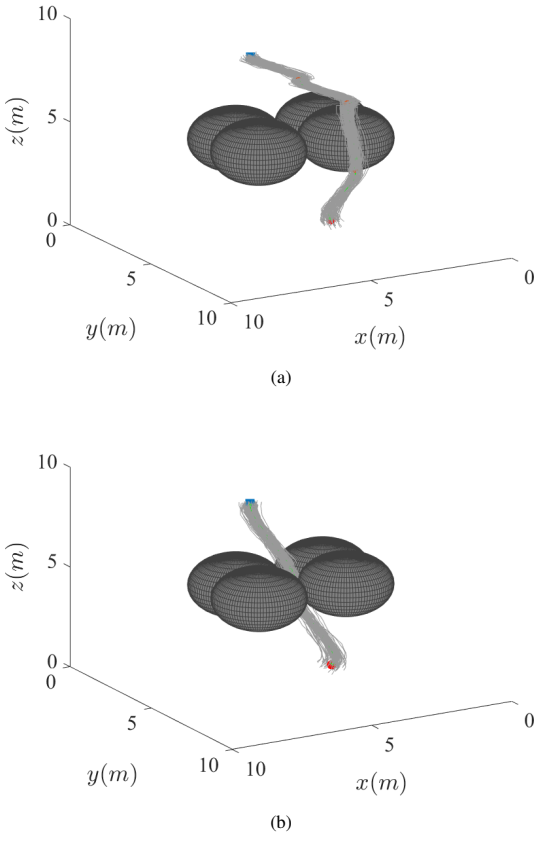


Fig. 14: Planning results using the CS-BRM for the fixed-wing vehicle. The gray lines are the trajectories from the Monte-Carlo simulations, 100 simulations are performed for each edge. (a) Based on the defined edge cost, which takes into account both the control effort and probability of collision, the algorithm finds the optimal path that trades off between control effort and collision cost. (b) Without considering the probability of collision, the algorithm finds a shorter length path, but which has a higher probability of collision compared to (a).

During each iteration of Algorithm 1, the nominal trajectory and the reference trajectory are updated using the computed mean trajectory of Section IV. The algorithm converged in eight iterations.

The final CS-BRM is given in Figure 13. The planned path using this CS-BRM is shown in Figure 14. The collision cost is considered in the edge cost of Figure 14(a), while the result without considering the probability of collision is shown in Figure 14(b). Similarly to the example in Section VI-A, the path in Figure 14(b) is shorter, but it requires the vehicle to fly through the narrow passage between the four obstacles resulting in a high probability of collision. Considering the motion uncertainty and observation uncertainty, a longer path is chosen in Figure 14(a), which trades off between control cost and collision cost while minimizing the total edge costs.

C. Quadrotor

The final example is a small quadrotor landing in a cluttered environment. Due to the differential flatness of the quadrotor model, one can recover the state trajectories of the quadrotor from its flat outputs [51]. Many works design smooth trajectories of the flat outputs for quadrotor trajectory planning. The differential flatness of the quadrotor equations of motion allows the modeling of the quadrotor dynamics as a 3-D double integrator system. In this work, we assume that this inner loop controller is already in place and use the 3-D double integrator model for planning. The system matrix G_k is estimated from flight data of the quadrotor and is given by $G_k = 10^{-3}\text{diag}(1, 1, 1, 6, 6, 6)$. A motion capture system is used to provide the ground truth of the quadrotor state. To imitate the measurement of a GPS sensor, artificial noises are added to the motion capture system measurements. The matrices of the measurement model are $C_k = I_6$ and $D_k = 0.03I_6$.

There are three landing points in the environment. After assigning zero velocity to the landing points, they are added to the graph as additionally sampled vertices. Following the CS-BRM algorithm, we can build a belief roadmap for quadrotor landing. By performing online graph searches, we can find safe motion plans for the quadrotor to land on either one of the landing points or fly from one landing point to another landing point.

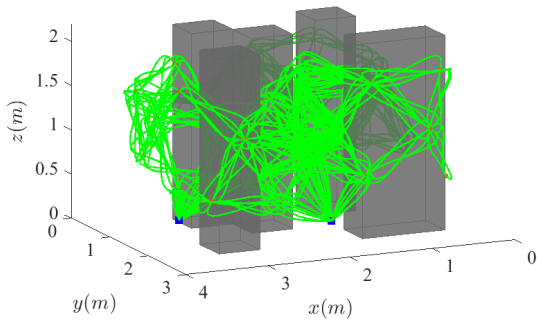


Fig. 15: CS-BRM roadmap of the 3-D quadrotor.

The environment along with the constructed CS-BRM roadmap is shown in Figure 15. The gray polyhedrons denote the obstacles. The green trajectories are the mean trajectories of the roadmap. The landing points are shown in blue. The proposed velocity sampling heuristic is used to sample multiple velocities at each position. After constructing the CS-BRM roadmap, it is used for multi-query motion planning.

Two landing motion plans are given in Figure 16. The gray trajectories are Monte Carlo simulation results. A comparison between CS-BRM and PRM is given in Figure 17. By taking plan safety into consideration and planning in belief space, CS-BRM generates a safer plan than PRM.

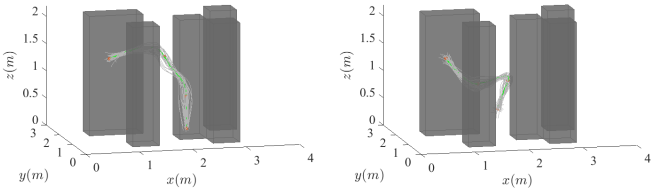


Fig. 16: Planning results of two quadrotor landing problems. The gray trajectories are the realization of the motion plans using Monte Carlo simulations.

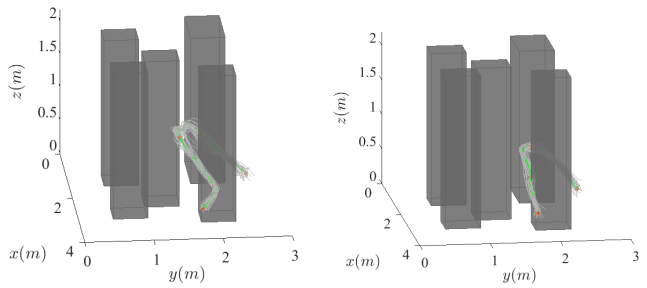


Fig. 17: Comparison between CS-BRM (left) and PRM (right). CS-BRM generation safe motion plans by planning in belief space and considering plan safety, while the plan from PRM can result in a high probability of collision.

We implemented this planner on a CrazyFlie quadrotor [52]. Since the planner uses a double integrator model, the plan computes the desired acceleration of the quadrotor. The desired acceleration is then sent to a low-level acceleration/attitude tracking controller, which is available with the Crazyflie, to close the control loop. A motion capture system is used to measure the position and velocity of the quadrotor. These measurements are the states x_k in the observation model (4), which are then corrupted by adding additive Gaussian noise v_k. Finally, the measurement y_k is used by the covariance steering controller. A snapshot of the quadrotor executing a landing plan is given in Figure 18. The complete experimental results are shown in Figure 19[1]. The gray trajectories are the results of 20 repeated experiments. A small trajectory tracking error along the z-axis is observed which is due to model mismatch and the fact that the desired acceleration is not achieved instantly by the inner tracking CrazyFlie controller. To reduce the error, a more sophisticated trajectory tracking controller [53] may be used to track the plan provided by the CS-BRM planner.

VII. Conclusion

A new belief space roadmap (BRM) algorithm is developed in this paper. The nodes in BRM represent distributions of the state of the system and are sampled in the belief space. The main idea is to use covariance steering theory to design the edge controllers of the BRM graph to steer the system from one distribution to another. Compared to the previous state-of-the-art [11], the proposed method directly deals with nonstationary belief nodes and can sample multiple velocities at the same position node, which has the advantage of more complete exploration of the belief space. For covariance steering of nonlinear systems, we introduce the concept of compatible nominal trajectories, which aim to better approximate the nonlinear dynamics through successive linearization. We also propose an efficient algorithm to compute compatible nominal trajectories. Compared to the standard PRM, the additional computation load comes from the computation of the edge controllers and edge cost evaluations, which, however, are done offline. The CS-BRM can be constructed incrementally, which allows balancing between the offline computation time and path quality. By explicitly incorporating motion and observation uncertainties, we show that the proposed CS-BRM algorithm generates efficient motion plans that take into account both the control effort and collision probability.

Fig. 18: A snapshot of the Crazyflie quadrotor executing a landing plan from CS-BRM.

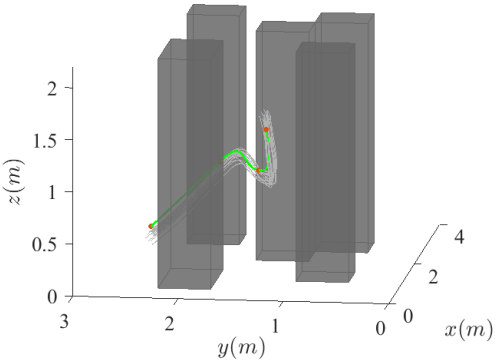


Fig. 19: Experiment results. The gray trajectories are the results of 20 repeated experiments.

Acknowledgment: This work has been supported by NSF awards IIS-1617630 and IIS-2008695, NASA NSTRF Fellowship 80NSSC17K0093, and ARL under CRA DCIST W911NF-17-2-0181. The authors would also like to thank Dipankar Maity for many insightful discussions regarding the

[1]A video of the experiment can be found at `https://youtu.be/bW45UtwTqiM`

output-feedback covariance steering problem.

REFERENCES

[1] B. Bonet and H. Geffner, "Planning with incomplete information as heuristic search in belief space," in *International Conference on Artificial Intelligence Planning Systems*, Breckenridge, CO, April 2000, pp. 52–61.

[2] 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.

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

[4] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, pp. 99–134, 1998.

[5] J. M. Porta, N. Vlassis, M. T. Spaan, and P. Poupart, "Point-based value iteration for continuous POMDPs," *Journal of Machine Learning Research*, pp. 2329–2367, November 2006.

[6] S. C. Ong, S. W. Png, D. Hsu, and W. S. Lee, "Planning under uncertainty for robotic tasks with mixed observability," *The International Journal of Robotics Research*, vol. 29, no. 8, pp. 1053–1068, 2010.

[7] K. Sun, B. Schlotfeldt, G. J. Pappas, and V. Kumar, "Stochastic motion planning under partial observability for mobile robots with continuous range measurements," *IEEE Transactions on Robotics*, vol. 37, no. 3, pp. 979–995, 2020.

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

[9] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[10] S. Prentice and N. Roy, "The belief roadmap: Efficient planning in belief space by factoring the covariance," *The International Journal of Robotics Research*, vol. 28, pp. 1448–1465, 2009.

[11] A. A. Agha-Mohammadi, S. Chakravorty, and N. M. Amato, "FIRM: sampling-based feedback motion-planning under motion uncertainty and imperfect measurements," *The International Journal of Robotics Research*, vol. 33, no. 2, pp. 268–304, 2014.

[12] 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.

[13] J. Pineau, G. Gordon, and S. Thrun, "Point-based value iteration: An anytime algorithm for POMDPs," in *International Joint Conference on Artificial Intelligence*, vol. 3, Acapulco, Mexico, August 2003, pp. 1025–1032.

[14] Y. Chen, T. T. Georgiou, and M. Pavon, "Optimal steering of a linear stochastic system to a final probability distribution, Part I," *IEEE Transactions on Automatic Control*, vol. 61, p. 1158–1169, 2015.

[15] ——, "Optimal steering of a linear stochastic system to a final probability distribution, part ii," *IEEE Transactions on Automatic Control*, vol. 61, no. 5, pp. 1170–1180, 2015.

[16] E. Bakolas, "Optimal covariance control for discrete-time stochastic linear systems subject to constraints," in *IEEE Conference on Decision and Control*, Las Vegas, NV, December 2016, pp. 1153–1158.

[17] M. Goldshtein and P. Tsiotras, "Finite-horizon covariance control of linear time-varying systems," in *IEEE Conference on Decision and Control*, Melbourne, Australia, December 2017, p. 3606–3611.

[18] K. Okamoto, M. Goldshtein, and P. Tsiotras, "Optimal covariance control for stochastic systems under chance constraints," *IEEE Control Systems Letters*, vol. 2, no. 2, pp. 266–271, 2018.

[19] J. Ridderhof and P. Tsiotras, "Uncertainty quantification and control during Mars powered descent and landing using covariance steering," in *AIAA Guidance, Navigation, and Control Conference*, Kissimmee, FL, Jan 2018, pp. 0611–0622.

[20] ——, "Chance-constrained covariance steering in a Gaussian random field via successive convex programming," *Journal of Guidance, Control, and Dynamics*, vol. 45, no. 4, pp. 599–610, 2022.

[21] J. Ridderhof, K. Okamoto, and P. Tsiotras, "Nonlinear uncertainty control with iterative covariance steering," in *IEEE Conference on Decision and Control*, Nice, France, December 2019, pp. 3484–3490.

[22] ——, "Chance constrained covariance control for linear stochastic systems with output feedback," in *IEEE Conference on Decision and Control*, Jeju Island, South Korea, December 2020, pp. 1153–1158.

[23] Y. Chen, T. Georgiou, and M. Pavon, "Steering state statistics with output feedback," in *IEEE Conference on Decision and Control*, Osaka, Japan, December 2015, pp. 6502–6507.

[24] E. Bakolas, "Covariance control for discrete-time stochastic linear systems with incomplete state information," in *American Control Conference*, Seattle, WA, May 2017, pp. 432–437.

[25] S. Ross, J. Pineau, S. Paquet, and B. Chaib-Draa, "Online planning algorithms for POMDP," *Journal of Artificial Intelligence Research*, vol. 32, pp. 663–704, 2008.

[26] G. Shani, J. Pineau, and R. Kaplow, "A survey of point-based POMDP solvers," *Autonomous Agents and Multi-Agent Systems*, vol. 27, no. 1, pp. 1–51, 2013.

[27] A. Somani, N. Ye, D. Hsu, and W. S. Lee, "DESPOT: online POMDP planning with regularization," *Advances in Neural Information Processing Systems*, vol. 26, pp. 1772–1780, 2013.

[28] R. Alterovitz, T. Simèon, and K. Y. Goldberg, "The stochastic motion roadmap: A sampling framework for planning with Markov motion uncertainty." in *Robotics: Science and systems*, 2007, pp. 233–241.

[29] S. D. Bopardikar, B. Englot, A. Speranzon, and J. van den Berg, "Robust belief space planning under intermittent sensing via a maximum eigenvalue-based bound," *The International Journal of Robotics Research*, vol. 35, no. 13, pp. 1609–1626, 2016.

[30] A. Bry and N. Roy, "Rapidly-exploring random belief trees for motion planning under uncertainty," in *IEEE International Conference on Robotics and Automation*, Shanghai, China, May 2011, pp. 723–730.

[31] A. A. Agha-Mohammadi, S. Chakravorty, and N. M. Amato, "Periodic feedback motion planning in belief space for non-holonomic and/or nonstoppable robots," *Technical Report: TR12-003, Parasol Lab., CSE Dept., Texas A&M University*, pp. 1–23, 2012.

[32] ——, "Online replanning in belief space for dynamical systems: Towards handling discrete changes of goal location," in *IEEE Conference on Robotics and Automation: Workshop on Combining Task and Motion Planning*, Karlsrühe, Germany, May 2013.

[33] A. A. Agha-mohammadi, S. Agarwal, S. K. Kim, S. Chakravorty, and N. M. Amato, "SLAP: simultaneous localization and planning under uncertainty via dynamic replanning in belief space," *IEEE Transactions on Robotics*, vol. 34, no. 5, pp. 1195–1214, 2018.

[34] J. T. Betts, "Survey of numerical methods for trajectory optimization," *Journal of Guidance, Control, and Dynamics*, vol. 21, no. 2, pp. 193–207, 1998.

[35] D. J. Webb and J. Van Den Berg, "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics," in *IEEE International Conference on Robotics and Automation*, Karlsrühe, Germany, May 2013, pp. 5054–5061.

[36] J. Van Den Berg and M. Overmars, "Kinodynamic motion planning on roadmaps in dynamic environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Diego, CA, 2007, pp. 4253–4258.

[37] M. A. Patterson and A. V. Rao, "GPOPS-II: A MATLAB software for solving multiple-phase optimal control problems using hp-adaptive Gaussian quadrature collocation methods and sparse nonlinear programming," *ACM Transactions on Mathematical Software*, vol. 41, no. 1, pp. 1–41, 2014.

[38] D. Zheng, J. Ridderhof, P. Tsiotras, and A. A. Agha-mohammadi, "Belief space planning: A covariance steering approach," in *IEEE International Conference on Robotics and Automation*, Philadelphia, PA, May 2022, pp. 11051–11057.

[39] E. Plaku, "Region-guided and sampling-based tree search for motion planning with dynamics," *IEEE Transactions on Robotics*, vol. 31, no. 3, pp. 723–735, 2015.

[40] A. Francis, A. Faust, H.-T. Chiang, J. Hsu, J. C. Kew, M. Fiser, and T.-W. E. Lee, "Long-range indoor navigation with PRM-RL," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1115–1134, 2020.

[41] D. Maity and P. Tsiotras, "Optimal controller and quantizer selection for partially observable Linear-Quadratic-Gaussian Systems," *arXiv preprint, arXiv:1909.13609*, 2019.

[42] J. Skaf and S. P. Boyd, "Design of affine controllers via convex optimization," *IEEE Transactions on Automatic Control*, vol. 55, no. 11, pp. 2476–2487, 2010.

[43] K. Okamoto and P. Tsiotras, "Optimal stochastic vehicle path planning using covariance steering," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2276–2281, 2019.

[44] A. Mosek, *The MOSEK Optimization Toolbox for MATLAB Manual*, 2020. [Online]. Available: https://docs.mosek.com/9.2/toolbox.pdf

[45] S. Diamond and S. Boyd, "CVXPY: a Python-embedded modeling language for convex optimization," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2909–2913, 2016.

[46] I. Olkin and F. Pukelsheim, "The distance between two random vectors with given dispersion matrices," *Linear Algebra and its Applications*, vol. 48, pp. 257–263, 1982.

[47] Q. H. Ho, Z. N. Sunberg, and M. Lahijanian, "Gaussian belief trees for chance constrained asymptotically optimal motion planning," in *International Conference on Robotics and Automation*, Philadelphia, PA, 2022, pp. 11029–11035.

[48] S. Patil, J. Van Den Berg, and R. Alterovitz, "Estimating probability of collision for safe motion planning under Gaussian motion and sensing uncertainty," in *IEEE International Conference on Robotics and Automation*, Saint Paul, MN, May 2012, pp. 3238–3244.

[49] N. E. Du Toit and J. W. Burdick, "Robotic motion planning in dynamic, cluttered, uncertain environments," in *IEEE International Conference on Robotics and Automation*, Anchorage, AK, May 2010, pp. 966–973.

[50] M. Owen, R. Beard, and T. McLain, "Implementing Dubins airplane paths on fixed-wing UAVs," in *Handbook of Unmanned Aerial Vehicles*, K. Valavanis and G. Vachtsevanos, Eds. Springer, 2015, ch. 64, pp. 1677–1701.

[51] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *IEEE International Conference on Robotics and Automation*, Shanghai, China, May 2011, pp. 2520–2525.

[52] J. Förster, "System identification of the crazyflie 2.0 nano quadrocopter," Bachelor's thesis, ETH Zurich, Zurich, Switzerland, 2015.

[53] S. Sun, A. Romero, P. Foehn, E. Kaufmann, and D. Scaramuzza, "A comparative study of nonlinear MPC and differential-flatness-based control for quadrotor agile flight," *IEEE Transactions on Robotics*, vol. 38, no.6, pp. 3357-3373, 2022.

APPENDIX A
DERIVATION OF SYSTEM MATRICES

At each time step k, we obtain the expression of x_k by forward propagating equation (17). By combining the steps of the iterative equation (17), $\hat{x}_k$ can be written as

$$\hat{x}_k = \bar{A}_k \hat{x}_{0^-} + \bar{B}_k U_{k-1} + H_k + \bar{L}_k \Xi_k, \tag{A.1}$$

for $k = 1, \cdots, N$, where

$$\begin{aligned} \bar{A}_k &= A_{k-1,0}, \\ \bar{B}_k &= [B_{k-1,0} \ B_{k-1,1} \ \cdots \ B_{k-1,k-1}], \\ H_k &= h_{k-1,0} + h_{k-1,1} + \cdots + h_{k-1,k-1}, \\ \bar{L}_k &= [L_{k,0} \ L_{k,1} \ \cdots \ L_{k,k}], \end{aligned} \tag{A.2}$$

and

$$\begin{aligned} A_{k_1,k_0} &= A_{k_1} A_{k_1-1} \cdots A_{k_0}, \quad A_{k,k} = A_k, \\ B_{k_1,k_0} &= A_{k_1,k_0+1} B_{k_0}, \quad B_{k,k} = B_k, \\ h_{k_1,k_0} &= A_{k_1,k_0+1} h_{k_0}, \quad h_{k,k} = h_k, \\ L_{k_1,k_0} &= A_{k_1-1,k_0} L_{k_0}, \quad L_{k,k} = L_k, \end{aligned} \tag{A.3}$$

Using (20) and stacking the equations of $\hat{x}_k$ from different time-steps, we obtain

$$\hat{X} = A\hat{x}_{0^-} + BU + H + L\Xi, \tag{A.4}$$

where

$$A = \begin{bmatrix} I \\ \bar{A}_1 \\ \bar{A}_2 \\ \vdots \\ \bar{A}_N \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ B_{0,0} & 0 & \cdots & 0 \\ B_{1,0} & B_{1,1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ B_{N-1,0} & B_{N-1,1} & \cdots & B_{N-1,N-1} \end{bmatrix},$$
$$H = \begin{bmatrix} 0 \\ H_1 \\ H_2 \\ \vdots \\ H_N \end{bmatrix}, \quad L = \begin{bmatrix} L_{0,0} & 0 & \cdots & 0 \\ L_{1,0} & L_{1,1} & \cdots & 0 \\ L_{2,0} & L_{2,1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ L_{N,0} & L_{N,1} & \cdots & L_{N,N} \end{bmatrix}. \tag{A.5}$$

APPENDIX B
PROOF OF PROPOSITION 1

Proof. The terminal constraint in (27) can be written as $\bar{x}_N = \bar{A}_N \bar{x}_0 + \bar{B}_N \bar{U} + H_N$. Define the Lagrangian as

$$\begin{aligned} \mathcal{L}(\bar{U}, \lambda) =& \bar{X}^\top Q \bar{X} + \bar{U}^\top R \bar{U} - 2\bar{X}^\top Q M_r \\ &+ \lambda^\top (\bar{x}_N - \bar{A}_N \bar{x}_0 - \bar{B}_N \bar{U} - H_N) \\ =& (A\bar{x}_0 + B\bar{U} + H)^\top Q (A\bar{x}_0 + B\bar{U} + H) \\ &+ \bar{U}^\top R \bar{U} - 2(A\bar{x}_0 + B\bar{U} + H)^\top Q M_r \\ &+ \lambda^\top (\bar{x}_N - \bar{A}_N \bar{x}_0 - \bar{B}_N \bar{U} - H_N), \end{aligned} \tag{B.1}$$

where λ is the Lagrangian multiplier. Take the partial derivative of $\mathcal{L}(\bar{U}, \lambda)$ with respect to $\bar{U}$. The first-order optimality condition yields

$$\begin{aligned} \nabla_{\bar{U}} \mathcal{L} =& 2(B^\top Q B + R)\bar{U} \\ &+ 2B^\top Q (A\bar{x}_0 + H - M_r) - \bar{B}_N^\top \lambda = 0, \end{aligned} \tag{B.2}$$

and thus,

$$\bar{U}^* = W(-V + \frac{1}{2}\bar{B}_N^\top \lambda), \tag{B.3}$$

where $W = (B^\top \bar{Q} B + R)^{-1}$ and $V = B^\top Q(A\bar{x}_0 + H - M_r)$. Substituting $\bar{U}^*$ into the terminal condition $E_N \bar{X} = \bar{x}_N$, where $\bar{X}$ is given by (24), we have

$$\bar{x}_N = \bar{A}_N \bar{x}_0 + \bar{B}_N W(-V + \frac{1}{2}\bar{B}_N^\top \lambda) + H_N. \tag{B.4}$$

It follows that

$$\frac{1}{2}\bar{B}_N W \bar{B}_N^\top \lambda = \bar{x}_N - \bar{A}_N \bar{x}_0 - H_N + \bar{B}_N W V. \tag{B.5}$$

Given that the system is controllable, the matrix $\bar{B}_N$ is full row rank and the matrix $\bar{B}_N W \bar{B}_N^\top$ is invertible. Thus,

$$\lambda = 2(\bar{B}_N W \bar{B}_N^\top)^{-1}(\bar{x}_N - \bar{A}_N \bar{x}_0 - H_N + \bar{B}_N W V). \tag{B.6}$$

Finally, substitute (B.6) into (B.3) to obtain the optimal control

$$\begin{aligned} \bar{U}^* =& W(-V + \bar{B}_N^\top (\bar{B}_N W \bar{B}_N^\top)^{-1} \times \\ & (\bar{x}_N - \bar{A}_N \bar{x}_0 - H_N + \bar{B}_N W V)). \end{aligned} \tag{B.7}$$

□

APPENDIX C
PROOF OF PROPOSITION 2

Proof. From (26), the cost functional in (12) may be rewritten as

$$J(u) = \mathbb{E}[\check{X}^\top Q\check{X} + \tilde{U}^\top R\tilde{U}] + \bar{U}^\top R\bar{U} + \bar{X}^\top Q\bar{X} - 2\bar{X}^\top QM_r + M_r^\top QM_r + \sum_{k=0}^{N-1} \text{trace}\tilde{P}_k Q_k. \tag{C.1}$$

Note that the quadratic term $\bar{X}^\top Q\bar{X} - 2\bar{X}^\top QM_r + M_r^\top QM_r \geq 0$ and is zero only when $M_r = \bar{X}$. The last term $\sum_{k=0}^{N-1} \text{trace}(\tilde{P}_k Q_k)$ is a constant that does not depend on the control $\bar{U}$ and M_r.

In (C.1), M_r is assumed given and fixed. Now, let M_r also be a design variable and define $J(u, M_r) = J(u) - \sum_{k=0}^{N-1} \text{trace}(\tilde{P}_k Q_k)$, then

$$\begin{aligned}\min_{u,M_r} J(u, M_r) &\geq \min_{u,M_r} (\mathbb{E}[\check{X}^\top Q\check{X} + \tilde{U}^\top R\tilde{U}] + \bar{U}^\top R\bar{U}) \\ &\quad + \min_{u,M_r} (\bar{X}^\top Q\bar{X} - 2\bar{X}^\top QM_r + M_r^\top QM_r) \\ &= \min_{u} (\mathbb{E}[\check{X}^\top Q\check{X} + \tilde{U}^\top R\tilde{U}] + \bar{U}^\top R\bar{U}) \\ &\quad + \min_{u,M_r} (\bar{X}^\top Q\bar{X} - 2\bar{X}^\top QM_r + M_r^\top QM_r) \\ &\geq \min_{u} (\mathbb{E}[\check{X}^\top Q\check{X} + \tilde{U}^\top R\tilde{U}] + \bar{U}^\top R\bar{U}).\end{aligned} \tag{C.2}$$

The two inequalities in (C.2) become equalities when $M_r = \bar{X}$. Thus, $J(u, M_r)$ achieves minimum only if $M_r = \bar{X}$. □

APPENDIX D
PROOF OF PROPOSITION 3

Proof. Since the Kalman filter is an iterative update, we only need to show that, if $\tilde{P}'_{0\text{-}} \preceq \tilde{P}_{0\text{-}}$, then $\tilde{P}'_{1\text{-}} \preceq \tilde{P}_{1\text{-}}$. The rest can be shown by induction. Define $V_0 = D_0 D_0^\top$ and $W_0 = G_0 G_0^\top$. Using equation (10), we have

$$L_0 = \tilde{P}_{0\text{-}} C_0^\top (C_0 \tilde{P}_{0\text{-}} C_0^\top + V_0)^{-1}, \tag{D.1}$$

$$\tilde{P}_0 = (I - L_0 C_0)\tilde{P}_{0\text{-}} = \tag{D.2}$$

$$\tilde{P}_{0\text{-}} - \tilde{P}_{0\text{-}} C_0^\top (C_0 \tilde{P}_{0\text{-}} C_0^\top + V_0)^{-1} C_0 \tilde{P}_{0\text{-}}, \tag{D.3}$$

$$\tilde{P}_{1\text{-}} = A_0[\tilde{P}_{0\text{-}} - \tilde{P}_{0\text{-}} C_0^\top (C_0 \tilde{P}_{0\text{-}} C_0^\top + V_0)^{-1} C_0 \tilde{P}_{0\text{-}}] A_0^\top + W_0. \tag{D.4}$$

Next, define

$$\Phi(H, \tilde{P}_{0\text{-}}) = (A_0 + HC_0)\tilde{P}_{0\text{-}}(A_0 + HC_0)^\top + W_0 + HV_0H^\top. \tag{D.5}$$

Note that $\Phi(H, \tilde{P}_{0\text{-}})$ is non-decreasing in $\tilde{P}_{0\text{-}}$. That is, if $\tilde{P}'_{0\text{-}} \preceq \tilde{P}_{0\text{-}}$, then $\Phi(H, \tilde{P}'_{0\text{-}}) \preceq \Phi(H, \tilde{P}_{0\text{-}})$. Since $\tilde{P}_{0\text{-}}$ and $V_0 \succeq 0$, it follows that $\Phi(H, \tilde{P}_{0\text{-}})$ is quadratic and convex in the variable H. By taking the partial derivative of $\Phi(H, \tilde{P}_{0\text{-}})$ respect to H and setting it equal to zero yields

$$\nabla_H \Phi(H, \tilde{P}_{0\text{-}}) = 2(A_0 + HC_0)\tilde{P}_{0\text{-}} C_0^\top + 2HV_0 = 0. \tag{D.6}$$

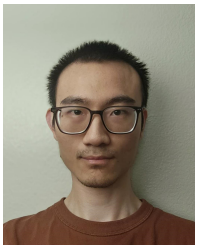
Thus, the minimum of $\Phi(H, \tilde{P}_{0\text{-}})$ with respect to H is achieved when

$$H^* = -A_0 \tilde{P}_{0\text{-}} C_0^\top [C_0 \tilde{P}_{0\text{-}} C_0^\top + V_0]^{-1}. \tag{D.7}$$

The minimum is

$$\Phi(H^*, \tilde{P}_{0\text{-}}) = A_0[\tilde{P}_{0\text{-}} - \tilde{P}_{0\text{-}} C_0^\top (C_0 \tilde{P}_{0\text{-}} C_0^\top + V_0)^{-1} C_0 \tilde{P}_{0\text{-}}] A_0^\top + W_0. \tag{D.8}$$

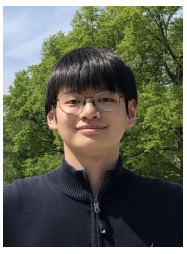
Thus, $\tilde{P}_{1\text{-}} = \min_H \Phi(H, \tilde{P}_{0\text{-}}) = \Phi(H^*, \tilde{P}_{0\text{-}})$. Now given a new estimation error covariance $\tilde{P}'_{0\text{-}}$ that satisfies $\tilde{P}'_{0\text{-}} \preceq \tilde{P}_{0\text{-}}$, we have $\tilde{P}'_{1\text{-}} = \min_H \Phi(H, \tilde{P}'_{0\text{-}}) = \Phi(H^*(\tilde{P}'_{0\text{-}}), \tilde{P}'_{0\text{-}}) \preceq \Phi(H^*(\tilde{P}'_{0\text{-}}), \tilde{P}_{0\text{-}}) \preceq \Phi(H^*(\tilde{P}_{0\text{-}}), \tilde{P}_{0\text{-}}) = \tilde{P}_{1\text{-}}$. □



Dongliang Zheng received the B.Eng. degree in automation from Northeastern University, Shenyang, China, in 2015, and the M.S. degree in control engineering at Shanghai Jiao Tong University, Shanghai, China, in 2018. He is currently working toward the Ph.D. degree in aerospace engineering at Georgia Institute of Technology, Atlanta, GA, USA.

His current research interests include motion planning and control for autonomous robots.

Jack Ridderhof received an MS in mathematics and a PhD in aerospace engineering both from the Georgia Institute of Technology, Atlanta, USA where he was a NASA Space Technology Fellow. His research interests are in stochastic control for spacecraft guidance.



Zhiyuan Zhang is a robotics Ph.D. student at the Georgia Institute of Technology, Atlanta, USA. He earned his B.S. degree in mechanical engineering in 2020 and an M.S. in aerospace engineering at the same institution.

Zhiyuan's research focuses on game theoretical control and stochastic optimal control, exploring algorithms applicable to physical devices with real-life applications. Passionate about merging theory with practical solutions, he envisions contributing to advancements in control systems for tangible impact.

Panagiotis Tsiotras (F'19) is the David and Andrew Lewis Chair Professor in the Daniel Guggenheim School of Aerospace Engineering at the Georgia Institute of Technology, Atlanta, GA, USA, He received his Ph.D. degree in aerospace engineering and an M.S. degree in mathematics from Purdue University, IN, USA, an M.S. degree in aerospace engineering from Virginia Tech, VA, USA and an Eng. Dipl. in mechanical engineering from NTUA, Athens, Greece. He has held visiting research appointments at MIT, JPL, INRIA Rocquencourt, and Mines ParisTech.

His research interests include optimal control of nonlinear systems and ground, aerial and space vehicle autonomy. He has served on the Editorial Boards of the Transactions on Automatic Control, the IEEE Control Systems Magazine, the AIAA Journal of Guidance, Control and Dynamics, the Dynamic Games and Applications, and Dynamics and Control. He is the recipient of the NSF CAREER award, the Outstanding Aerospace Engineer award from Purdue, and the Technical Excellence Award in Aerospace Control from IEEE. He is a Fellow of AIAA and AAS.

Ali-akbar Agha-mohammadi received the B.S. and M.S. degrees in electrical and computer engineering (control systems) from Tabriz University and K.N. Toosi University of Technology, in 2005 and 2008, respectively. He received a Ph.D. degree in computer science and engineering from Texas A&M University, TX, USA, in 2013.

He is the co-founder and CEO of FieldAI, a company specializing in advanced robotic autonomy for challenging and off-road environments. Prior to establishing FieldAI, Dr. Agha held roles as a Technologist and Group Leader at NASA-JPL and Caltech, and as a researcher at Qualcomm Research and MIT. His expertise centers on autonomous robotic vehicles, particularly in perception and planning.