Belief Space Planning: A Covariance Steering Approach

Dongliang Zheng, Jack Ridderhof, Panagiotis Tsiotras, and Ali-akbar Agha-mohammadi

***Abstract*— A new belief space planning algorithm, called covariance steering Belief RoadMap (CS-BRM), is introduced, which is a multi-query algorithm for motion planning of dynamical systems under simultaneous motion and observation uncertainties. CS-BRM extends the probabilistic roadmap (PRM) approach to belief spaces and is based on the recently developed theory of covariance steering (CS) that enables guaranteed satisfaction of terminal belief constraints in finite-time. The CS-BRM algorithm allows the sampling of non-stationary belief nodes, and thus is able to explore the velocity space and find efficient motion plans. We evaluate CS-BRM in different planning problems and demonstrate the benefits of the proposed approach.**

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 function (pdf) over all possible states, which is commonly referred to as the *belief* or information state [1]. 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 [2]. Despite some recent progress in terms of more efficient POMDP solvers [3], [4], solving POMDP in general domains is very challenging, especially for long-horizon, global planning problem in continuous domain. Planning in infinite-dimensional distributional (e.g., belief) spaces can become more tractable by the use of roadmaps, that is, graphs constructed by sampling. Since their introduction [5], such belief roadmaps (BRMs) have increased in popularity owing to their simplicity and their ability to avoid local minima.

Sampling-based motion planning algorithms such as probabilistic roadmaps (PRM) [6] and rapidly exploring random trees (RRTs) [7] can be used to solve planning problems in high-dimensional continuous state spaces. through sampling the search space. However, traditional PRM-based methods only address deterministic systems. PRM methods have been extended to belief space planning using belief space roadmaps (BRMs) [5], [8], [9]. One of the main challenges of belief space roadmap (BRM) methods is that the cost of the edge depend on the path taken to that edge, resulting in the "curse of history" problem for POMDPs [8], [10]. This problem arises from the unreachability of the belief nodes; even if the robot has full control of its mean, it is difficult to reach higher order moments (e.g., a specified covariance). Since the nodes in BRM are sampled in the belief space, the edges in BRM should ideally steer the robot from one distribution to another. If reachability of BRM nodes is not achieved, an edge in the BRM depends on all preceding edges along the path.

The BRM algorithm proposed in [5] finds the minimum estimation uncertainty path for a robot from any starting position to a goal. Since the reachability of belief node is not achieved, [5] requires a new search whenever the starting position is changed. By taking into account the controller and the sensors used, [9] computes the true *a priori* probability distribution of the state of the robot and uses this true *a priori* distribution along with the RRT algorithm for motion planning. Reference [11] also uses the *a priori* distribution and builds a belief tree. Searching over a roadmap is also studied in [12] for localization uncertainty. However, the independence between edges is not satisfied in [9], [11], [12] and these tree-based methods can not be extended to PRM.

The state-of-the-art in terms of BRM methods is probably [8], which tackles the "curse of history" problem. The proposed SLQG-FIRM method achieves node reachability using a stationary LQG controller. 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. Thus, this method cannot explore the velocity space and 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. The method in [13] improves the online phase by recomputing the local plans, while the offline roadmap construction phase is the same as in FIRM [8].

Recent developments in explicitly controlling the covariance of a linear system [14], [15] provide an appealing approach to construct BRMs 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], [17], [18]. In [16], the covariance steering problem is formulated as a convex program. Additional state chance constraints are

Dongliang Zheng, Jack Ridderhof, and Panagiotis Tsiotrasis are with School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA, USA

Ali-akbar Agha-mohammadi is with NASA-JPL, California Institute of Technology, Pasadena, CA, USA

considered in [18] and nonlinear systems are considered in [19] using iterative linearization. The covariance steering problem with output feedback has also been studied in [20], [21], [22].

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. In addition, covariance steering avoids the limitation of sampling in the equilibrium space, and thus the proposed CS-BRM method allows sampling of non-stationary belief nodes. Our method allows searching in the velocity space and thus finds paths with lower cost. For nonlinear systems that are well approximated by its linearization along a nominal trajectory, we also develop a simple, yet efficient, algorithm to find suitable nominal trajectories between the nodes of the BRM graph to steer the mean states in an optimal fashion.

The contributions of the paper are summarized as follows. First, a new belief space planning method, called CS-BRM, is developed, to construct a roadmap in belief space. CS-BRM achieves finite-time belief node reachability using covariance steering and overcomes the limitation of sampling stationary nodes. Second, the concept of *compatible nominal trajectory* is introduced, which aims to improve the performance of linearization-based control methods to control nonlinear systems. The CNT algorithm is proposed to compute nominal feasible trajectories for nonlinear systems. Finally, the algorithm is tested in different environments.

II. Problem Statement

We consider the problem of planning for a nonholonomic robot in an uncertain environment which contains obstacles. The uncertainty in the problem stems from model uncertainty, as well 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. This is a difficult problem to solve in its full generality. Here we use graph-based methods to build a roadmap in the space of distributions of the states (e.g., the belief space) for multi-query motion planning. In the next section, we describe a methodology to design BRM edge controllers that allow to steer from one node (i.e., distribution) to another. Similar to previous works [5], [8], [9], we consider Gaussian distributions where the belief is given by the state mean and the state covariance.

III. Covariance Steering

In this section, we give a introduction of covariance steering and outline some key results. Some of these results are explicitly used in subsequent sections. The derivation in this section follows closely [20], [18].

Given a nominal trajectory $(n_k)_{k=0}^{N-1}$, where $n_k = (x_k^r, u_k^r)$, we can construct a linear approximation of (1)-(2) around $(n_k)_{k=0}^{N-1}$ via linearization as follows

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

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

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.

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 $x_N \sim \mathcal{N}(\bar{x}_N, P_N)$ where $P_N \preceq P_f$, while minimizing the cost functional

$$J(u) = \mathbb{E}\left[\sum_{k=0}^{N-1} (x_k - m_k)^\top Q_k (x_k - m_k) + u_k^\top R_k u_k\right], \tag{5}$$

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_N are the covariance matrices of x_0 and x_N, respectively, the terminal covariance P_N is upper bounded by a given covariance matrix P_f, and the matrices $(Q_k \succeq 0)$ and $(R_k \succ 0)$ are given.

A. Separation of Observation and Control

Similarly to [20], 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 $\hat{x}_{0^-}$ and distributed according to $\hat{x}_{0^-} \sim \mathcal{N}(\bar{x}_0, \hat{P}_{0^-})$, and 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 estimated state at time k, denoted as $\hat{x}_k = \mathbb{E}[x_k | Y_k]$, with Y_k denoting the filtration generated by $\{\hat{x}_{0^-}, y_i : 0 \le i \le k\}$, is computed from [23]

$$\begin{aligned} \hat{x}_k &= \hat{x}_{k^-} + L_k(y_k - C_k \hat{x}_{k^-}), \\ \hat{x}_{k^-} &= A_{k-1}\hat{x}_{k-1} + B_{k-1}u_{k-1} + h_{k-1}, \end{aligned} \tag{6}$$

$$\begin{aligned} L_k &= \tilde{P}_{k^-} C_k^\top (C_k \tilde{P}_{k^-} C_k^\top + D_k D_k^\top)^{-1}, \\ \tilde{P}_k &= (I - L_k C_k)\tilde{P}_{k^-}, \\ \tilde{P}_{k^-} &= A_{k-1}\tilde{P}_{k-1}A_{k-1}^\top + G_{k-1}G_{k-1}^\top, \end{aligned} \tag{7}$$

where L_k is the Kalman gain and where the covariances of x_k, $\hat{x}_k$ and $\tilde{x}_k$ are denoted 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.

It can be shown that the cost functional (5) can be written as

$$\begin{aligned} J(u) =& \mathbb{E}\left[\sum_{k=0}^{N-1} \hat{x}_k^\top Q_k \hat{x}_k + u_k^\top R_k u_k\right] - 2\sum_{k=0}^{N-1} \bar{x}_k^\top Q_k m_k \\ &+ \sum_{k=0}^{N-1} (\operatorname{trace}(\tilde{P}_k Q_k) + m_k^\top Q_k m_k), \end{aligned} \tag{8}$$

where the last summation is deterministic and does not depend on the control and thus it can be discarded.

By defining the *innovation process* $(\xi_k)_{k=0}^N$ as

$$\xi_k = y_k - \mathbb{E}[y_k | Y_{k-1}], \tag{9}$$

and noting that $\mathbb{E}[y_k|Y_{k-1}] = \mathbb{E}[C_k x_k + D_k v_k | Y_{k-1}] = C_k \hat{x}_{k^-}$, the estimated state dynamics in (6) can be rewritten as

$$\hat{x}_{k+1} = A_k \hat{x}_k + B_k u_k + h_k + L_{k+1} \xi_{k+1}, \tag{10}$$

with $\hat{x}_0 = \hat{x}_{0^-} + L_0 \xi_0$. We can then restate Problem 1 as follows.

Problem 2: Find the control sequence $(u_k)_{k=0}^{N-1}$, such that the system (10) starting from the initial distribution $\hat{x}_{0^-} \sim \mathcal{N}(\bar{x}_0, P_0 - \tilde{P}_{0^-})$ reaches the final distribution $\hat{x}_N \sim \mathcal{N}(\bar{x}_N, P_N - \tilde{P}_N)$, where $P_N \preceq P_f$, while minimizing the cost functional

$$\hat{J}(u) = \mathbb{E}\left[\sum_{k=0}^{N-1} \hat{x}_k^\top Q_k \hat{x}_k + u_k^\top R_k u_k\right] - 2\sum_{k=0}^{N-1} \bar{x}_k^\top Q_k m_k. \tag{11}$$

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 $\hat{x}_k$.

B. Separation of Mean Control and Covariance Control

We first separate Problem 2 into a mean control problem and a covariance control problem, then give their solutions.

By defining the augmented vectors $U_k = [u_0^\top \; u_1^\top \; \cdots \; u_k^\top]^\top$, $\Xi_k = [\xi_0^\top \; \xi_1^\top \; \cdots \; \xi_k^\top]^\top$, and $\hat{X}_k = [\hat{x}_0^\top \; \hat{x}_1^\top \; \cdots \; \hat{x}_k^\top]^\top$. We can compute $\hat{X}_k$ as follows

$$\hat{X} = A\hat{x}_{0^-} + BU + H + L\Xi, \tag{12}$$

where $\hat{X} = \hat{X}_N$, $U = U_{N-1}$, $\Xi = \Xi_N$, and A, B, H, and L are block matrices constructed using the system matrices in (10). Defining $\check{x}_{0^-} \triangleq \hat{x}_{0^-} - \bar{x}_0$, $\bar{X} \triangleq \mathbb{E}[\hat{X}]$, $\bar{U} \triangleq \mathbb{E}[U]$, $\check{X} \triangleq \hat{X} - \bar{X}$, and $\tilde{U} \triangleq U - \bar{U}$, and using (12), it follows that

$$\bar{X} = A\bar{x}_0 + B\bar{U} + H, \tag{13}$$

$$\check{X} = A\check{x}_{0^-} + B\tilde{U} + L\Xi. \tag{14}$$

The cost functional in (11) can be rewritten as

$$\begin{aligned} \hat{J}(u) =& \mathbb{E}[\check{X}^\top Q \check{X} + \tilde{U}^\top R \tilde{U}] \\ &+ \bar{X}^\top Q \bar{X} + \bar{U}^\top R \bar{U} - 2\bar{X}^\top Q M_r, \end{aligned} \tag{15}$$

where $Q = \operatorname{blkdiag}(Q_0, Q_1, \ldots, Q_{N-1}, 0)$, $R = \operatorname{blkdiag}(R_0, R_1, \ldots, R_{N-1})$, $M_r = [m_0^\top \; m_1^\top \; \cdots \; m_N^\top]^\top$. Note that the cost function (15) can be separated into two parts along with separated dynamics (13) and (14), respectively. From (13), (14), and (15), the covariance steering problem of the estimated state $\hat{x}$ (Problem 2) can thus be divided into mean control and covariance control problems, which can be found in the extended version of this paper [24].

The solution for the mean control problem to obtain the mean trajectory $(\bar{x}_k)_{k=0}^N$ and $(\bar{u}_k)_{k=0}^{N-1}$ is given by [18]

$$\begin{aligned} \bar{U}^* =& W(-V + \bar{B}_N^\top (\bar{B}_N W \bar{B}_N^\top)^{-1} \\ & (\bar{x}_N - \bar{A}_N \bar{x}_0 - H_N + \bar{B}_N W V)), \end{aligned} \tag{16}$$

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

The solution to the covariance control problem is given by the controller of the form

$$\tilde{u}_k = \sum_{i=0}^{k} K_{k,i} \check{x}_i, \tag{17}$$

equivalently, by $\tilde{U} = K\check{X}$, where the control gain matrix K is lower block diagonal and is computed from $K = F(I + BF)^{-1}$, where F is obtained by solving the following convex optimization problem [20]

$$\min_F \; \operatorname{trace}[((I + BF)^\top Q (I + BF) + F^\top R F) P_Z], \tag{18}$$

subject to

$$\|P_Z^{1/2} (I + BF)^\top E_N^\top (P_f - \tilde{P}_N)^{-1/2}\| - 1 \leq 0, \tag{19}$$

where $P_Z = A\hat{P}_{0^-} A^\top + L P_\Xi L^\top$ is the covariance of $Z \triangleq A\check{x}_{0^-} + L\Xi$ and $P_\Xi = \operatorname{blkdiag}(P_{\xi_0}, \cdots, P_{\xi_N})$ is the covariance of the innovation process. For more details on the construction of the covariance steering controller, please see [20], [25].

IV. Computing the Nominal Trajectory

In this section, we develop an efficient algorithm to find nominal trajectories for nonlinear systems using the mean controller (16). A deterministic nonlinear dynamic model, i.e., with the noise w_k set to zero,

$$x_{k+1} = f(x_k, u_k, 0), \tag{20}$$

is considered. 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 (16) to compute a control sequence $(u_k^c)_{k=0}^{N-1}$ and the corresponding state sequence $(x_k^c)_{k=0}^{N-1}$.

Definition 1: A nominal trajectory is called a *compatible nominal trajectory* for system (20) 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.

Algorithm 1: Compatible Nominal Trajectory

Initialize $(x_k^r)_{k=0}^{N-1}$ and $(u_k^r)_{k=0}^{N-1}$;
while *NotConverged* **do**
  Linearize (20) along $(x_k^r)_{k=0}^{N-1}$ and $(u_k^r)_{k=0}^{N-1}$;
  Compute the mean optimal control using (16) 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 (13) ;
  Set $(x_k^r = x_k^c)_{k=0}^{N-1}$ and $(u_k^r = u_k^c)_{k=0}^{N-1}$;
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 (16) 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.

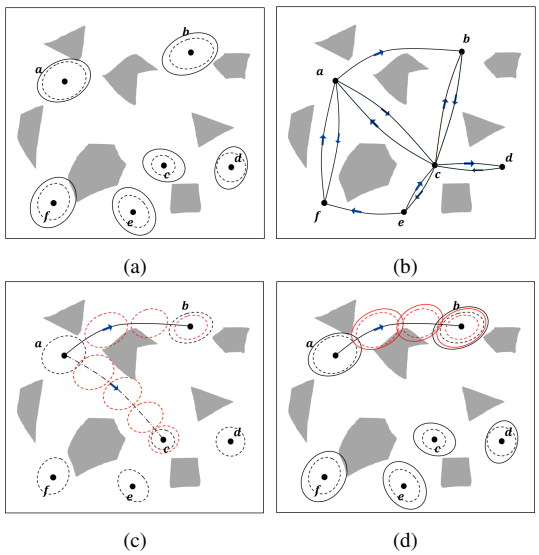


Fig. 1: Construction of the CS-BRM. Gray shapes denote obstacles. (a) Node sampling. (b) Mean trajectories are computed for all neighboring node pairs using the mean controller. Only mean trajectories that are collision-free are preserved. (c) Kalman filter updates are simulated for each possible edge; (d) 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 conditional 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], [8]. 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 $b_k = (\bar{x}_k, \hat{P}_k, \tilde{P}_k)$.

Algorithm 2: Constructing CS-BRM

1 $V = \{nod_1, \ldots, nod_n\} \leftarrow$ `SampleNodes`(n) ;
2 $V_r \leftarrow V$, $E \leftarrow \emptyset$;
3 **for** $i = 1 : n$ **do**
4 $V_{\text{near}} \leftarrow$ `Neighbor`$(V_r,\ nod_i)$;
5 **foreach** $nod_j \in V_{\text{near}}$ **do**
6 $(\bar{U}_{ij},\ \tau_{ij},\ MCost_{ij}) \leftarrow$ `MTraj`$(nod_i,\ nod_j)$;
7 **if** `ObstacleFree`(τ_{ij}) **then**
8 $\tilde{P}_{N^-} \leftarrow$ `KF`$(nod_i,\ nod_j)$;
9 **if** $\tilde{P}_{N^-} \preceq \tilde{P}_{nod_j}$ **then**
10 $(\tilde{U}_{ij},\ CovCost_{ij}) \leftarrow$ `CovControl`$(nod_i,\ nod_j)$;
11 $CollisionCost_{ij} \leftarrow$ `MonteCarlo`$(\bar{U}_{ij},\ \tilde{U}_{ij})$;
12 $E_{ij} \leftarrow (\bar{U}_{ij},\ \tilde{U}_{ij},\ EdgeCost_{ij})$;
13 $E \leftarrow E \cup E_{ij}$
14 $(\bar{U}_{ji},\ \tau_{ji},\ MCost_{ji}) \leftarrow$ `MTraj`$(nod_j,\ nod_i)$;
15 **if** `ObstacleFree`(τ_{ji}) **then**
16 Repeat line 8-13 with i and j swapped
17 $V_r \leftarrow V_r \setminus nod_i$;
18 CS-BRM $\leftarrow (V,\ E)$;
19 **return** CS-BRM;

An illustration of the steps for building the CS-BRM is shown in Fig. 1. The algorithm for constructing the CS-BRM is given in Algorithm 2. The following procedures are used in the algorithm.

Sample Nodes: The function `SampleNodes`(n) samples n CS-BRM nodes. A node in the CS-BRM is represented by the tuple $(\bar{x}, P, \tilde{P}_{\cdot})$. Since $P = \hat{P}_{\cdot} + \tilde{P}_{\cdot}$, the node can also be equivalently represented by $(\bar{x}, \hat{P}_{\cdot}, \tilde{P}_{\cdot})$. For constructing node j, the algorithm starts by sampling the free state space, which provides the mean $\bar{x}_j$ of the distribution. Then, $\hat{P}_{j^-}$ and $\tilde{P}_{j^-}$ are sampled from the space of symmetric and positive definite matrix space. $\tilde{P}_{j^-}$ is the initial condition of the Kalman filter update for the edges that are coming out of node j. In Fig. 1(a), for each node, $\bar{x}$ is shown as a black dot, P is shown as a solid ellipse, and $\tilde{P}_{\cdot}$ is shown as a dashed ellipse.

Neighbor: The function `Neighbor`(V_r, 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.

Mean Trajectory: Given two nodes nod_i and nod_j, `MTraj`(nod_i, nod_j) uses the mean controller (16) 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}, and the cost of the mean control $MCost_{ij}$.

Obstacle Checking: The function `ObstacleFree`(τ_{ij}) returns true if the mean trajectory τ_{ij} is collision free.

Kalman Filter: Given two nodes nod_i and nod_j, `KF`$(nod_i,\ nod_j)$ returns the prior estimation error covariance at the last time step of the trajectory from nod_i to nod_j.

Covariance Control: `CovControl`(nod_i, nod_j) solves the

covariance control problem from nod_i to nod_j. It returns the control $\tilde{U}_{ij}$ and the cost $CovCost_{ij}$.

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 $\hat{x}_{0|0}$ are sampled from their corresponding distributions. The state trajectory is simulated using the mean control $\bar{U}_{ij}$ and 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 $CollisionCost_{ij}$ is taken to be proportional to the probability of collision along the edge.

Algorithm 2 starts by sampling n nodes in the belief space using `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}. In addition to the edge controller, the edge cost is computed for each edge. The edge cost $EdgeCost_{ij}$ is a weighted sum of $MCost_{ij}$, $CovCost_{ij}$, and $CollisionCost_{ij}$. With covariance steering serving as the lower-level controller, the higher-level motion planning problem using the roadmap is a graph search problem similarly to a PRM. Thus, the covariance steering approach transforms the belief space roadmap into traditional PRM with specific edge costs.

In CS-BRM, each edge is obtained by solving a covariance steering problem and the planned path using CS-BRM consists of a concatenation of edges. Next, we given some results regarding the concatenation of edges. Let $(\tilde{P}_{k|k})_{k=1}^{N}$ be the sequence corresponding to the initial condition $\tilde{P}_{0|0}$ and let $(\tilde{P}'_{k|k})_{k=0}^{N}$ be the sequence corresponding $\tilde{P}'_{0|0}$. We have the following results.

Lemma 1: If $\tilde{P}'_{0|0} \preceq \tilde{P}_{0|0}$, then $\tilde{P}'_{k|k} \preceq \tilde{P}_{k|k}$, for all $k = 0, \cdots, N$. The proof of this lemma is omitted. A proof of a similar result can be found in [11]. From Lemma 1, it is straightforward to show that, if $\tilde{P}'_{0|0} \preceq \tilde{P}_{0|0}$, we also have $\tilde{P}'_k \preceq \tilde{P}_k$ for all $k = 0, \ldots, N$.

Proposition 2: 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 pre-computed 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 $\tilde{P} \preceq \tilde{P}_j$.

The proof of this proposition is straightforward and is omitted. Proposition 2 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.

VI. Numerical Examples

In this section, we illustrate our theoretical results for the motion planning problem of a 2-D double integrator, a 3-D double integrator, and a fixed-wing aerial vehicle in three environments.

A. 2-D Double Integrator

The environment considered is shown in Figure 2. 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 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

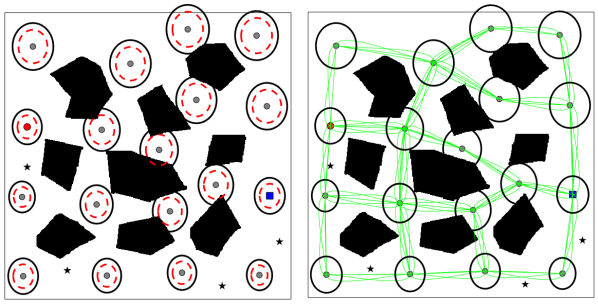
$$ {}^{j}y = [x^{(1)} \; x^{(2)}]^\top + \eta_p d_j v_p, \quad j = 1, 2, \cdots, \ell, \tag{21} $$

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

$$ y_v = [x^{(3)} \; x^{(4)}]^\top + \eta_v v_v, \tag{22} $$

where η_v is a parameter related to the intensity of the velocity measurement noise that is set to 0.2, and v_v is a two-dimensional standard Gaussian random vector. 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 2(a). To show the advantage of sampling non-stationary nodes, we sample multiple velocities at each position. Figure 2(b) shows the roadmap from the CS-BRM algorithm. Only the mean trajectories are shown (green lines). Note that each position have multiple belief nodes because of the multiple velocities.

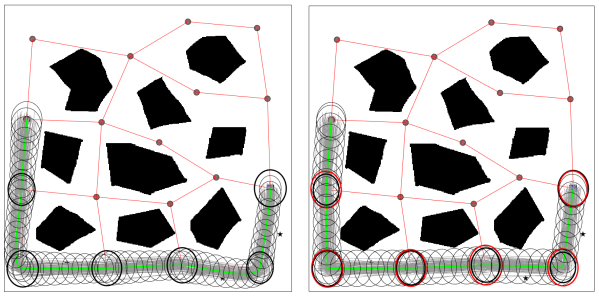


(a) (b)

Fig. 2: (a) Planning environment and sampled CS-BRM nodes. (b) CS-BRM roadmap by sampling multiple velocities at each position. The green lines are the mean trajectories between the nodes. The gray ellipses are the 3σ confidence intervals of the covariances of the positions.

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 is given in Figure 3(a). For comparison, SLQG-FIRM is used to solve the same problem. SLQG-FIRM uses the same belief nodes as in Figure 3. However,



(a) (b)

Fig. 3: (a) Path planned using CS-BRM; (b) Path planned using SLQG-FIRM.

the velocity at each node has to be zero. Each edge of SLQG-FIRM is constructed by a time-varying LQG controller and a stationary LQG controller The planned path is shown in Figure 3(b). The red ellipses are the state covariances at the last step of the time-varying LQG controller As seen in this figure, the ellipses are larger than the sampled state covariance (bold black ellipses). Thus, a converging step (stationary LQG) is required at every intermediate node. The time-varying LQG controller is switched to the SLQG controller until the state covariance converges to a value that is smaller than the sampled covariance. The cost of the planned paths in Figures 3(a) and 3(b) are 104.87 and 240.47, respectively. The decrease in the path cost is due to the decrease of the mean control cost. For SLQG-FIRM, the robot has to stop at every intermediate node. While for CS-BRM, the robot goes through each intermediate node smoothly, which results in a more efficient path.

B. Fixed-Wing Aircraft

The discrete-time system model of a fixed-wing aerial vehicle can be found in [24], [26]. The environment setting is similar to the 2-D double integrator example. The j^{th} position measurement corresponding to landmark j is

$$^{j}y = [x\ y\ z\ \phi]^{\top} + \eta d_j v, \quad j = 1, 2, \ldots, \ell, \tag{23}$$

where η is set to 0.05, and v is a 4-dimensional standard Gaussian random vector. The measurement vector y is a 4ℓ-dimensional vector, where ℓ is the number of landmarks.

The planned path is shown in Figure 4. Instead of flying through the narrow passage between the four obstacles, which resulting a high probability of collision, the algorithm chooses a path that trades off between control cost and collision cost while minimizing the total edge costs.

C. 3-D Double integrator

The final example is a 3-D double integrator robot in cluttered environment position cameras. The robot gets state measurement only if it is in the line-of-sight of one of the cameras. The planning environment along with the CS-BRM roadmap are shown in Figure 5(a). The red triangles are cameras. The gray polyhedron are obstacles. If the robot is in a position where all cameras are blocked by the obstacles, it receives no measurement and $C_k = 0$. The planed path is shown in Figure 5(b).

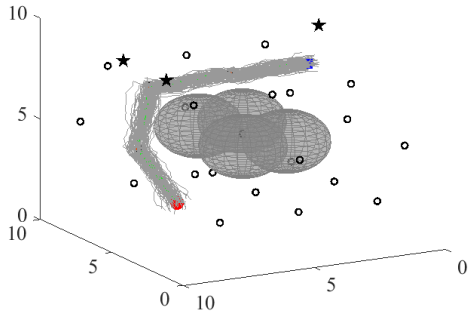
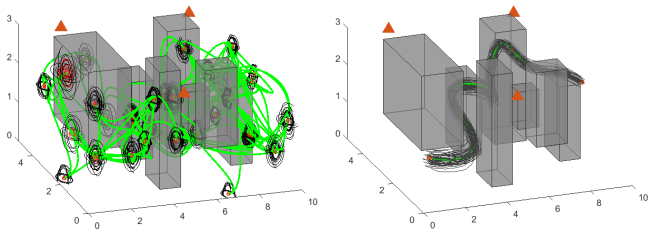


Fig. 4: Planning results using the CS-BRM for the fixed-wing vehicle. The four spheres are the obstacles. The black circles represent the state mean (position) of the nodes.



(a) (b)

Fig. 5: Planning results of the 3-D double integrator

VII. Conclusion

A 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 to design the edge controllers of the BRM graph to steer the system from one distribution to another. Compared to [8], the proposed method allows sampling non-stationary belief nodes, which has the advantage of more complete exploration of the belief space and finds low cost plans. For covariance steering of nonlinear systems, we introduce the concept of compatible nominal trajectories and 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.

References

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

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, pp. 846–894, June 2011.

[8] 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.

[9] 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.

[10] 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), pp. 1025–1032, August 2003.

[11] A. Bry and N. Roy, "Rapidly-exploring random belief trees for motion planning under uncertainty," in *2011 IEEE international conference on robotics and automation*, pp. 723–730, IEEE, 2011.

[12] T. Shan and B. Englot, "Belief roadmap search: Advances in optimal and efficient planning under uncertainty," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5318–5325, September 2017.

[13] 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.

[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] Y. Chen, T. T. Georgiou, and M. Pavon, "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), pp. 1153–1158, December 2016.

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

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

[20] J. Ridderhof, K. Okamoto, and P. Tsiotras, "Chance constrained covariance control for linear stochastic systems with output feedback," in *IEEE Conference on Decision and Control*, (Jeju Island, South Korea), pp. 1153–1158, Dec. 8–11 2020.

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

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

[23] A. E. Bryson and Y. C. Ho, *Applied Optimal Control: Optimization, Estimation and Control*. CRC Press, 1975.

[24] D. Zheng, J. Ridderhof, P. Tsiotras, and A. A. Agha-mohammadi, "Belief space planning: A covariance steering approach," *arXiv preprint arXiv:2105.11092*, 2021.

[25] 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.

[26] 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.), ch. 64, pp. 1677–1701, Springer, 2015.