Gaussian Belief Trees for Chance Constrained Asymptotically Optimal Motion Planning

Qi Heng Ho, Zachary Sunberg, and Morteza Lahijanian

***Abstract*— In this paper, we address the problem of sampling based motion planning under motion and measurement uncertainty with probabilistic guarantees. We generalize traditional sampling based tree-based motion planning algorithms for deterministic systems and propose belief-$\mathcal{A}$, a framework which extends any kinodynamical tree-based planner to the belief space for linear (or linearizable) systems. We introduce appropriate sampling techniques and distance metrics for the belief space that preserve the probabilistic completeness and asymptotic optimality properties of the underlying planner. We demonstrate the efficacy of our approach for finding safe low cost paths efficiently and asymptotically optimally in simulation, for both holonomic and non-holonomic systems.**

I. Introduction

In recent years, sampling-based algorithms (e.g., [1]–[4]) have emerged as powerful motion planning tools as they can search high dimensional spaces very efficiently and find solutions to complex problems quickly. They have enabled many applications such as self-driving cars, surgical robots, and autonomous UAVs. The majority of such planners assume perfect state information. However, an inseparable aspect of robotics is uncertainty in motion (e.g., due to modeling inaccuracies) and observation (e.g., due to noisy sensors). This challenge is especially significant when measurement uncertainty makes it difficult to find efficient paths while obeying safety constraints. This work focuses on this gap and aims to develop a general framework for adapting existing sampling-based algorithms to plan for uncertain systems with safety and optimality guarantees.

Consider a UAV equipped with a GPS receiver navigating under a canopy in a forest for a search and rescue task. The ability of the UAV to estimate its state is dependent on where it is in the forest, as some regions may have improved GPS accuracy or connectivity than others. To plan safe motion paths under dynamics and sensor noise, the robot has to tradeoff between visiting such regions or taking the shortest path to the goal location. Explicitly and intelligently accounting for both motion and measurement uncertainty is necessary to improve the quality and safety of motion plans.

In its most general form, planning under uncertainty can be formulated as a Partially Observable Markov Decision Process (POMDP) [5]. POMDPs solvers attempt to find an optimal policy in the belief space, which is the space of all possible probability distributions over the state space. Unfortunately, this problem is computationally intractable due to the curses of dimensionality and history [6]. Methods for approximating POMDP solutions for motion planning over continuous state, control, and observation spaces have been proposed, e.g., [7], [8]. However, they do not perform well especially when the action (control) space is large, which is a characteristic of robotics systems. Another drawback is the lack of guarantees in these solutions.

In recent years, sampling-based algorithms have been extended to account for motion uncertainty with probabilistic guarantees, e.g., [9]–[13]. Specifically, the chanced-constrained tree-based planners have shown to be efficient, and hence, employed in unknown environments via iterative planning [9]–[11]. A few studies extend those frameworks to systems with measurement uncertainty [14]–[16]. For instance, in [16], [17], maximum-likelihood observations are used to approximate solutions. While that approach works well in many cases, it lacks safety guarantees. Work [18] proposes an alternative approach that provides chance-constrained guarantees by combining optimal control and state estimation with sampling-based graphs. The method finds an optimal trajectory through the belief space by constructing a graph of trajectories in the state space and enumerating all possible uncertainty levels for these trajectories in the belief space. That planner is asymptotically optimal and has probabilistic safety guarantees, but it suffers from large computation times.

To mitigate that issue, work [19] extended the method by using branch-and-bound pruning, which leads to linear computation speedups. However, it also modifies the cost function of the problem to include uncertainty, which may affect the optimality of its solutions with respect to the original cost function.

In this paper, we generalize the tree-based motion planning framework to obtain a belief planning framework for linear (or linearizable) systems with both motion and observation uncertainty. The framework, called belief-$\mathcal{A}$, enables the use of any kinodynamical planner $\mathcal{A}$ for belief space motion planning. Belief-$\mathcal{A}$ provides probabilistic chance constraint guarantees, and preserves the probabilistic completeness and asymptotic optimality properties of the underlying planner as well as its computational efficiency. This is achieved by appropriate sampling techniques and distance functions in the belief space.

Specifically, our contributions are 1) a general method to extend state space sampling-based tree search algorithms directly to the Gaussian belief space, 2) a method to uniformly sample in the belief space and 3) a way to heuristically bias sampling towards low uncertainty belief states, and lastly 4) the use of the 2-Wasserstein metric for distance computation

Authors are with the department of Aerospace Engineering Sciences at the University of Colorado Boulder, CO, USA {*firstname.lastname*}@colorado.edu

between beliefs. We show the efficacy of our framework through benchmarking and numerical characterization in several scenarios that highlight the relative strengths of different aspects of our framework. Our results indicate that our method performs much faster (up to 50 times speedup) compared to prior belief space tree search methods while still having asymptotically near-optimal guarantees, allowing for safe and efficient planning. As a by-product of the computational speedups, we also show the ability of our planner to conduct online re-planning when knowledge of the environment changes.

II. Problem Formulation

We are interested in motion planning for a robotic system with uncertainty in both motion and observation (measurement) with safety guarantees. We assume the robot evolves in a bounded workspace ($\mathbb{R}^2$ or $\mathbb{R}^3$) with obstacles according to linear or linearizable dynamics and a measurement model that can change in different parts of the environment, e.g., GPS signal may be available only in parts of the environment. Below, we formalize this problem.

Let $\mathcal{X} \subset \mathbb{R}^n$ and $\mathcal{U} \subset \mathbb{R}^m$ be the state and control spaces, respectively. Then, the robot model is given by:

$$\begin{aligned} x_k &= Ax_{k-1} + Bu_{k-1} + w_k, \quad w_k \sim \mathcal{N}(0, Q), \\ z_k &= Cx_k + v_k(x_k), \quad v_k(x_k) \sim \mathcal{N}(0, R(x_k)), \end{aligned} \tag{1}$$

where $x_k \in \mathcal{X}$ is the state, $u_k \in \mathcal{U}$ is the input, $z_k \in \mathbb{R}^p$ is the measurement. Furthermore, matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, and $C \in \mathbb{R}^{n\times p}$. Terms w_k and v_k are i.i.d white Gaussian noise with zero mean and Q and $R(x_k)$ covariance matrices, respectively. Note that a unique aspect of this robot model is that covariance matrix is a function of the state, representing, e.g., various types of measurement regions in the environment.

We assume the system's dynamics are fully controllable and observable and the noise covariance matrices Q and $R(x_k)$ are non-degenerate. For presentation simplicity, we focus on the time-invariant system in (1), but the formulation can be easily extended to time-varying systems.

The evolution of the robotic system in (1) can be described by a discrete-time Gaussian Markov process. The robot state x_k at each time step can be described as a Gaussian distribution $b_k = \mathcal{N}(\hat{x}_k, \Sigma_k)$, where $b_k \in \mathcal{B}$ is referred to as the belief of the robot state, $\mathcal{B}$ is the *belief space* of the robot, and $\hat{x}_k$ and Σ_k are the state mean and covariance matrix, respectively. In this work, we search for a policy characterized by a sequence of nominal control inputs and the nominal trajectory that they would produce in the absence of uncertainty, along with a linear feedback controller that seeks to follow this trajectory using an online state estimate $\hat{x}$.

Let $\check{U}^{0,t} = (\check{u}_0, \check{u}_1, \check{u}_2, \cdots, \check{u}_{t-1})$ be a sequence of control inputs. Given an initial state x_0 and nominal system dynamics $\check{x}_{k+1} = A\check{x}_k + B\check{u}_k$, a nominal trajectory $\check{X}^{x_0,x_t} = (\check{x}_0, \check{x}_1, \check{x}_2, \cdots, \check{x}_t)$ is obtained. This motion plan is then executed online via a stabilizing controller given online state estimates $\hat{x}_k$. This gives us the following online control input

$$u_k = \check{u}_{k-1} - K(\hat{x}_k - \check{x}_k), \tag{2}$$

where K is the closed loop gain. For simplicity, we use a fixed controller gain, but a variable gain can also be used. We refer to $(\check{U}, \check{X})$ as a *motion plan* for System (1).

Let $\mathcal{X}_{obs}, \mathcal{X}_{goal} \subset \mathcal{X}$ represent the state space obstacles and the goal region respectively. For robot state x_k and its belief b_k, the probability of the robot being in $\mathcal{X}_i$, where $i \in \{obs, goal\}$, is given by

$$p(x_k \in \mathcal{X}_i) = \int_{\mathcal{X}_i} b_k(y)dy. \tag{3}$$

Furthermore, consider a stage cost function $J : \mathcal{X} \to \mathbb{R}^+$ that is monotonic and Lipschitz continuous. Given safety probability P_{safe}, we desire an optimal motion plan that minimizes total cost given by J and whose probability of collision at every time step and probability of not ending in goal are upper bounded by P_{safe}. The formal statement of this problem is as follows.

Problem 1. *Given a robot with noisy dynamics and measurements as in* (1)*, a set of state space obstacles $\mathcal{X}_{obs}$, a goal region $\mathcal{X}_{goal}$, a Lipschitz continuous cost function $J : \mathcal{X} \to \mathbb{R}^+$, and a safety probably bound P_{safe}, find a motion plan $(\check{U}, \check{X})^*$ as a pair of sequence of nominal controls $\check{U} = (\check{u}_0, \ldots, \check{u}_{T-1})$ for some $T \geq 1$ and its resulting nominal trajectory $\check{X} = (\check{x}_0, \ldots, \check{x}_T)$ that minimizes the expected total cost,*

$$(\check{U}, \check{X})^* = \arg \min_{(\check{U}, \check{X})} \mathbb{E}\Big[\sum_{t=0}^{T} J(x_t)\Big], \tag{4}$$

subject to, when executed via controller in (2)*, the probabilities of collision with $\mathcal{X}_{obs}$ and not ending in $\mathcal{X}_{goal}$ are bounded by P_{safe}, i.e.,*

$$p(x_t \in \mathcal{X}_{obs}) < P_{safe}, \quad \forall t \in [0, T], \tag{5}$$

$$p(x_T \notin \mathcal{X}_{goal}) < P_{safe}. \tag{6}$$

To solve this optimal motion planning problem, we aim to design an algorithm that is asymptotically optimal, i.e. the probability of finding the optimal solution converges to 1 as the number of iterations approaches infinity. For our applications of interest, asymptotic near-optimality (i.e. the solution converges to within an approximation factor ϵ of the optimal solution) is sufficient, but our proposed framework is not limited to asymptotic near-optimality.

III. Preliminaries

Here, we introduce the preliminaries required to introduce our solution to the above problem.

A. Asymptotically-Optimal Tree Search Planners

Typical single query sampling-based tree search algorithms construct a tree in the search space. For kinodynamical systems without any uncertainty, this search space is the state space. In this work, we aim to develop a method to transform such existing such planners into belief space planners that solves Problem 1. To this end, we first present an overview of such planners.

Algorithm 1 shows a generic form of an asymptotically optimal tree-based planner for kinodynamical systems. It takes state space $\mathcal{X}$, input space $\mathcal{U}$, goal region $\mathcal{X}_{goal}$, obstacle regions $\mathcal{X}_{obs}$, an initial state x_{init}, and a maximum planning time or iteration count as input and returns a near-optimal solution, if one is found. Search is performed by growing a motion tree in which states and the connections between them are stored as nodes and edges, respectively. Note that the Prune() subroutine is not mandatory for asymptotic optimality, but many asymptotically optimal planners, such as the Stable Sparse-RRT (SST) [20] use it.

Algorithm 1: Generic Asymptotically-Optimal Tree-based Planner $\mathcal{A}$ ($\mathcal{X}, \mathcal{U}, \mathcal{X}_{goal}, \mathcal{X}_{obs}, x_{init}$, N)

Output: Valid Trajectory $x_{1:T}$ if one is found
$G = (\mathbb{V} \leftarrow \{x_{init}\}, \mathbb{E} \leftarrow \emptyset)$
for *N iterations* **do**
    $x_{rand} \leftarrow$ Sample()
    $n_{select} \leftarrow$ Select(x_{rand})
    $n_{new} \leftarrow$ Extend($n_{selected}$)
    if *isValidPath(n_{select}, n_{new})* **then**
        $\mathbb{V} \leftarrow \mathbb{V} \cup \{n_{new}\}$
        $\mathbb{E} \leftarrow \mathbb{E} \cup \{edge(n_{select}, n_{new})\}$
    Prune($\mathbb{V}, \mathbb{E}$)
return $G = (\mathbb{V}, \mathbb{E})$

B. Propagation of Beliefs through Feedback Dynamics

Following [18], given a stabilizing feedback controller (2) and online state estimate $\hat{x}_k$, the system closed loop dynamics are:

$$x_k = Ax_k + B(\check{u}_{k-1} - K(\hat{x}_k - \check{x}_k)) + w_k. \tag{7}$$

The belief during planning is parameterized as follows:

$$b_k = P(x_k) = \mathcal{N}(\check{x}, \Sigma_k^+ + \Lambda_k^+). \tag{8}$$

This belief accounts for both the online state estimation error $\Sigma_k +$ and the uncertainty over possible state estimates that could arise during execution, Λ_k^+, that is a result of considering all possible measurements received. Σ_k^+ and Λ_k^+ can be computed recursively using the Kalman Filter:

$$\Sigma_k^- = A\Sigma_{\hat{x}_{k-1}}A^T + Q, \tag{9}$$
$$L_k = \Sigma_k^- C^T (C\Sigma_k^- C^T + R(\hat{x}_k)), \tag{10}$$
$$\Sigma_k^+ = \Sigma_k^- - L_k C \Sigma_k^-, \tag{11}$$
$$\Lambda_k^+ = A_{cl}\Lambda_{k-1}^+ A_{cl}^T + L_k C \Sigma_k^-. \tag{12}$$

IV. Gaussian Belief Trees

Here, we present our methodology for finding chance-constrained motion plans in the belief space to solve Problem 1. We present our framework in its general form, which can be used with any single-query state space tree-based motion planner with a structure similar to Algorithm 1.

Instead of planning in the state space, we plan in belief space $\mathcal{B}$, where $b \in \mathcal{B}$ is a Gaussian distribution as defined in (8). Vertices in the tree structure are now belief nodes, rather than state nodes.

As discussed in Section III-A and Algorithm 1, the main subroutines for tree search algorithms are the sample, select, extend and validity checking subroutines. This section presents methods for reformulating these functions for reasoning in the belief space. Specifically, we show how to sample directly in the belief space, and propose a method for heuristically biasing the sampling. Then, we present a belief space distance function for node selection through the use of a suitable distance metric for probability distributions. Finally, we present our approach to belief propagation and validity checking.

A. Sampling Strategy

Since our beliefs are Gaussian distributions, the Sample function must randomly sample the first moment (mean μ_s) and second moment (covariance matrix Σ_s) to obtain a sample belief.

1) Sampling of first moment: Similar to state space planning, we sample the mean of a belief within the state space, such that $\mu_s \in \mathcal{X}$. We also utilize a goal bias, in which we sample μ_s within the goal region with a probability p_{goal}, which has been shown to yield faster convergence in practice.

2) Sampling of covariance matrices: The optimal sampling strategy for motion planning in Gaussian belief spaces is an open question, but it is crucial that the method for sampling covariance matrices completely samples the space of covariances for a problem environment in order to preserve probabilistic completeness of the state space version of the algorithm. We show how to uniformly sample the space of covariance matrices for a given environment.

We first observe that the covariance matrix of a Gaussian distribution is positive semi-definite. Hence it is always diagonalizable, i.e. we can write $\Sigma_s = ODO^T$, where D is a $n \times n$ diagonal matrix whose diagonal elements consist of its non-negative eigenvalues and O is a $n \times n$ orthogonal matrix whose columns consist of its linearly independent real and orthonormal eigenvectors. Therefore, we can sample Gaussian covariance matrices by generating eigenvalues for D and an orthogonal matrix O of eigenvectors.

To do this, we first uniformly sample eigenvalues λ_i from the interval $(0, \lambda_{i,max}]$ where $\lambda_{i,max}$ defines the maximum allowed eigenvalue for the i^{th} dimension, forming a diagonal matrix of eigenvalues. $\lambda_{i,max}$ depends on the environment. One method for setting $\lambda_{i,max}$ is finding the minimum $\lambda_i > \lambda_{i,max}$ such that forming a diagonal covariance matrix with λ_i as the i^{th} diagonal element and setting the rest of the diagonal elements as a small ϵ violates the chance constraint at every state in the state space.

Next, to uniformly sample random orthogonal matrices O, we first generate an $n \times n$ matrix M whose elements are independently sampled from a standard normal distribution. The QR decomposition of M ,i.e. $M = OR$, then provides a uniform distribution of an orthogonal matrix O[1] (see [21] for a proof). Finally, we obtain our randomly sampled Gaussian covariance matrix by computing $\Sigma_s = ODO^T$.

3) Sampling bias: Instead of uniformly sampling eigenvalues, we make use of the intuition that having low uncertainty is generally advantageous for finding solution paths compared to high uncertainty. This is because for a given mean of a belief, a lower uncertainty effectively lowers the probability of collision, which provides a higher probability of finding new valid edges and nodes. Additionally, since the problems we are interested in have an inherent tradeoff between motion plans that have lower cost and those that have higher cost but lead to lower uncertainty trajectories, this low uncertainty bias provides a way to prioritize lower uncertainty paths without explicitly adding it into the cost function. For example, trajectories that deviate from the shortest path to visit measurement regions will have longer path length but leads to paths that have a higher probability of satisfying the chance constraints. A low uncertainty bias leads to faster convergence in such cases.

Therefore, with probability p_{bias}, we sample a fixed low eigenvalue $\lambda_{i,low}$ instead of uniformly in $(0, \lambda_{i,max})$. This effectively pulls the tree towards low uncertainty areas in the belief space, converging to good solutions more efficiently as shown empirically in our experiments while still ensuring completeness. The optimum p_{bias} depends on the environment, where a high p_{bias} is useful for environments in which solution paths require low uncertainty but may slow down convergence if low uncertainty is not required. In the general case, p_{bias} should be set to a low value to ensure a more general coverage of the search space.

B. Belief Space metric

A key component of many tree-based algorithms, such as RRT [2] and SST [20], is a distance metric to compute distances between nodes in the tree. This distance function is used for near neighbor computations, to select which node is to be extended in the Select() function, and for pruning. A good choice of distance metric can greatly affect coverage of the search space, planning time and path quality [?], [22].

In the belief space, since nodes are belief states, a typical distance metric such as Euclidean distance can still be used on the means of the nodes, but that is not ideal as they do not capture the uncertainty of the belief state. Instead, we use the Wasserstein distance, which is a metric on a space of probability measures. The Wasserstein metric is the amount of work required to move the probability mass of one distribution to another. For two non-degenerate Gaussian beliefs b_1 and b_2 with $b_i \sim N(\mu_i, \Sigma_i)$, with respect to the Euclidean norm on $\mathbb{R}^n$, the 2-Wasserstein distance is given by

$$D_{W^2}(b_1, b_2) = \|\mu_1 - \mu_2\|_2^2 + Tr(\Sigma_1 + \Sigma_2 - 2(\Sigma_1^{\frac{1}{2}}\Sigma_2\Sigma_1^{\frac{1}{2}})^{\frac{1}{2}}), \tag{13}$$

where $Tr(M)$ is the trace of square matrix M. The Wasserstein metric is a true metric in the belief space, as compared to other distance functions like KL-Divergence and LP-distance which are not true metrics, and has desirable properties for the preservation of asymptotic optimality of the state space algorithm (see Section V-A).

C. Extend/Propagate

After selecting a belief node, using the sampling method and distance metric described above, kinodynamic tree-search algorithms create a new node in the tree by extending the selected node using a computed nominal control input and random time duration. Given a belief node b defined by $\check{x}$ and (Σ and Λ), nominal control inputs $\check{u}$ and a time duration, we propagate the belief to a new belief node b' using (8) - (12). The control inputs can be computed by either repeatedly sampling the control space directly or sampling a state x_{sample} and applying a closed loop controller for a sampled time duration.

Note that although the system is linear which implies that we can always compute a control input to steer a state node to any other state nodes, steering between two Gaussian belief nodes involves not only steering the means from one belief to another, but propagation from one covariance to another. Even for a linear system, a solution for controlling an arbitary Gaussian covariance to converge to another arbitary covariance is not guaranteed to exist. Therefore, steering between two belief nodes is highly nontrivial, and non-kinodynamic tree-search algorithms that rely on a steering function between two nodes, such as RRT*, cannot be readily extended to the belief space.

D. Validity Checking

Validity checking in the state space involves the computation of collision, i.e. checking if $x \in \mathcal{X}_{obs}$. In the belief space, we are required to compute the probability of collision $p(x) \in \mathcal{X}_{obs}$ in (3), which involves an integral. The exact evaluation of this integral with obstacles is not computable, and numerical integration is computationally expensive. Efficient conservative approximations can be computed very quickly for convex polygonal obstacles using error functions as proposed in [10]. For non-convex obstacles, methods such as [11], [23] can be used with high accuracy at the cost of computation time.

[ML: Need a sentence or two to wrap up this section.]

V. Analysis

A. Completeness and Optimality

The belief space extension presented inherits the probabilistic completeness and asymptotic (near)-optimality properties of the state space versions of the algorithms. The following theorems formalize these properties.

Theorem 1 (Completeness)**.** *If the underlying state space algorithm $\mathcal{A}$ is probabilistically complete under the nominal conditions in* (1)*, algorithm belief-$\mathcal{A}$ is also probabilistically complete.*

[1]Note that sampling elements of M with another distribution, e.g. uniform, would still randomly samples an orthogonal matrix, but it would not be uniformly distributed

Proof Sketch. Probabilistic completeness for tree-search algorithms requires that the search space is completely sampled almost surely. The main relevant consequence of the change to the belief space is the difference in sampling of the search space and propagation dynamics. However, with the sampling method described in IV-A, beliefs are sampled uniformly, which implies the search space is completely sampled almost surely (the inclusion of the bias term does not change the result). Additionally, if the dynamics is Lipschitz continuous, the propagation method presented in the belief space is also Lipschitz continuous. Hence, Belief-$\mathcal{A}$ inherits the probabilistic completeness properties of $\mathcal{A}$. □

The use of the 2-Wasserstein metric makes the new belief space a proper metric space, and it is known that the 2-Wasserstein metric preserves the Lipschitz continuity of the cost function [22], [24].

Lemma 1 ([22])**.** *For any control input $u \in \mathcal{U}$, if the cost function in the state space is Lipschitz continuous in the state space, then the expected cost function in the belief space with the 2-Wasserstein metric is also Lipschitz continuous.*

Convergence guarantees for asymptotically ($\epsilon-$)optimal sampling based planners typically rely on Lipschitz continuity in the cost function [**?**]. This motivates the following theorem:

Theorem 2 (Optimality)**.** *If the cost function is Lipschitz continuous and the underlying state space algorithm $\mathcal{A}$ is asymptotically (ϵ-)optimal, belief-$\mathcal{A}$ is also asymptotically (ϵ-)optimal.*

Proof Sketch. Since the cost function is Lipschitz continuous in the state and control space, by Lemma 1, the cost function in the belief space under the 2-Wasserstein metric is also Lipschitz continuous. For belief-$\mathcal{A}$, the control space, method to sample controls and node selection procedure remains the same as $\mathcal{A}$. Hence, Belief-$\mathcal{A}$ inherits the asymptotic (ϵ-)optimality properties of $\mathcal{A}$. □

B. Computational Efficiency

The overall time complexity for the underlying tree search algorithm does not change. However, planning in the belief space increases the computational effort in two ways. Firstly, planning in the belief space changes the dimension of the search space from n state dimensions to $n + n^2$ Gaussian belief dimensions. Additionally, our extensions change specific subroutines, giving a computation time increase per call.

However, the use of the Wasserstein metric for nearest neighbor queries is slower than usual state space metrics. For general belief distributions, the Wasserstein metric has a time complexity of $O(N^3 logN)$ for N bin histograms. However, since our beliefs are Gaussian distributions, the Wasserstein distance computation in (13) is more efficient, $O(n^3)$. This is compared to state space distance computations which are $O(n)$. Similarly, we analyze the time complexity of sampling in this metric space. The time complexity of sampling random $n \times n$ matrix is $O(n)$, and that of QR decomposition is $O(n^3)$. Therefore, we have an overall time complexity for the sampling subroutine of $O(n^3)$, which is compared to state space sampling of $O(n)$.

Therefore, the main changes in the computation time of the belief-$\mathcal{A}$ compared to $\mathcal{A}$ is the increased dimensionality of the state space, and a fixed higher computational cost of the above subroutines per call. For example, for a 2 dimensional system if $\mathcal{A}$ is RRT with time complexity of $O(\mathtt{n} \log \mathtt{n} + \mathtt{n} \log^d \mathtt{n})$, where $\mathtt{n}$ is the number of tree nodes and $d = 2$ is the search space dimension, belief-$\mathcal{A}$ has the same time complexity but with $d = n + n^2 = 6$. Although this is more computationally expensive than usual state space sampling, since n is fixed and does not grow with the number of iterations, this leads to a bounded computational time increase per iteration. Empirically, as seen in the experiments, the belief space extension subroutines are reasonably efficient, allowing for fast planning in the belief space. We also compared to other approaches in the literature which solve similar problems in the belief space, in which our method is shown to have superior computational efficiency.

VI. Experiments and Results

A. Time and Path Length Benchmarks

We compare our methodology on two main systems, a 2D system with dynamics described by:

$$x_t = x_{t-1} + u_{t-1} + w_t, \quad z_t = x_t + v_t$$

and a second order unicycle system with dynamics described by:

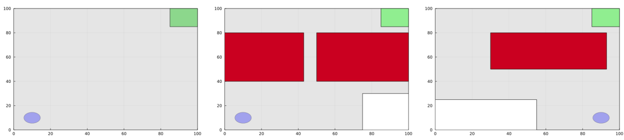
$$\dot{x} = v\cos(\phi), \quad \dot{y} = v\sin(\phi), \quad \dot{\phi} = \omega, \quad \dot{v} = a$$

We use dynamic feedback linearization to obtain a linearized closed loop model as presented in [25].

For each of these systems, we compared them in specific environments that forces a trade-off between motion plans that visit measurement regions to localize and those that correspond to shorter path lengths. These environments, depicted in Fig. 1, are

1) Non-observable environment with no obstacles and no measurement region.
2) Narrow passage environment with a measurement region. Moving straight to the goal will not satisfy the chance constraint.
3) Multi mode solution environment. Two main solution paths are available - A larger passageway that allows reaching the goal region without measurements, and a narrow passage that requires a lower uncertainty.

We compared the effect of using the Wasserstein metric (W_2) with the Euclidean metric (l_2), and the effect of the inclusion of belief sampling bias p_{bias} for belief-RRT and belief-SST. We also ran benchmarks against the Rapidly-exploring Random Belief Trees (RRBT) in [18], a single query RRG-based optimal planner that also attempts to solve similar problems. For each case, we use $p_{goal} = 0.05$ and $P_{\text{safe}} = 0.95$, and probabilistic validity checking is performed using the chance constraint formulation described in [10]. All algorithms are implemented on the Open Motion



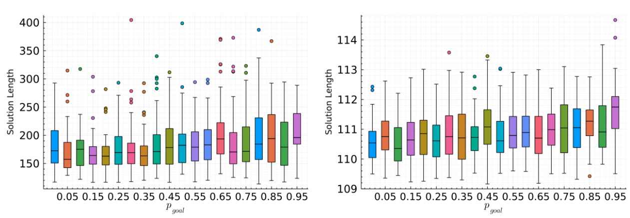
(a) Environment 1 (b) Environment 2 (c) Environment 3

Fig. 1: The initial belief, goal region and obstacles are represented as blue, green and red respectively. White region are measurement regions, in which the measurement noise covariance $R(x_k) = 0.01I$ in those region, and with no measurements everywhere else (gray). Environments are $100 \times 100(m^2)$.

Planning Library (OMPL) [26]. The average of 100 trials of is reported.

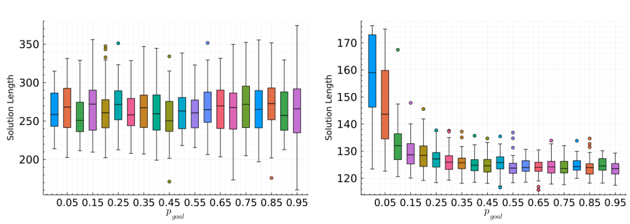
Table I shows the time taken to compute a first solution, the cost of the first solution, and the solution cost after 10 seconds for each of the environments and dynamics. As seen in the table, belief-SST consistently finds initial solutions very quickly and given additional time, is able to optimize solutions to low path costs. Using the Euclidean metric instead of the Wasserstein metric computes each iteration faster, but suffers from an inability to find solutions quickly in narrow passageway situations due to the pruning subroutine causing lower uncertainty but higher cost paths to repeatedly be pruned. Belief-RRT finds solutions the fastest but does not improve solutions much when given more time, which is expected as RRT is not asymptotically optimal.

Lastly, RRBT computes high quality initial solutions but takes a very long time to do so. This is to be expected, as RRBT exhaustively propagates candidate solutions in the belief space from a state space RRG, and so solution paths once found already have low cost. However, RRBT maintains an excessive number of belief nodes for each state RRG vertex that get propagated to a new vertex at each iteration, which slows down computation immensely.



(a) Initial solution. (b) After 10 seconds.

Fig. 2: Effect of varying p_{bias} on solution length for environment 1.



(a) Initial solution. (b) After 5 seconds.

Fig. 3: Effect of varying p_{bias} on solution length for environment 3.

B. Sampling Bias

We study the effect of different p_{bias} bias values for the sampling of low uncertainty covariances on solution length and time. The performance of varying bias values with SST is depicted in Figures 2-3. The addition of p_{bias} has a direct effect on the rate of convergence to low solution costs. In environment 3, it is clear that a higher p_{bias} leads to faster convergence to better solutions. However, there is a small tradeoff in solution cost for environment 1, in which low uncertainty is not needed. In general, a low value such as $p_{bias} = 0.2$ balances the tradeoff well, leading to large gains for environments that require low uncertainty while also performing well in those that do not.

C. Belief-SST with covariance sampling bias

Fig. 4 shows an example of motion plans computed by belief-SST for environment 3 with the feedback linearized unicycle. Given a time limit of 1 second, belief-SST quickly finds an initial solution through the larger passageway. As the time limit increases, belief-SST is able to iteratively improve the solution, converging to one with lower cost by visiting the measurement region to localize before finding a safe path through the smaller passageway.

D. Online planning demonstration

Due to the improved speed of computation, our algorithm is able to plan in an online fashion. We show a simple example which showcases the effectiveness of our algorithm for replanning in real time when changes to the state space is detected. These changes can be the detection of new static obstacles or measurements that are previously unaccounted for.

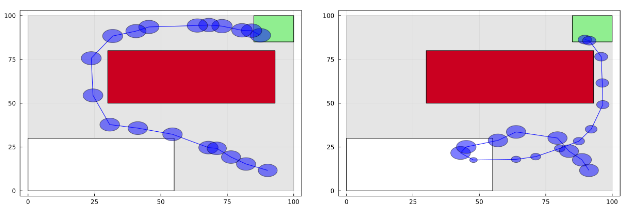
Fig. 5 shows an example of the online replanning capabilities of our planner due to the computational efficiency of the method. In this example, the robot initially plans with the knowledge that there is one obstacle in red, and finds a motion plan that reaches the goal region. Note that in this plan, there is no need to go to the measurement region since the chance constraint can be satisfied without doing so. However, as the robot moves along the motion plan, it updates the environment after detecting the presence of obstacles and re-plans. In this updated environment, moving directly to the goal will not satisfy the chance constraint. Therefore, the robot finds a solution plan that moves to the measurement region to localize before passing safely through the obstacles to the goal region.

VII. Conclusion

This paper proposes belief-$\mathcal{A}$, a general framework for converting any kinodynamical tree based planner for linear systems to the belief space. The proposed belief space extension method preserves the convergence properties of the underlying algorithm, and has been shown to be computationally efficient empirically. While we have demonstrated the efficiency and of our framework through simulation, future work includes implementing the framework on an actual robot and extending the method to work with a broader class of motion planning algorithms.

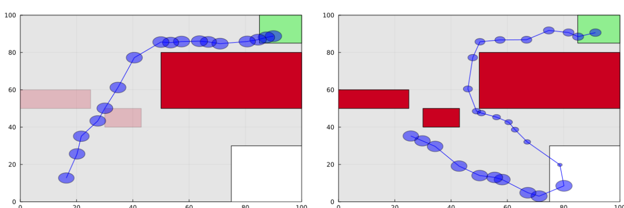
	Environment 1			Environment 2			Environment 3		
Algorithm	Time (1st)	Cost (1st)	Cost (10s)	Time (1st)	Cost (1st)	Cost (10s)	Time (1st)	Cost (1st)	Cost (10s)
2D System									
RRBT	0.17	**113.8**	**109.8**	1.56	**216.1**	**199.2**	2.24	**146.8**	**121.2**
B-RRT (l_2)	**0.003**	194.8	192.6	**0.024**	360.8	341.7	**0.009**	293.0	292.6
B-RRT (W_2)	0.029	195.7	186.1	0.2	353.7	345.1	0.016	288.6	285.3
B-RRT (W_2, $0.2\, p_{bias}$)	0.029	189.8	185.8	0.071	348.4	340.0	0.016	294.9	293.8
B-SST (l_2)	**0.004**	163.6	**111.8**	2.62	259.4	236.1	**0.008**	264.9	166.9
B-SST (W_2)	0.029	181.3	**111.1**	0.51	295.2	236	0.043	263.9	157.4
B-SST (W_2, $0.2\, p_{bias}$)	0.027	168	**110.2**	0.152	304.7	**204.9**	0.041	263.2	**128.6**
Linearized Unicycle									
RRBT	0.225	**123.9**	**120**	15.4	**227.1**	**222.6**	8.95	**201.6**	199.4
B-RRT (l_2)	**0.011**	250.0	250.0	**0.447**	399.4	367.7	0.045	391.9	390.9
B-RRT (W_2)	0.034	247.6	221.0	0.577	286.4	256.3	0.2	341.8	268.3
B-RRT (W_2, $0.2\, p_{bias}$)	0.041	244.8	231.6	0.61	287.0	260.8	0.19	332.5	289.3
B-SST (l_2)	0.013	198.7	138.5	7.08	267.4	**237.9**	**0.035**	339.3	**177.8**
B-SST (W_2)	0.055	190.3	137.2	2.19	277.8	**232.1**	0.193	347.2	188.7
B-SST (W_2, $0.2\, p_{bias}$)	0.054	193.8	140.4	0.88	274.5	**231.1**	0.153	316.1	**164.4**

TABLE I: Benchmark planner performance results. Each entry is the mean of 100 simulations. Quantification of the standard error may be found in the appendix of the full version of the paper and is generally significantly less than the variation between rows. The scores within 10% of the best score in each column are bolded.



(a) 1 second planning time (b) 5 second planning time

Fig. 4: Runtime results of belief-SST with Wasserstein metric and $p_{bias} = 0.02$, with Gaussian covariance shown as ellipses.



(a) Initial Plan (b) 1 second re-plan

Fig. 5: Online planning scenario. The robot is initially unaware of the two smaller obstacles on the left, and plans according to the red obstacle on the right (a). After updating the environment, the robot finds a safe path by localization at the measurement region in white (b).

REFERENCES

[1] 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*, 12(4):566–580, 1996.

[2] S. LaValle. Rapidly-exploring random trees : a new tool for path planning. *The annual research report*, 1998.

[3] D. Hsu, J.-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. In *Proceedings of International Conference on Robotics and Automation*, volume 3, pages 2719–2726 vol.3, 1997.

[4] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research*, 30(7):846–894, 2011.

[5] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[6] Christos H. Papadimitriou and John N. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.

[7] Zachary Sunberg and Mykel J. Kochenderfer. Online algorithms for POMDPs with continuous state, action, and observation spaces. In *International Conference on Automated Planning and Scheduling*, 2018.

[8] Neha P. Garg, David Hsu, and Wee Sun Lee. DESPOT-α: Online POMDP planning with large state and observation spaces. In *Robotics: Science and Systems*, 2019.

[9] B. Luders, M. Kothari, and J. How. Chance constrained RRT for probabilistic robustness to environmental uncertainty. *AIAA guidance, navigation, and control conference*, 2010.

[10] Lars Blackmore, Masahiro Ono, and Brian Williams. Chance-constrained optimal path planning with obstacles. In *IEEE Trans. on Robotics*, volume 27, pages 1080–1094. IEEE, 2011.

[11] Éric Pairet, David Hernández, Juan, Marc Carreras, Yvan Petillot, and Morteza Lahijanian. Online mapping and motion planning under uncertainty for safe navigation in unknown environments. In *arXiv preprint arXiv:2004.12317*, 2020.

[12] Ernst Moritz Hahn, Vahid Hashemi, Holger Hermanns, Morteza Lahijanian, and Andrea Turrini. Interval markov decision processes with multiple objectives: From robust strategies to pareto curves. *ACM Trans. Model. Comput. Simul.*, 29(4), November 2019.

[13] Wolfram Burgard, Oliver Brock, and Cyrill Stachniss. *The Stochastic Motion Roadmap: A Sampling Framework for Planning with Markov Motion Uncertainty*, pages 233–240. 2008.

[14] Ali-Akbar Agha-Mohammadi, Suman Chakravorty, and Nancy M Amato. FIRM: Sampling-based feedback motion-planning under motion uncertainty and imperfect measurements. *International Journal of Robotics Research*, 33(2):268–304, 2014.

[15] Tixiao Shan and Brendan Englot. Belief roadmap search: Advances in optimal and efficient planning under uncertainty. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5318–5325, 2017.

[16] Samuel Prentice and Nicholas Roy. The belief roadmap: Efficient planning in belief space by factoring the covariance. *The International Journal of Robotics Research*, 28(11-12):1448–1465, 2009.

[17] Kaelbling L Platt R, Tedrake R and Lozano-Perez T. Belief space planning assuming maximum likelihood observations. In *Robotics: Science and Systems*. IEEE, 2010.

[18] Adam Bry and Nicholas Roy. Rapidly-exploring random belief trees for motion planning under uncertainty. In *Proc. IEEE Int. Conf. on Robotics & Automation*. IEEE, 2011.

[19] Hyunchul Yang, Jongwoo Lim, and Sung eui Yoon. Anytime rrbt for handling uncertainty and dynamic objects. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016.

[20] Y. Li, Z. Littlefield, and K. E. Bekris. Asymptotically optimal sampling-based kinodynamic planning. *International Journal of Robotics Research*, 35(5):528–564, 2016.

[21] M. L. Eaton. Multivariate statistics : a vector space approach. *Journal of the American Statistical Association*, 80:1069, 1985.

[22] Z. Littlefield, D. Klimenko, H. Kurniawati, and K.E. Bekris. The importance of a suitable distance function in belief-space planning. In *Robotics Research. Springer Proceedings in Advanced Robotics*, volume 3. Springer, 2018.

[23] J.S. Park, Chonyon Park, and Dinesh Manocha. Efficient probabilistic collision detection for non-convex shapes. In *Proc. IEEE Int. Conf. on Robotics & Automation*. IEEE, 2017.

[24] Hanna Kurniawati and Nicholas M. Patrikalakis. Point-based policy transformation: Adapting policy to changing pomdp models. In *Algorithmic Foundations of Robotics X*, pages 493–509, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[25] Alessandro De Luca, Giuseppe Oriolo, and Marilena Vendittelli. Stabilization of the unicycle via dynamic feedback linearization. *IFAC Proceedings Volumes*, 33(27):687–692, 2000. 6th IFAC Symposium on Robot Control (SYROCO 2000), Vienna, Austria, 21-23 September 2000.

[26] Ioan A. Şucan, Mark Moll, and Lydia E. Kavraki. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, December 2012. `https://ompl.kavrakilab.org`.